Cisco Video Surveillance Media Server User Guide

Release 6.2

Americas Headquarters
Cisco Systems, Inc.
170 West Tasman Drive
San Jose, CA 95134-1706
USA
http://www.cisco.com
Tel: 408 526-4000
800 553-NETS (6387)
Fax: 408 527-0883

Text Part Number: OL-19678-01

CONTENTS

Preface

Overview

This document provides information about installing and configuring the Cisco Video Surveillance Media Server.

Organization

This manual is organized as follows:

Chapter 1, “Overview”	Describes the Cisco Video Media Server and provides an overview of the Video Surveillance System hardware and software components.
Chapter 2, “Proxy Commands”	Provides an overview and instructions for running specific devices acting as sources for encoder or IP cameras.
Chapter 3, “Archive Commands”	Describes archive commands that permit the recording, storage, and management of resources for audio and video archives.
Chapter 4, “Event Commands”	Describes event commands structured around the ability of client devices to send alerts to VSMS.
Chapter 5, “AXClient API”	Explains how to use VSMS APIs for archive controls such as play, pause, seek and live controls such as pan, tilt, zoom, and presets.
Chapter 6, “Interactive Media Clients”	Explains how to use the Interactive Media Client (IMC), a group of ActiveX Controls that display video and control cameras.
Chapter 7, “Camera Control API”	Explains how VSMS camera controls permit clients to configure and control different types of cameras via the network.
Chapter 8, “DVR Integration”	Discusses the DVR Gateway and how to integrate 3rd party DVR devices into the Video Surveillance Manager suite.

Chapter 9, “Legacy Client Applets”	Discusses the minimum required NAME/VALUE pairs to view videos.
Chapter 10, “Advanced Configurations”	Explains advanced security setup configurations permitting VSMS, VSVM and VSOM to work together.
Chapter 11, “On Demand Viewer/Media Out”	
Chapter 12, “Command Line Tools”	Discusses Command Line Tools utilities used to manage the event database and archive repositories.
Chapter 13, “SNMP Configuration”	Explains how SNMP provides a way to monitor and control network devices, manage configurations, performance, security, and system resources such as disk space, CPU utilization, and Ethernet packets.

Obtaining Documentation, Support, and Security Guidelines

For information about obtaining documentation, support, providing documentation feedback, security guidelines, and recommended aliases and general Cisco documents, see the monthly *What's New in Cisco Product Documentation*, which also lists all new and revised Cisco technical documentation, at:

http://www.cisco.com/en/US/docs/general/whatsnew/whatsnew.html

For links to Cisco Video Surveillance documentation, go to

http://www.cisco.com/en/US/support/index.html, click the Physical Security link, and select your product.

The list of supported devices is available at
http://<*mediaserver*>/doc/BMS/source_types.html
where <*mediaserver*> is the host name or IP address of the media server (VSMS).

CHAPTER 1

Overview

What's New

This section describes the new Cisco Video Surveillance features. A list of supported devices is available at http://<*mediaserver*>/doc/BMS/source_types.html, where <*mediaserver*> is the host name or IP address of the media server (VSMS).

The following features are new in Cisco VSM 4.2/6.2:

- Video startup performance—Reduces the playback start-up time for multiple video streams by starting all streams in parallel.
- Pelco D driver updates—Support added for PTZ Patterns and On-screen Programming (OSP).
- Cisco high definition IP camera driver updates—HTTPS has been implemented for commands that are sent to configure Cisco Video Surveillance IP camera high definition models.
- Cisco standard definition IP camera driver updates—Supports existing firmware versions and the new firmware version that includes the Cisco Media API. The new firmware version enables motion detection, event triggers, and other features.
- Seeking—Seeking within archives has been improved.
- Driver pack consolidation—This release consolidates the driver packs for the following devices:
  - Cisco IP camera high definition models
  - Cisco IP camera standard definition models
  - Optelecom C-44 4-port encoder
  - Pelco Spectra IV IP PTZ dome camera
  - ICX serial driver
  - AXIS Q7406 6-port encoder blade
  - Sony SNC-DF85 network mini-dome camera

The VSMS Advantage

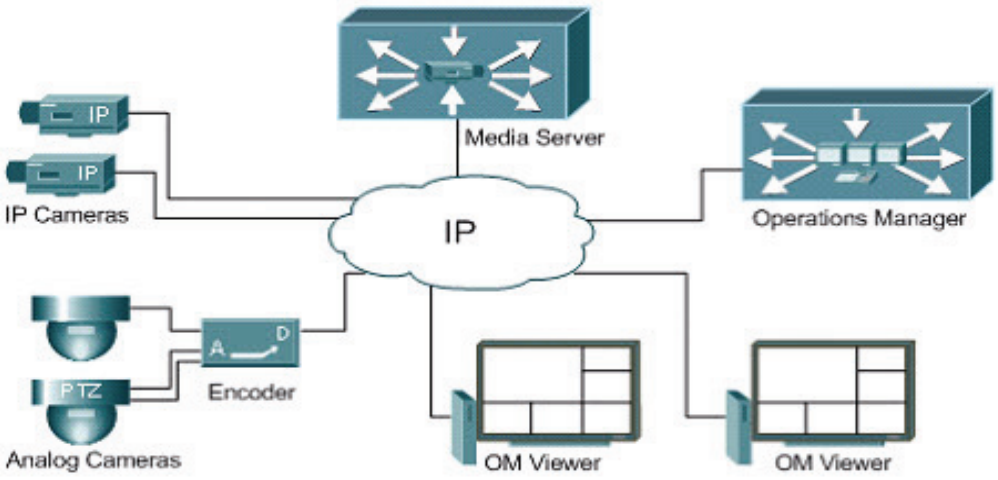
VSMS can create large scale, low latency media applications requiring management and archiving of a large number of live streams for multiple, simultaneous users. Specific features of VSMS are outlined as follows:

- HTTP-based APIs—Easy to integrate using standard application development tools.
- Regular and loop archives—Flexible archiving of streaming media at multiple locations and in various framerates and duration.
- Event trigger support—Integrates external events such as alarms, process controls, and other system events.
- Cascading architecture—Available scaling to accommodate any number of cameras, users, and archives.
- Dynamic file allocation—Optimizes disk usage for stored media and assures availability of full archive data.
- Bandwidth management capability—Permits multiple site locations to manage and/or restrict bandwidth consumption at the LAN, intranet and Internet level.
- ActiveX Controls that display video and control cameras—Permits display of video and control of cameras.
- Open interface standards—Expands to incorporate new codecs, camera and device controls, biometrics, and other systems.

VSMS System Architecture

VSMS is built in a modular architecture to permit inclusion of new technologies throughout and to provide for implementation of systems for virtually any number of media streams, users, and archives. Figure 1-1 depicts the four layers of the VSMS architecture:

Figure 1-1 VSMS Architecture



The first layer is video and audio acquisition. VSMS relies on third-party sources to provide encoded video and audio streams to the system. Supported standards include Motion JPEG (JPEG), MPEG-2, and MPEG-4.

The second layer represents the VSMS core functionality. Operating on the Linux operating system, the VSMS core is divided into two main components: Proxy Server and Archive Server. Both components can run on a single host or can be distributed across multiple hosts as necessary to meet redundancy and volume requirements. Proxy Servers and Archive Servers can also be distributed as applicable across a physical network so that media streams and storage are optimized and network bandwidth is managed efficiently. Applications in the third layer interact with the VSMS core via HTTP-based APIs.

The third layer is the application layer. VSMS includes the Apache web server so that applications with static web pages can be developed without additional software. For complex applications, systems integrators can typically employ application development environments such as WebLogic or WebSphere, along with integration of relational database management software such as Oracle or DB2.

The fourth layer provides a variety of thin client applets, controls, and objects to present live and archived media inside the finished application. Applets are available for single and multiple media streams and for archived media playback with a flexible API-based control interface.

CHAPTER 2

Proxy Commands

A proxy runs on a specific device acting as a source for an encoder or IP camera. This enables a single encoder or IP camera source to be viewed and recorded by multiple other sources. At least one proxy is required per a video source and supports other sources such as clients, child proxies, and archivers.

Child proxies run on local or other proxies and have the same resolution, quality, and media type of its host, and can have a lower framerate for motion JPEG. VSMS supports multiple types of media proxies.

Start Video Proxy

The start proxy command starts a proxy for a JPEG, MPEG, or video source. The source can be the original encoding device or another VSMS proxy. Parent-child proxies can be nested indefinitely as resources permit.

See the encoder documentation for device specific limitations on bitrates, framerates, resolution, and quality settings.

Command

```
http://<host>/command.bwt?command=start&type=proxy&name=p_Test<port#>&source=<port#>@10.10
.55.153&srctype= Supported Devices>
```

Table 2-1 Required Fields

Command	Description
`host`	Format: [hostname.domain | IP address] Web address of host where VSMS is running. **Note** VSMS runs on port 80 by default. Specify additional ports similar to vsms_host.cisco.com:8080.
`command`	Reserved value: [start]
`type`	Reserved value: [proxy]
`name`	Character class: [0-9 | A-Z | a-z | _ | -]; Reserved: (-1); Length: (1-64) Name for this proxy. This is the value that would be used as a source for a child proxy. **Note** Each proxy must have a unique name on a given VSMS host.

Table 2-1 Required Fields (continued)

Command	Description
`source`	Format: [#@address:port# (required): The source input number. This is the camera input number on the source. address (optional): IP address/hostname.domain of the source, if no address given, the address defaults to localhost. port (optional): Port number, if no port is given the port defaults to 80. or - Format: [#_#@address:port] #_# (required): The device input number on the device and the configuration number for the input. address (optional): IP address/hostname.domain of the device, if no address given, the address defaults to localhost. port (optional): Port number, if no port is given the port defaults to 80.
`mediatype`	Reserved values: Codec is used to encode the media.
`srctype`	Reserved values: Supported devices such as VSMS proxies, video servers, and network cameras for this proxy. • For some devices, only one proxy connection is permitted for each JPEG, MPEG-2, and MPEG-4. The JPEG maximum framerate is 2 frames per second.

Table 2-2 Optional Fields

Command	Description
`quality`	Range: [1-100](default=50) Quality of the JPEG feed, where 100 is the highest possible quality. If the quality parameter is set to less than 50, the framerate has priority and the requested framerate is as per the proxy_<device>_mpeg4.xml file. If the quality parameter is set greater than 50, the generated image quality has priority (while maintaining the bitrate) and lower framerates are returned. Any number between 1 and 49 indicates the same priority for the framerate. Any number between 50 and 99 indicates the same priority for the image quality. The higher the bitrate, the higher the image quality. **Note** A child proxy must have the same quality value as the parent proxy.
`framerate`	Range:[30, 15, 7.5, 3] Maximum number of JPEG frames per second transmitted. **Note** A child proxy cannot have a higher framerate value than the parent proxy.
`width`	Range: [160-720](320) Width of the video feed in pixels. **Note** A child proxy must have the same width value as the parent proxy.
`height`	Range: `[112-576](240)` Height of the video feed in pixels. **Note** A child proxy must have the same height value as the parent proxy.
`bitrate`	Range: [28.8-5000](640) Kilobytes per second transmitted for MPEG feed. If the quality parameter is set to less than 50, the framerate has priority and the requested framerate is as per the proxy_<device>_mpeg4.xml file. If the quality parameter is set greater than 50, the generated image quality has priority (while maintaining the bitrate) and lower framerates are returned. Any number between 1 and 49 indicates the same priority for the framerate. Any number between 50 and 99 indicates the same priority for the image quality. The higher the bitrate, the higher the image quality.

Table 2-2 Optional Fields (continued)

Command	Description
username	Format: [user name] Administration user name for encoding device. **Note** If a username and password is enabled for a device or to do Camera Controls and Events Setup, this parameter is required.
password	Format: [password] Administration password for encoding device. **Note** If a username and password is enabled for a device or to do Camera Controls and Events Setup, this parameter is required.
resolution	Reserved values: [qcif | cif | 2cif | 4cif | d1|1M| 2M|3M|4M|5M](cif) Resolution used to start proxy using look-up values for width and height of video. De-Interlacing is supported for 4cif | d1.
format	Reserved values: [ntsc | pal](ntsc) Video standard name for format used to start proxy using look-up values for width and height of video. **Note** The width and height parameters take precedence over format.
udp	Reserved values: [0 | 1](0) Protocol for MPEG-4 multicast streams. Use 1 to turn on. **Note** To start a multicast stream an address must be provided.
multicast	Address: [IP address | hostname] To start or join a multicast stream the address of the device must be included. When upgrading IMC, replace the previous CLASSID with the new CLASSID

Return Values

A standard HTTP/1.x header followed by:

```
Content-Type: text/plain
Return Code: <[proxy name] or -1 or output>
    [proxy name] Successful completion of the URL command
    -1 Error in execution of the URL command
        Error String: <error-message>
```

Example

The following command starts a video proxy using just the required parameters. VSMS is running on <host> and the video source is running on video input 1 of a <device> with IP address 192.169.168.2.

```
http://<host>/command.bwt?command=start&type=proxy&name=entrance&srctype=c&mediatype=jpeg&
source=1@192.169.168.2
```

Example

The following command establishes parent-child proxy set up on the same VSMS host.

```
http://<host>/command.bwt?command=start&type=proxy&name=proxy2proxy&srctype=proxy&mediatype=jpeg&source=officecam&framerate=5&width=320&height=240&quality=50
```

Example

The following command starts a <device> video server proxy of media type mpeg2-v (no audio) at a bit rate of 1024.

```
http://<host>/command.bwt?command=start&type=proxy&name=receptionarea&srctype=<device>&source=1@mpeg.cisco.com&bitrate=1024&mediatype=mpeg2-v&username=admin&password=qwertyJoe
```

Example

The following command starts an audio proxy. Optional parameter fields are not required for an audio proxy.

```
http://<host>/command.bwt?command=start&type=proxy&name=audiocast&srctype=audio&source=0@audio1
```

Start Audio Proxy

The audio proxy command starts a proxy for an audio source. The source can be the original encoding device or another VSMS proxy. Parent-child proxies can be nested indefinitely as resources permit.

See the encoder documentation for device specific limitations on bitrates, framerates, resolution, and quality settings.

Command

```
http://<host>/command.bwt?command=start&type=proxy&name=p_Test_<port#>&source=<port#>@10.10.55.153&srctype=<Supported Devices>
```

Table 2-3 Required Fields

Command	Description
type	Reserved value: [proxy]
mediatype	Reserved values: Codec used to encode the media.
name	Character class: [0-9 | A-Z | a-z | _ | -]; Reserved: (-1); Length: (1-64) Name for this proxy. This is the value that would be used as a source for a child proxy. **Note** Each proxy must have a unique name on a given VSMS host.

Table 2-3 Required Fields (continued)

Command	Description
source	Format: [#@address:port]# (required): The source input number. This is the camera input number on the source. address (optional): IP address/hostname.domain of the source, if no address given, the address defaults to localhost. port (optional): Port number, if no port is given the port defaults to 80. or - Format: [#_#@address:port] #_# (required): The device input number on the device and the configuration number for the input. address (optional): IP address/hostname.domain of the device, if no address given, the address defaults to localhost. port (optional): Port number, if no port is given the port defaults to 80.
srctype	Reserved values: Supported devices such as VSMS proxies, video servers, and network cameras for this proxy. – For some devices, only one proxy connection is permitted for each JPEG, MPEG-2, and MPEG-4. The JPEG maximum framerate is 2 frames per second.
upd	Reserved values: [0 | 1](0) Protocol for MPEG-4 multicast streams. Use 1 to turn on. **Note** An address must be provided to start a multicast stream.
multicast	Address: [IP address | hostname] To start or join a multicast stream the address of the device must be included here.
username	Format: [user name] Administration user name for encoding device. **Note** If a username and password is enabled for a device or to do Camera Controls and Events Setup, this parameter is required.
password	Format: [password] Administration password for encoding device. **Note** If a username and password is enabled for a device or to do Camera Controls and Events Setup, this parameter is required.

Return Values

A standard HTTP/1.x header followed by:

```
Content-Type: text/plain
Return Code: <[proxy name] or -1 or output>
    [proxy name] Successful completion of the URL command
    -1 Error in execution of the URL command
        Error String: <error-message>
```

Update Video Proxy

The following command updates an existing proxy with different parameter values. For example, updating a proxy to a different source will seamlessly change the feed being viewed by any clients.

Note If an archive is running against the proxy, updating the proxy source causes the archive to be unplayable. In this case, stop the archive first, update the proxy, and then start a new archive.

Command

```
http://<host>/command.bwt?command=update&name=<proxy_name>&source=<name@address:port>&srct
ype=<SupportedDevices>&quality=<1-100>&width=<_>&height=<_>&framerate=<0.001-30>&bitrate=<
28.8-5000>&username=<user name>&password=<password>&resolution=<[qcif | cif |
4cif]>&format=<[ntsc | pal]>
```

Table 2-4 Required Fields

Command	Description
host	Format: [hostname.domain | IP address] Web address of host where VSMS is running. **Note** VSMS runs on similar to vsms_host.cisco.com:8080.
command	Reserved value: [update]
type	Reserved value: [proxy]
name	Character class: [0-9 | A-Z | a-z | _ | -]; Reserved: (-1); Length: (1-64) Proxy name for the proxy **Note** Name changing is not supported.

Table 2-4 Required Fields (continued)

Command	Description
source	Format: [#@address:port] # (required): The source input number. This is the camera input number on the source. address (optional): IP address/hostname.domain of the source, if no address given, the address defaults to localhost. port (optional): Port number, if no port is given the port defaults to 80. or- Format: [#_#@address:port] #_# (required): The device input number on the device and the configuration number for the input. address (optional): IP address/hostname.domain of the device, if no address given, the address defaults to localhost. port (optional): Port number, if no port is given the port defaults to 80.
srctype	Reserved values: Supported devices such as VSMS proxies, video servers, and network cameras for this proxy.

Table 2-5 Optional Fields

Command	Description
quality	Range: [1-100](default=50) Quality of the JPEG feed, where 100 is the highest possible quality. **Note** A child proxy must have the same quality value as the parent proxy.
framerate	Range:[30, 15, 7.5, 3] Maximum number of JPEG frames per second transmitted. **Note** A child proxy cannot have a higher framerate value than the parent proxy.
width	Range: [160-720](320) Width of the video feed in pixels. **Note** A child proxy must have the same width value as the parent proxy.
height	Range: [112-576](240) Height of the video feed in pixels. **Note** A child proxy must have the same height value as the parent proxy

Table 2-5 Optional Fields (continued)

Command	Description
`username`	Format: [user name] Administration user name for encoding device. **Note** If a username and password is enabled for a device or to do Camera Controls and Events Setup, this parameter is required.
`password`	Format: [password] Administration password for encoding device. **Note** If a username and password is enabled for a device or to do Camera Controls and Events Setup, this parameter is required.
`bitrate`	Range: [28.8-5000](640) Kilobytes per second transmitted for MPEG feed. **Note** A child proxy must have the same bitrate value as the parent proxy.
`resolution`	Reserved values: [qcif | cif | 2cif | 4cif | d1|1M| 2M|3M|4M|5M](cif) Resolution used to start proxy using look-up values for width and height of video. De-Interlacing is supported for 4cif | d1. **Note** A child proxy must have the same resolution value as the parent proxy. – The width and height parameters take precedence over resolution.
`format`	Reserved values: [ntsc | pal](ntsc) Video standard name for format used to start proxy using look-up values for width and height of video. **Note** A child proxy must have the same format value as the parent proxy. – The width and height parameters take precedence over format.
`udp`	Reserved values: [0 | 1](0) Protocol for MPEG-4 multicast streams. Use 1 to turn on. **Note** An address must be provided to start a multicast stream.
`multicast`	Address: [IP address | hostname] To start or join a multicast stream the address of the device must be included here.

Return Values

A standard HTTP/1.x header followed by:

```
Content-Type: text/plain
Return Code: <[proxy name] or -1 or output>
    [proxy name] Successful completion of the URL command
    -1 Error in execution of the URL command
```

```
Error String: <error-message>
```

Example

The following command updates the proxy OFFICECAM on <device>_host remote server 0 at a framerate of 10.

```
http://<host>/command.bwt?command=update&source=0@<device>_host&type=proxy&srctype=sproxy&
name=OFFICECAM&framerate=10
```

Update Audio Proxy

The following command updates an existing proxy with different parameter values. For example, updating a proxy to a different source will seamlessly change the feed being viewed by any clients.

Note If an archive is running against the proxy, updating the proxy source causes the archive to be unplayable. In this case, stop the archive first, update the proxy, and then start a new archive.

Command

```
http://<host>/command.bwt?command=update&name=<proxy_name>&source=<name@address:port>&srct
ype=<Supported Devices>
```

Table 2-6 Required Fields

Command	Description
name	Character class: [0-9 | A-Z | a-z | _ | -]; Reserved: (-1); Length: (1-64) Proxy name for the proxy **Note** Name changing is not supported.
source	Format: [#@address:port] # (required): The source input number. This is the camera input number on the source. address (optional): IP address/hostname.domain of the source, if no address given, the address defaults to localhost. port (optional): Port number, if no port is given the port defaults to 80. or - Format: [#_#@address:port] #_# (required): The device input number on the device and the configuration number for the input. address (optional): IP address/hostname.domain of the device, if no address given, the address defaults to localhost. port (optional): Port number, if no port is given the port defaults to 80.

Table 2-6 Required Fields (continued)

Command	Description
srctype	Reserved values Supported devices such as VSMS proxies, video servers, and network cameras for this proxy.
username/password	Enter the user name and password for the device.

Return Values

A standard HTTP/1.x header followed by:

```
Content-Type: text/plain
Return Code: <[proxy name] or -1 or output>
    [proxy name] Successful completion of the URL command
    -1 Error in execution of the URL command
        Error String: <error-message>
```

Stop Proxy

The stop proxy command stops a proxy and any archives running against it on the VSMS host.

Command

```
http://<host>/command.bwt?command=stop&type=proxy&name=<proxy_name>
```

Table 2-7 Required Fields

Command	Description
host	Format: [hostname.domain | IP address] Web address of host where VSMS is running. **Note** VSMS runs on port 80 by default. Specify additional ports similar to vsms_host.cisco.com:8080.
command	Reserved value: [stop]
type	Reserved value: [proxy]
name	Character class: [0-9 | A-Z | a-z | _ | -]; Reserved: (-1); Length: (1-64) Name for the proxy being stopped. This is the value that would be used as a source for a child proxy. **Note** Each proxy must have a unique name on a given VSMS host.

Return Values

A standard HTTP/1.x header followed by:

```
Content-Type: text/plain
Return Code: <[proxy name] or -1 or output>
    [proxy name] Successful completion of the URL command
    -1 Error in execution of the URL command
```

```
Error String: <error-message>
```

Example

This command stops a proxy with name OFFICECAM and any archives running against that proxy on this VSMS host.

```
http://<host>/command.bwt?command=stop&type=proxy&name=OFFICECAM
```

View JPEG Frames

A quick and easy way to view a JPEG proxy is to use thin-client URL shown below. In addition this thin-client URL can be used with an HTML <IMG> tag. Examples with and without an HTML <IMG> tag are provided in this section.

Command

```
http://<host>/video.jpg?source=<proxy_name>&framerate=<n>&timeout=<n>
```

Table 2-8 Required Fields

Command	Description
host	Format: [hostname.domain | IP address] Web address of host where VSMS is running. **Note** VSMS runs on port 80 by default. Specify additional ports similar to vsms_host.cisco.com:8080.
device	Format: [proxy name] proxy name (required): The device name. This is a proxy name for the media source. **Note** Remote address:port is not supported. To view a JPEG frame from another VSMS host specify it in the <host> portion of the URL.

Table 2-9 Optional Fields

Command	Description
framerate	Range: [30, 15, 7.5, 3] Maximum number of frames per second transmitted to view proxy. For a single frame snapshot use framerate=Ø. If proxy framerate is running lower than 5 frames per second, then the proxy framerate is the default framerate.
timeout	Format: [integer in hours](600) Time in hours of when to disconnect the client. For no time out, use timeout=Ø. The default is 5.

Example

The following command requests a single frame snapshot from device, camera1.

```
http://<host>/video.jpg?source=camera1&framerate=0
```

Example

Using the HTML <IMG> tag included in a web page, the following displays the proxy camera1 at a framerate of 1 frame per second and a time out of 900 seconds.

```
<img src="http://<host>/video.jpg?source=camera1&framerate=1&timeout=900" width="160"
height="120" alt="View Proxy with HTML Image Tag" />
```

Note Microsoft Internet Explorer does not support framerates >Ø).

List All Proxies

The list all proxies command displays all the proxies running on a VSMS host.

Command

```
http://<host>/info.bwt?type=proxy&name=<proxy_name>&display=<[html | text | ssv]>
```

Table 2-10 Required Fields

Command	Description
host	Format: [hostname.domain | IP address] Web address of host where VSMS is running. **Note** VSMS runs on port 80 by default. Specify additional ports similar to vsms_host.cisco.com:8080.
type	Reserved value: [proxy]

Table 2-11 Optional Fields

Command	Description
name	Character class: [0-9 | A-Z | a-z | _ | -]; Reserved: (-1); Length: (1-64) Proxy name to display information. **Note** All proxies are listed if no name is given.
display	Reserved values: [html | text | ssv](default=html)

Return Values

A standard HTTP/1.x header followed by:

```
Content-Type: text/plain
Return Code: <[There are no active proxies at this time] or [-1 if there are no proxies on
the server or proxies in the user designated format (text, html, ssv)] or output>
    [proxy info] Successful completion of the URL command
```

```
        A table containing the following columns: name, status, type, exec, source,
        mediatype, framerate, bitrate, quality, width, height, model
```

Return values are as follows:

```
status = running (normal)/stopped (if the proxy crashed)
type = the device type that is the stream source (the device list provides a comprehensive
list of all the supported devices)
exec = proxy (the name of the executable)
source = ip address of the device along with the video input number
mediatype = media mpeg4-v, jpeg or mpeg2
f/b-rate = means framerate (jpeg only) or bitrate (for mpeg2 and mpeg4)
quality = the quality of the stream 1-100 scale
width, height = geometry of the stream
model = numeric value that uniquely identify a device type
    -1 Error in execution of the URL command
    Error String: <error-message>
```

Example

The following command gets all running proxy information on <host> in text format.

```
http://<host>/info.bwt?type=proxy&display=text
```

Get Proxy Source

The get proxy source command gets the source value for a proxy. The return value is in text only.

Command

```
http://<host>/info.bwt?type=proxy&name=<proxy_name>&property=source
```

Table 2-12 Required Fields

Command	Description
host	Format: [hostname.domain | IP address] Web address of host where VSMS is running. **Note** VSMS runs on port 80 by default. Specify additional ports similar to vsms_host.cisco.com:8080.
type	Reserved value: [proxy]
name	Character class: [0-9 | A-Z | a-z | _ | -]; Reserved: (-1); Length: (1-64) Proxy being queried for property information.
property	Reserved value: [device] Returns device value of proxy.

Return Values

A standard HTTP/1.x header followed by:

```
Content-Type: text/plain
Return Code: <[name@address:port] or -1 or output>
    [name@address:port] Successful completion of the URL command
```

```
    -1 Error in execution of the URL command
        Error String: <error-message>
```

Example

The following command gets device information for proxy 1036:

```
http://<host>/info.bwt?type=proxy&name=1036&property=source Returns: 1@10.10.45.234
```

Get Proxy Media Type

The get proxy media type command gets the media value for a proxy. The return value is in text only.

Command

```
http://<host>/info.bwt?type=proxy&name=<proxy_name>&property=mediatype
```

Table 2-13 Required Fields

Command	Description
host	Format: [hostname.domain | IP address] Web address of host where VSMS is running. **Note** VSMS runs on port 80 by default. Specify additional ports similar to vsms_host.cisco.com:8080.
type	Reserved value: [proxy]
name	Character class: [0-9 | A-Z | a-z | _ | -]; Reserved: (-1); Length: (1-64) Proxy being queried for property information.
property	Reserved value: Returns JPEG, MPEG, or audio values.

Return Values

A standard HTTP/1.x header followed by:

```
Content-Type: text/plain
Return Code: <[jpeg | mpeg2-v | mpeg4-v | audio] or -1 or output>
    [jpeg | mpeg2-v | mpeg4-v | audio] Successful completion of the URL command
    -1 Error in execution of the URL command
        Error String: <error-message>
```

Example

The following command gets media type information for proxy 1036:

```
http://<host>/info.bwt?type=proxy&name=1036&property=mediatypeReturns: jpeg
```

Get Proxy Framerate or Bitrate

The get proxy framerate or bitrate command gets the framerate for a JPEG proxy and the bitrate for a MPEG or audio proxy. The return value is in text only.

Command

```
http://<host>/info.bwt?type=proxy&name=<proxy_name>&property=rate
```

Table 2-14 Required Fields

Command	Description
host	Format: [hostname.domain | IP address] Web address of host where VSMS is running. **Note** VSMS runs on port 80 by default. Specify additional ports similar to vsms_host.cisco.com:8080.
type	Reserved value: [proxy]
name	Character class: [0-9 | A-Z | a-z | _ | -]; Reserved: (-1); Length: (1-64) Proxy being queried for property information.
property	Reserved value: [rate] Returns current framerate for JPEG or bit rate for MPEG and audio proxy.

Return Values

A standard HTTP/1.x header followed by:

```
Content-Type: text/plain
Return Code: <[framerate | bitrate] or -1 or output>
    [framerate | bitrate] Successful completion of the URL command
    -1 Error in execution of the URL command
        Error String: <error-message>
```

Example

The following command gets framerate information for JPEG proxy 1036:

```
http://<host>/info.bwt?type=proxy&name=1036&property=rateReturns: 15.000000
```

Example

The following command gets bitrate information for MPEG-2 proxy Example_Proxy.

```
http://<host>/info.bwt?type=proxy&name=Example_Proxy&property=rateReturns: 5000
```

Get Proxy JPEG Quality

The get proxy quality command gets the quality value for a JPEG proxy. The return value is in text only.

Command

```
http://<host>/info.bwt?type=proxy&name=<proxy_name>&property=quality
```

Table 2-15 Required Fields

Command	Description
host	Format: [hostname.domain | IP address] Web address of host where VSMS is running. **Note** VSMS runs on port 80 by default. Specify additional ports similar to vsms_host.cisco.com:8080.
type	Reserved value: [proxy]
name	Character class: [0-9 | A-Z | a-z | _ | -]; Reserved: (-1); Length: (1-64) Proxy being queried for property information.
property	Reserved value: [quality] Returns quality value for JPEG video being encoded.

Return Values

A standard HTTP/1.x header followed by:

```
Content-Type: text/plain
Return Code: <[1-100] or -1 or output>
    [1-100] Successful completion of the URL command
    -1 Error in execution of the URL command
        Error String: <error-message>
```

Example

The following command gets quality information for proxy 1036:

```
http://<host>/info.bwt?type=proxy&name=1036&property=qualityReturns: 50
```

Get Proxy Video Width

The get proxy width command gets the width in pixels for a video proxy. The return value is in text only.

Command

```
http://<host>/info.bwt?type=proxy&name=<proxy_name>&property=width
```

Table 2-16 Required Fields

Command	Description
host	Format: [hostname.domain | IP address] Web address of host where VSMS is running. **Note** VSMS runs on port 80 by default. Specify additional ports similar to vsms_host.cisco.com:8080.
type	Reserved value: [proxy]
name	Character class: [0-9 | A-Z | a-z | _ | -]; Reserved: (-1); Length: (1-64) Proxy being queried for property information.
property	Reserved value: [width] Returns width in pixels of JPEG video feed.

Return Values

A standard HTTP/1.x header followed by:

```
Content-Type: text/plain
Return Code: <[160-720] or -1 or output>
    [160-720] Successful completion of the URL command
    -1 Error in execution of the URL command
        Error String: <error-message>
```

Example

The following command gets width information for proxy 1036:

```
http://<host>/info.bwt?type=proxy&name=1036&property=widthReturns: 352
```

Get Proxy Video Height

The get proxy width command gets the height in pixels for a video proxy. The return value is in text only.

Command

```
http://<host>/info.bwt?type=proxy&name=<proxy_name>&property=height
```

Table 2-17 Required Fields

Command	Description
host	Format: [hostname.domain | IP address] Web address of host where VSMS is running. **Note** VSMS runs on port 80 by default. Specify additional ports similar to vsms_host.cisco.com:8080.
type	Reserved value: [proxy]

Table 2-17 Required Fields (continued)

Command	Description
name	Character class: [0-9 | A-Z | a-z | _ | -]; Reserved: (-1); Length: (1-64) Proxy being queried for property information.
property	Reserved value: [height] Returns height in pixels of JPEG video feed.

Return Values

A standard HTTP/1.x header followed by:

```
Content-Type: text/plain
Return Code: <[112-576] or -1 or output>
    [112-576] Successful completion of the URL command
    -1 Error in execution of the URL command
        Error String: <error-message>
```

Example

```
http://<host>/info.bwt?type=proxy&name=1036&property=heightReturns: 240
```

Get Proxy Model Type

The get proxy model type command gets the device model for a proxy. The return value is in text only.

Command

```
http://<host>/info.bwt?type=proxy&name=<proxy_name>&property=model
```

Table 2-18 Required Fields

Command	Description
host	Format: [hostname.domain | IP address] Web address of host where VSMS is running. **Note** VSMS runs on port 80 by default. Specify additional ports such as vsms_host.cisco.com:8080.
type	Reserved value: [proxy]
name	Character class: [0-9 | A-Z | a-z | _ | -]; Reserved: (-1); Length: (1-64) Proxy being queried for property information.
property	Reserved values: [model] Supported devices for a given proxy. Possible return values are mapped integers.

Return Values

A standard HTTP/1.x header followed by:

```
Content-Type: text/plain
Return Code: <[1-20] or -1 or output>
    [1-20] Successful completion of the URL command
    -1 Error in execution of the URL command
        Error String: <error-message>
```

Example

The following command gets model information for proxy 1036:

```
http://<host>/info.bwt?type=proxy&name=1036&property=modelReturns: 7
```

Get Proxy Status

The get proxy status command gets the status for a proxy (running or stopped). The return value is in text only.

Command

```
http://<host>/info.bwt?type=proxy&name=<proxy_name>&property=status
```

Table 2-19 Required Fields

Command	Description
host	Format: [hostname.domain | IP address] Web address of host where VSMS is running. **Note** VSMS runs on port 80 by default. Specify additional ports similar to vsms_host.cisco.com:8080.
type	Reserved value: [proxy]
name	Character class: [0-9 | A-Z | a-z | _ | -]; Reserved: (-1); Length: (1-64) Proxy being queried for property information.
property	Reserved values: [status] Returns current status of proxy. Possible return values are: Running: Proxy is running normally. Stopped: Failed due to an error. A proxy stopped by a stop proxy command would not be listed here. Suspended: Proxy is in a postponed state.

Return Values

A standard HTTP/1.x header followed by:

```
Content-Type: text/plain
Return Code: <[Running | Stopped] or -1 or output>
    [Running | Stopped|Suspended] Successful completion of the URL command
    -1 Error in execution of the URL command
        Error String: <error-message>
```

Example

The following command gets current status information for proxy 1036:

```
http://<host>/info.bwt?type=proxy&name=1036&property=statusReturns: Running
```

CHAPTER 3

Archive Commands

The VSMS core functionality is distributed between two major components; proxy and archiver/recorder. The proxy acts as a multiplexer by retrieving media data from one encoder over TCP/UDP or MULTICAST UDP (specified at its configuration). It is distributed to a large number of clients (archiver(s) and viewing applications) by writing the data alternatively in two segments of shared memory.

The archiver records media data from a proxy and is able to play back the recorded data within two to three seconds of the current time.

Archive Types

An archive can be RUNNING, SHELVED or PAUSED.

RUNNING

During RUNNING, an “archiver” process exists that is actively recording and storing data to disk. RUNNING means a process thread exists that is running inside the system bootup process called xvcrman. This process reads frames from a proxy and writes them to disk.

SHELVED

During SHELVED, the “archiver” process has terminated and no data is being stored the disk. SHELVED means there is no process thread, no proxy, and no data being written to disk.

PAUSED

During PAUSED, there is a process thread that is running within the xvcrman. This thread is not reading data from the proxy and no data is being written to disk. It remains in this state until it is restarted and enters the RUNNING state or stops and enters the SHELVED state.

A PAUSED archiver is useful for scheduled archive implementation where a client starts a regular archiver for a duration of time and tells the archiver to pause when it completes the duration. When the next schedule period arrives the client restarts it. It is also useful for improving the performance of the archiver startup. To start recording, the archiver start its proxy process and begins recording.

Archive Commands

Start

Command

```
http://<host>/command.bwt?command=start&type=archive&name=<unique archive
id>&source=<proxy name>&duration=<duration in seconds>&framerate=<0.001-30>&loop=<0 |
1>&desc=<description text>&repos=<volume>&daystolive=<# of days until
expires>&killproxy=<0 | 1>
```

Table 3-1 Required Fields

Command	Description
host	Format: [hostname.domain | IP address] Web address of host where VSMS is running. **Note** VSMS runs on port 80 by default. Specify additional ports here similar to vsms_host.cisco.com:8080.
command	Reserved value: [start]
type	Reserved value: [archive]
name	Character class: [0-9 | A-Z | a-z | _ | -] Reserved: (-1); Length: (1-127) Descriptive Name of archive being recorded. **Note** Each archive must have a unique name on the VSMS host.
Source	Format: [name] name (required): This is the name of the source proxy being archived on the VSMS host.

Table 3-2 Optional Fields

Command	Description
duration	Format: [integer in seconds](default=3600) Total archive time in seconds. **Note** For loop archives, minimum duration supported is 3600 seconds.
desc	Character class: [0-9 | A-Z | a-z | _ | <space> | -](proxy source); Reserved: (-1); Length: (0-20) Brief description of the archive. **Note** If desc is left blank, VSMS will store the name of the proxy source in the description field.

Table 3-2 Optional Fields (continued)

Command	Description
framerate	Range: [0.001-30](proxy framerate) Maximum number of JPEG frames per second requested from source proxy. **Note** The framerate cannot be higher than source proxy.
loop	Boolean values: [0 | 1](0) Record a loop [1] or regular archive [0]. A loop archive continuously records over its beginning once it reaches the end of its duration. A regular archive stops once it reaches the end of its duration.
repos	Format: [repository_mount] Location where the new archive will be saved.
daystolive	Format: [integer in days](0) Number of days from the date archive stops the archive will be stored before system removal. For permanent storage set daystolive=Ø.
Killproxy	Boolean values: [0 | 1] (0) If set to 1, archive will be placed into PAUSED state when it completes. If duration is set to 0 and killproxy is set to 0, archive will be immediately placed into PAUSED state.

Return Values

A standard HTTP/1.x header followed by:

```
Content-Type: text/plain
Return Code: <[archive name] or -1 output>
[archive name] Successful completion of the URL command
-1 Error in execution of the URL command
```

Example

The following command starts a 60 minute archive.

```
http://<host>/command.bwt?command=start&type=archive&source=officecam&duration=3600&name=o
fficecam_for_60min&pid=9934
```

Example

The following command starts a continuous 24 hour loop archive

```
http://<host>/command.bwt?command=start&type=archive&name=baybridgearchi&pid=42845&source=
baybridge&duration=86400&loop=1
```

Example

The following command starts a 2 hour audio archive with the command to pause it when complete.

http://<host>/command.bwt?command=start&type=archive&name=audio_arch&pid=1265&source=audiocast
&duration=7200&killproxy=1

Update

Command

```
http://<host>/command.bwt?command=update&type=archive&name=<archive
name>&framerate=<0.001-30>

http://<host>/command.bwt?command=update&type=archive&name=<archive
name>&duration=<duration in seconds>&framerate=<0.001-30>&loop=<0 | 1>&desc=<description
text>&daystolive=<# of days until expires>&killproxy=<0 | 1>
```

Table 3-3 Required Fields

Command	Description
host	Format: [hostname.domain | IP address] Web address of host where VSMS is running. **Note** VSMS runs on port 80 by default. Specify additional ports here similar to vsms_host.cisco.com:8080.
command	Reserved value: [start]
type	Reserved value: [archive]
name	Character class: [0-9 | A-Z | a-z | _ | -] Reserved: (-1); Length: (1-127) Descriptive Name of archive being recorded. **Note** Each archive must have a unique name on the VSMS host.

Table 3-4 Optional Fields

Command	Description
duration	Format: [integer in seconds](default=3600) Total archive time in seconds. **Note** For loop archives, minimum duration supported would be 3600 seconds.
desc	Character class: [0-9 | A-Z | a-z | _ | <space> | -](proxy source); Reserved: (-1); Length: (0-20) Brief description of the archive. **Note** If left blank, VSMS will store the name of the proxy source in the description field.
framerate	Range: [0.001-30](proxy framerate) Maximum number of JPEG frames per second requested from source proxy. **Note** The framerate cannot be higher than source proxy.

Table 3-4 Optional Fields (continued)

Command	Description
loop	Boolean values: [0 | 1](0) Record a loop [1] or regular archive [0]. A loop archive continuously records over its beginning once it reaches the end of its duration. A regular archive stops once it reaches the end of its duration.
daystolive	Format: [integer in days](0) Number of days from the date archive stops the archive will be stored before system removal. For permanent storage set daystolive=Ø.
Killproxy	Boolean values: [0 | 1] (0) If set to 1, archive will be placed into PAUSED state when it completes. If duration is set to 0 and killproxy is set to 0, archive will be immediately placed into PAUSED state.

Return Values

A standard HTTP/1.x header followed by:

```
Content-Type: text/plain
Return Code: <[archive name] or -1 or output>
[archive name] Successful completion of the URL command
-1 Error in execution of the URL command
Error String: <error-message>
```

Example

The following command updates an archive with server name archive30 to 15 frames per second, sets its duration to 1 hour, and pauses the archive upon completion.

```
http://<host>/command.bwt?command=update&type=archive&name=archive30&framerate=15&duration
=3600&loop=1&killproxy=1
```

Stop

```
http://<host>/command.bwt?command=stop&type=archive&name=<archive name>
```

Table 3-5 Required Fields

Command	Description
host	Format: [hostname.domain | IP address] Web address of host where VSMS is running. **Note** VSMS runs on port 80 by default. Specify additional ports here similar to vsms_host.cisco.com:8080.
command	Reserved value: [stop]

Table 3-5 Required Fields (continued)

Command	Description
type	Reserved value: [archive]
name	Character class: [0-9 | A-Z | a-z | _ | -]; Reserved: (-1); Length: (1-127) Name of archive being stopped. **Note** Each archive must have a unique name on the VSMS host.

Table 3-6 Optional Fields

Command	Description
Killproxy	Boolean values: [0 | 1] (0) If set to 1, archive will be placed into PAUSED state.

Return Values

A standard HTTP/1.x header followed by:

```
Content-Type: text/plain
Return Code: <[archive name] or -1 or output>
[archive name] Successful completion of the URL command
-1 Error in execution of the URL command
Error String: <error-message>
```

Example

The following command stops an archive with server name archive3001.

```
http://<host>/command.bwt?command=stop&type=archive&name=archive3001
```

Example

The following command pauses an archive with server name archive_sched1.

```
http://<host>/command.bwt?command=stop&type=archive&name=archive_sched1&killproxy=1
```

Remove

```
http://<host>/cgi-bin/smanager.bwt?command=remove&name=<archive name>
```

Table 3-7 Required Fields

Command	Description
host	Format: [hostname.domain | IP address] Web address of host where VSMS is running. **Note** VSMS runs on port 80 by default. Specify additional ports here similar to vsms_host.cisco.com:8080.
command	Reserved value: [remove]
type	Reserved value: [archive]
name	Character class: [0-9 | A-Z | a-z | _ | -]; Reserved: (-1); Length: (1-127) Name of archive being stopped. **Note** Each archive must have a unique name on the VSMS host.

Return Values

A standard HTTP/1.x header followed by:

```
Content-Type: text/plain
Return Code: <[archive name] or -1 or output>
0 [archive name] Successful completion of the URL command
-1 Error in execution of the URL command
Error String: <error-message>
```

Example

The following command removes an archive 3001.

```
http://<host>/cgi-bin/smanager.bwt?command=remove&name=3001
```

Archive Backup Command

Backs up recorded data to a repository.

To initiate an archive backup, post an application xml file to the following URI:

```
http://<hostname>/archive_backup.bwt
```

Archive Backup XML File Format

The format of the xml file is as follows:

```
<archive_backup>
    <archive_name>axis243q_archive</archive_name>
    <backup_name>axis243q_backup</backup_name>
    <backup_type>all</backup_type>
    <remote_address>psbu-cjwhite-lnx.cisco.com</remote_address>
    <remote_port>80</remote_port>
    <retention_days>0</retention_days>
    <compress>0</compress>
    <secure>0</secure>
```

```
    <backup_period type="range" >
        <start_hour>0</start_hour>
        <start_minute>0</start_minute>
        <end_hour>1</end_hour>
        <end_minute>0</end_minute>
    </backup_period>
    <backup_period type="lastHours" >
        <last_hours>24</last_hours>
    </backup_period>
    <backup_period type="sinceLastBackup" />
    <schedule>
        <hour>06</hour>
        <minute>50</minute>
        <daysofweek>all</daysofweek>
    </schedule>
    <schedule>
        <hour>06</hour>
        <minute>50</minute>
        <daysofmonth>1,15</daysofmonth>
    </schedule>
</archive_backup>
```

In this file:

- <archive_name>—Name of the archive to be backed up.
- <backup_name>—Name under which the backup is stored on the backup server.
- <backup_type>—**all** or **events**.
- <remote_address> —Host name or IP address of the receiving backup server.
- <remote_port> (*optional*)—Defines the HTTP/HTTPS port of the remote system. Default values are 80/445.
- <retention_days> (*optional*)—Defines the life of the archive backup file, in days. 0 means indefinite. The default value is 7 days.
- <compress> (*optional*)—Has values of 0 or 1 and defines whether files are compressed before they are sent. The default value is 0 (no compression).
- <secure> (*optional*)—Has values 0 or 1 and defines whether a HTTPS connection is uses. The default value is 0 (no HTTPS).
- <backup_period> defines the type of archive backup:
  - range—Time range between 0:00 and 23:59 that is specified to define files to be backed up.
  - lastHours —Previous N hours of files are backed up.
  - sinceLastBackup—All files since the last backup are backed up.
- <schedule> is optional and defines:
  - start time in the range 00:00 thru 23:59, which specifies when the backup starts
  - Either of the following:

    - daysofweek—a comma separated list of three-letter acronyms (mon, tue, wed, thu, fri, sat, sun) for days of week that the backup should run or all

    - daysofmonth—a comma separated list of days of month (1 through 31) the backup should run

Obtaining a Backup Report

To receive a backup report, send one of the following URLs:

- http://<hostname>/archive_backup?type=report&format=summary—Provides a summary report

- http://<hostname>/archive_backup.bwt?type=report&format=detail&archive=<archive_name>—Provides a detailed report

The system returns an application/xml form that contains the following information, depending on the type of report that you requested:

- Summary report:

```
<backup_summary>
    <archive name="archive1" >
        <status>pending|running|success|failed</status>
        <start_time>NNNNNN</start_time>
        <end_time>NNNNNN</end_time>
        <files_sent>nnnn</files_sent>
    </archive>
    <archive name="archive1" >
        <status>pending|running|success|failed</status>
        <start_time>NNNNNN</start_time>
        <end_time>NNNNNN</end_time>
        <files_sent>nnnn</files_sent>
    </archive>
</backup_summary>
```

Detailed report:

```
<backup_detail>
    <archive name="archive1" >
        <status>pending|running|success|failed</status>
        <start_time>NNNNNN</start_time>
        <end_time>NNNNNN</end_time>
        <files_sent>nnnn</files_sent>
        <bytes_sent>nnnnn</bytes_sent>
        <logfile size="nnn" >
log file contents
        </logfile>
    </archive>
</backup_detail>
```

In the returned information:

- start_time and end_time are specified in seconds since 00:00 01/01/1970
- end_time, files_sent, and logfile are included if the backup executed

Clipping Commands

Clip Command

```
http://<host>/cgi-bin/smanager.bwt?command=save&source=<source_id>&startutc=<utc date in
msec>&stoputc=<utc date in msec>&name=<target_id >&savemode=<[local]>&desc=<description
text>&repos=<repository>&saveformat=<[regular | bwm | bwx]>&daystolive=<n>&notifyurl=<[URL
to notification handler]> &nolog=<0 | 1>
```

Table 3-8 Required Fields

Command	Description
`host`	Format: `[hostname.domain | IP address]` Web address of host where VSMS is running. **Note** VSMS runs on port 80 by default. Specify additional ports here similar to vsms_host.cisco.com:8080.
`command`	Reserved value: [save]
`source`	Format: [source_id] source_id (required): The source archive name. This is the parent archive from which to create a clip.
`startutc`	Format: [UTC milliseconds] Start date of the child clip in UTC milliseconds. Verify the parent archive contains data for this date.
`stoputc`	Format: [UTC milliseconds] Stop date of the child clip in UTC milliseconds. Make sure the parent archive contains data for this date.

Table 3-9 Optional Fields

Command	Description
`name`	Format: [target_id@address] Character class: [0-9 | A-Z | a-z | _ | -]; Reserved: (-1); Length: (1-127) target_id: The new archive clip name. If no name is specified, the name will be generated by VSMS. <source ID>_<month MM>_<day DD>_<year YYYY>_<hours hh-24>_<minutes mm>_<seconds ss>_<hundredths .ØØ> For example, lobby_07_02_2003_15_25_52_01.
`desc`	Character class: [0-9 | A-Z | a-z | _ | <space> | -]; Reserved: (-1); Length: (0-20) Brief description for the new archive clip. **Note** For an archive clip to have type clip in the archive listing (/info.bwt?type=archive), a description entry is not required. The description will default to clip.

Table 3-9 Optional Fields (continued)

Command	Description
repos	Format: [repository_mount] Location where the new archive clip will be saved. Saves clips directly to the repository mount location. Only one mount will be recognized. If no mount is specified, then the clip repository must be specified using the repos field in the save clip XML API request.
saveformat	Format: [regular | smd | virtual](default=regular) Type of archive clip to generate: regular archive clip, single file smd clip, or virtual clip.
key	Format: Character class: [0-9 | A-Z | a-z | _ | <space> | -]; Reserved: (-1); Length: (6-64) The key used to sign the smd archive. The key is not stored in the archive.
daystolive	Days to live where liveNum value specifies the number of days (starting from the day the clip is created) that the clip is stored before being removed from the repository. Valid liveNum values are integers. The default value is 0, which permanently stores the clip in the repository. **Note** The daystolive value is inherited from the parent archive, regardless of the format.
notifyurl	Format: [http://<host>/handler_path] URL to send upon the successful completion or failure of a clip. This is used to report status to the application after the clipping process finishes execution.

Return Values

A standard HTTP/1.x header followed by:

```
Content-Type: text/plain
Return Code: <0 or -1 output>
0 Successful completion of the URL command
-1 Error in execution of the URL command
```

VSMS will return a status code after the parameters have been validated. Save clip will happen in the background. VSMS will not send a second return code when the clip is completed, but a handler can be configured at the save clip notifyurl to receive notification if a clip has succeeded or failed.

Example

The following command saves a clip from archive southexit, to localhost on port 80, beginning at 1020530754089 (UTC milliseconds) and ending at 1020530786232 (UTC milliseconds). VSMS will create the name for this archive clip and create a virtual clip on the local host.

```
http://vsms_host/cgi-bin/smanager.bwt?command=save&startutc=1020530754089&stoputc=10205307
86232&source=southexit&savemode=local&saveformat=regular
```

Event Recording Commands

Event Profile Setup

Command

```
http://<host>/event.bwt?command=setup&data=<xml data>
```

Table 3-10 Required Fields

Command	Description
host	Format: [hostname.domain | IP address] Web address of host where VSMS is running. **Note** VSMS runs on port 80 by default. Specify additional ports here similar to vsms_host.cisco.com:8080.
command	Reserved value: [setup]
data	Reserved value: [xml data]

Table 3-11 Parameters

Field	Description
xml	Start XML parsing tag; contains event tag and child.
event	Start event data tag; contains name, ipdevice, srctype, notifyurl and trigger tags and child.
name	Character class: [0-9 | A-Z | a-z | _ | -]; Reserved: (-1); Length: (1-32) Name for this event; contains no other tags.
ipdevice	Format: [hostname.domain | IP address] Web address where trigger is set up or a unique ID for generic or soft triggers; contains no other tags.
srctype	Reserved values: All supported devices. Specifies the type of video server to set up the trigger; contains no other tags. **Note** Sending soft triggers from other devices or applications is supported by the generic <srctype>. Use input of Ø with a unique ID for ipdevice.

Table 3-11 Parameters (continued)

Field	Description
notifyurl	Format: [http://<host>/handler_path] The URL will send when an event trigger is received by VSMS, if archive clips are requested and after archives are saved; contains no other tags. Use this tag in conjunction with the notificationtype tag. For notification types: • 0: An event notification is sent. • 2: An event notification and after event clip saved notification are sent.
cliphost	Start cliphost data tag; contains localhost and/or remotehost tag(s). **Note** If no cliphost tag is specified, then the event clip is saved to local host.
localhost	Tag Format: <localhost/>. Event clip is saved to local host. Save the event clip directly to the repository mount location. Only one mount will be recognized. If no repository is specified, here then an error will be generated when an event clip is attempted. This repository will also serve as a workspace area for remote event clip generation. **Note** The "Local Event Clip Repository" parameter must be specified on the VSM Console.
action	Start action data tag; contains clip and/or accelerate tag(s). **Note** If the action tag is not specified but notificationtype is set to 2 (record event triggered archives), event clips will still be recorded.
clip	Tag Format: <clip/>. Makes a clip when event occurs. **Note** notificationtype must be set to 2 (record event triggered archives).
accelerate	Tag Format: <accelerate/> Accelerate the event archive recording framerate at the event for the postbuffer time. **Note** notificationtype must be set to 2 (record event triggered archives).

Table 3-11 Parameters (continued)

Field	Description
input	Range: [0] Reserved for generic trigger input number. Make sure to pair with unique ID for ipdevice value. Range: [1-6] Trigger input number on device. Range: [1-10] Window number for motion detection.
state	Reserved values: [rising | falling] Specifies whether the circuit for the event trigger mechanism is open (rising) or closed (falling).
type	Reserved values: [motion | alarm] Specifies whether the type of event is motion detection or trigger.
notificationtype	Reserved values: [0 | 1 | 2 | 3] • 0: Only track events, no archives • 1: Unsupported • 2: Record event triggered archives • 3: Unsupported
maxevents	Format: [integer per month] Maximum number of events recorded per month.
daystolive	Format: [integer in days](default=30) Number of days event files (–E) are kept on disk before getting groomed away. For permanent storage, set daystolive=Ø.
framerate	Range: [0.001-30] (proxy framerate) Maximum number of frames per second transmitted to record proxy.
acclframerate	Range: [0.001-30] Accelerated archive framerate that the event is recorded at. **Note** Accelerated framerate must be less than or equal to the proxy framerate.
duration	Format: [integer in seconds] (3600 seconds) Duration of the event archive. The minimum duration value is 300 seconds.
prebuffer	Format: [integer in seconds] (10 seconds) Amount of seconds before event that will be included in event archive clip.

Table 3-11 Parameters (continued)

Field	Description
postbuffer	Format: [integer in seconds] (30 seconds) Amount of seconds after event that will be included in event archive clip.
proxysource	Character class: [0-9 | A-Z | a-z | _ | -]; Reserved: (-1); Length: (1-64) Each triggered event can record up to 10 different sources (proxies or archives). **Note** The proxy must exist prior to adding an event trigger.

Return Values

A standard HTTP/1.x header followed by:

```
Content-Type: text/plain
Return Code: <1 or -1 or output>
1 Successful completion of the URL command
-1 Error in execution of the URL command
Error String: <error-message>
```

Archive Information Commands

List All Archives

The list all archives command displays information for all archives (RUNNING, PAUSED or SHELVED) on a VSMS host.

Command

```
http://<host>/info.bwt?type=archive&display=<[html | text | ssv]>
```

Table 3-12 Required Fields

Command	Description
host	Format: [hostname.domain | IP address] Web address of host where VSMS is running. **Note** VSMS runs on port 80 by default. Specify additional ports here similar to vsms_host.cisco.com:8080.
type	Reserved value: [archive]

Table 3-13 Optional Fields

Command	Description
display	Reserved values: [html | text | ssv](default=html)

Return Values

A standard HTTP/1.x header followed by:

```
Content-Type: text/plain
Return Code: <[archive info] or -1 or output>
[archive info] Successful completion of the URL command
Table containing the following columns: ID, Directory Path, Type, Actual Size(kb), Status,
Begin Time, End Time, Expire Time
-1 Error in execution of the URL command
Error String: <error-message>
```

Example

The following command displays all archive files in host vsms_host.

```
http://vsms_host/info.bwt?type=archive
```

List All Running Archives

The list running archive command displays information for all currently running archives. When specifying an archive name, only that archive's information will be displayed.

Command

```
http://<host>/info.bwt?type=archiver&name=<archiveid>&display=<[html | text | ssv]>
```

Table 3-14 Required Fields

Field	Description
host	Format: [hostname.domain | IP address] Web address of host where VSMS is running. **Note** VSMS runs on port 80 by default. Specify additional ports here similar to vsms_host.cisco.com:8080.
type	Reserved value: [archiver]

Table 3-15 Optional Fields

Field	Description
name	Character class: [0-9 | A-Z | a-z | _ | -]; Reserved: (-1); Length: (1-127) Archive name to display information. **Note** All running archives will be listed if no name is displayed.
display	Reserved values: [html | text | ssv](default=html)

Return Values

A standard HTTP/1.x header followed by:

```
Content-Type: text/plain
Return Code: <[archive info] or -1 or output>
[archive info] Successful completion of the URL command
Table containing the following columns: exec, media type, name, source, frame/bitrate,
quality, width, height, loop, duration, video_file size(K), DaysToLive, starting_time,
kill proxy
-1 Error in execution of the URL command
Error String: <error-message>
```

Example

The following command lists archive information for archive baybridge24hrloop.

```
http://vsms_host/info.bwt?type=archiver&name=baybridge24hrloop
```

Get Archive MediaType

The get archive media type command displays the media type for an archive.

Command

```
http://<host>/info.bwt?type=archive&name=<archive id>&property=mediatype
```

Table 3-16 Required Field

Field	Description
host	Format: [hostname.domain | IP address] Web address of host where VSMS is running. **Note** VSMS runs on port 80 by default. Specify additional ports here similar to vsms_host.cisco.com:8080.
type	Reserved value: [archive]
name	Character class: [0-9 | A-Z | a-z | _ | -]; Reserved: (-1); Length: (1-127) Archive being queried for property information.
property	Reserved value: Returns JPEG, MPEG or audio values.

Return Values

A standard HTTP/1.x header followed by:

```
Content-Type: text/plain
Return Code: <[jpeg | mpeg2-v | mpeg4-v | audio] or -1 or output>
[jpeg | mpeg2-v | mpeg4-v | audio] Successful completion of the URL command
-1 Error in execution of the URL command
Error String: <error-message>
```

Example

The following command gets the media type for archive dayatglance.

```
http://vsms_host/info.bwt?type=archive&name=8745&property=mediatype
Returns: jpeg
```

Get Archive Monitoring Detail

The get archive monitoring detail command displays performance details for an archive.

Command

```
http://<host>/info.bwt?type=archive&name=<archive id>&property=armon_detail
```

Table 3-17 Required Field

Field	Description
host	Format: [hostname.domain | IP address] Web address of host where VSMS is running. **Note** VSMS runs on port 80 by default. Specify additional ports here similar to vsms_host.cisco.com:8080.
type	Reserved value: [archive]
name	Character class: [0-9 | A-Z | a-z | _ | -]; Reserved: (-1); Length: (1-127) Archive being queried for property information.
property	Return value: XML information that shows details about the archive.

Return Values

XML information that shows details about the archive.

Example

The following command gets the XML information that shows detailed recording information for the archive dayatglance.

```
http://vsms_host/info.bwt?type=archive&name=dayataglance=&property=armon_detail
Returns:
<archive-detail>
<archive>
<name>dayataglance</name>
<path>/media1/1576</path>
<archive-type>loop</archive-type>
```

```
<source>iqeye705_5M</source>
<width>2560</width>
<height>1920</height>
<status>RUNNING</status>
<mediatype>jpeg</mediatype>
<recording-rate>21.548613</recording-rate>
<quality>50</quality>
<framerate>10.000000</framerate>
<bitrate>640</bitrate>
<time>3600</time>
<expires>1</expires>
<archive-Size>8820492</archive-Size>
<start-Time>Mon Dec  1 10:27:21 2008</start-Time>
<end-Time>Mon Dec  1 11:27:21 2008</end-Time>
<max-fps>7.503949</max-fps>
<max-frame-size>434418</max-frame-size>
</archive>
</archive-detail>
```

Archive Summary

Displays recording rate information for an archive.

Command

```
http://<host>/info.bwt?type=archive&name=<archive id>&property=armon_summary.ex
```

Table 3-18 Required Field

Field	Description
host	Format: [hostname.domain | IP address] Web address of host where VSMS is running. **Note** VSMS runs on port 80 by default. Specify additional ports here similar to vsms_host.cisco.com:8080.
type	Reserved value: [archive]
name	Character class: [0-9 | A-Z | a-z | _ | -]; Reserved: (-1); Length: (1-127) Archive being queried for property information.
property	Return value: XML information that shows recording rate of the archive in MB per second.

Return Values

XML information that shows recording rate of the archive in MB per second

Example

The following command gets the XML information that shows detailed recording information for the archive dayatglance.

```
http://vsms_host/info.bwt?type=archive&name=dayataglance=&property=armon_summary
Returns:
<archive-summary>
<count>1</count>
```

```
<total-recording-rate>21.548613</total-recording-rate>
</archive-summary>
```

CHAPTER 4

Event Commands

Event commands are structured around the capability of client devices to send alerts to VSMS when an event occurs. Client devices can be set up to send notifications for physical triggers or motion detection. Using the URL-based API, it is possible for an HTTP-enabled application to send soft (on-screen or email notifications) triggers or alerts to VSMS. Alerts are used by VSMS to capture pre and post-event archive clips and uploads them to a remote or local VSMS host or to track the events as they occur.

Event Setup

The event command uses the generic srctype and an <input> of Ø with a unique <ipdevice> value to set up events for systems not directly related to video encoding to send soft triggers. The <ipdevice> value must be unique as it used by VSMS internally as part of the unique key (input and ipdevice) for events. Then configure the trigger to send the correct URL notification to VSMS.

`event.bwt`

VSMS can generate event clips where the post event portion of the clip is captured at a higher or accelerated framerate. When setting up the event, specify the <clip/> and <accelerate/> tags within the <action> tags. The event buffered archive is started at a framerate specified in the <framerate> tag. When an event occurs, the post event portion of the clip is recorded at the accelerated framerate specified in the <acclframerate> tag. To view an accelerated event clip, use the AXClient. It is also possible to set up event profiles that simply accelerate for the postbuffer time in the associated archive or archives when the event is triggered. To accelerate without clipping, specify only the <acclerate/> tag for the <action> parameter.

Note Changing the framerate the event buffered archive will affect how the allocated storage gets used. Allocated storage is based on the recording framerate specified in the <framerate> tag at the time the event buffered archive is started and assumed constant for the loop duration. Increasing the recording framerate will shorten the duration of the archive loop because the allocated storage will get used up sooner as more frames are written onto storage. During each loop more storage can be allocated but the amount of storage can never be de-allocated, hence space may be wasted if the frequency of events is significantly less than the number of events anticipated.

Command

`http://<host>/event.bwt?command=setup&data=<xml data>`

Table 4-1 Required Fields

Command	Description
host	Format: [hostname.domain | IP address] Web address of host where VSMS is running. **Note** VSMS runs on port 80 by default. Specify additional ports similar to vsms_host.cisco.com:8080.
command	Reserved value: [setup]
data	Reserved value: [xml data]

Table 4-2 Parameters

Command	Description
xml	Start XML parsing tag; contains event tag and child.
event	Start event data tag; contains name, ipdevice, srctype, notifyurl, cliphost, and trigger tags and child.
name	Character class: [0-9 | A-Z | a-z | _ | -]; Reserved: (-1); Length: (1-32) Name for this event; contains no other tags.
ipdevice	Format: [hostname.domain | IP address] Web address where trigger is set up or a unique ID for generic or soft triggers; contains no other tags.
srctype	Reserved values: Specifies the type of video server to set up the trigger; contains no other tags. **Note** Sending soft triggers from other devices or applications is supported by the generic <srctype>. Use input of Ø with a unique ID for ipdevice.

Table 4-2 Parameters (continued)

Command	Description
notifyurl	Format: [http://<host>/handler_path] URL to send when an event trigger is received by VSMS, and if archive clips are requested, after archives are saved; contains no other tags. Use this tag in conjunction with the notificationtype tag. For notification types: 0: An event notification is sent. See Send Event Notification: from VSMS to Event-handler at Notify URL, page 4-11. 2: An event notification and after event clip saved notification are sent. For event clip notification, see Send Event Notification: from VSMS to Event-handler at Notify URL After Event Clip Saved, page 4-12 section. For after event clip saved notification, see Send Event Notification: from VSMS to Event-handler at Notify URL After Event Clip Saved, page 4-12.
cliphost	Start cliphost data tag; contains localhost and/or remotehost tag(s). **Note** If no cliphost tag is specified, then the event clip is saved to local host.
localhost	Tag Format: <localhost/>. Event clip is saved to local host. Save the event clip directly to the repository mount location. Only one mount will be recognized. If no repository is specified, here then an error will be generated when an event clip is attempted. This repository will also serve as a workspace area for remote event clip generation. **Note** The EVENT_REPOS parameter must be specified on the VSMC Console.

Table 4-2 Parameters (continued)

Command	Description
remotehost	Tag Format: <remotehost/>. Event clip is saved to remote host. Save the event clip directly to the repository mount location. Only one mount will be recognized. If no repository is specified, here then an error will be generated when an event clip is attempted. This repository will also serve as a workspace area for remote event clip generation. **Note** The EVENT_REPOS, EVENT_REMOTE_REPOS, and EVENT_REMOTE_HOST parameters must be specified on the VSMC Console. **Note** This tag does not support event notification for a remote host, use the notifyurl tag.
action	Start action data tag; contains clip and/or acclerate tag(s). **Note** If the action tag is not specified but notificationtype is set to 2 (record event triggered archives), event clips will still be recorded.
clip	Tag Format: <clip/>. Make a clip when event occurs. **Note** The notificationtype must be set to 2 (record event triggered archives).
accelerate	Tag Format: <accelerate/> Accelerate the event archive recording framerate at the event for the postbuffer time. **Note** The notificationtype must be set to 2 (record event triggered archives).
input	Range: [0] Reserved for generic trigger input number. Make sure to pair with unique ID for ipdevice value. Range: [1-6] Trigger input number on device. Range: [1-10] Window number for motion detection.
state	Reserved values: [rising | falling] Specifies whether the circuit for the event trigger mechanism is open (rising) or closed (falling).
type	Reserved values: [motion | alarm] Specifies whether the type of event is motion detection or trigger.

Table 4-2 Parameters (continued)

Command	Description
notificationtype	Reserved values: [0 | 1 | 2 | 3] 0: Only track events, no archives 1: Unsupported 2: Record event triggered archives 3: Unsupported
maxevents	Format: [integer per month] Maximum number of events recorded per month.
daystolive	Format: [integer in days](default=0) Number of days from the date archive stops the archive will be stored before system removal. For permanent storage set daystolive=Ø.
framerate	Range: [0.001-30](proxy framerate) Maximum number of frames per second transmitted to record proxy.
acclframerate	Range: [0.001-30] Accelerated archive framerate that the event is recorded at. **Note** The accelerated framerate must be less than or equal to the proxy framerate.
duration	Format: [integer in seconds] (900 seconds) Duration of the event archive loop. **Note** The minimum duration value is 300 seconds.
prebuffer	Format: [integer in seconds] (10 seconds) Amount of seconds before event that will be included in event archive clip.
postbuffer	Format: [integer in seconds] (30 seconds) Amount of seconds after event that will be included in event archive clip.
proxysource	Character class: [0-9 | A-Z | a-z | _ | -]; Reserved: (-1); Length: (1-64) Proxy name for event to archive. Each triggered event can record up to 10 different proxies. **Note** The proxy must exist before adding an event trigger.

Return Values

A standard HTTP/1.x header followed by:

```
Content-Type: text/plain
Return Code: <1 or -1 or output>
    1 Successful completion of the URL command
    -1 Error in execution of the URL command
        Error String: <error-message>
```

Example

The following command will set up for a generic or soft trigger with a unique ID for ipdevice to archive event videos from proxy (3827) and proxy (13131) with a 10 second pre-buffer and a 30 second post-buffer at one frame per second. Event clips are defaulted to save on the local host.

```
http://host/event.bwt?command=setup&data=<xml data>
```

Example

The following command will set up a <device> video server to archive event videos from proxy westsidedoor with a 10 second pre-buffer and a 20 second post-buffer at five frames per second. Event clips are saved locally and remotely.

```
http://host/event.bwt?command=setup&data=<xml data>
```

Example

The following command will set up a <device> video server to capture accelerated event clips from proxy mainEntrance (where proxy framerate is 20 fps) at 20 frames per second at post event. The event buffered loop archive is started a 1 frame per second and upon an event the recording framerate will accelerate to 15 frames per second for 20 seconds. Duration of the event buffered archive is 2 hours (7200 seconds). The event clips are saved to localhost.

```
http://host/event.bwt?command=setup&data=<xml data>
```

Enable Event

Enabling an event is accomplished by sending a HTTP request to VSMS with the event name to be enabled. This command will enable the event setup on the device.

Command

```
http://<host>/event.bwt?command=enable&name=<event name>
```

Table 4-3 Required Fields

Command	Description
host	Format: [hostname.domain | IP address] Web address of host where VSMS is running. **Note** VSMS runs on port 80 by default. Specify additional ports similar to vsms_host.cisco.com:8080.
command	Reserved value: [enable]
name	Character class: [0-9 | A-Z | a-z | _ | -]; Reserved: (-1); Length: (1-32) Name for the event to be enabled.

Disable Event

Disabling an event is accomplished by sending a HTTP request to VSMS with the event name to be disabled. This command will disable the event setup on the device.

Command

```
http://<host>/event.bwt?command=disable&name=<event name>
```

Table 4-4 Required Fields

Command	Description
host	Format: [hostname.domain | IP address] Web address of host where VSMS is running. **Note** VSMS runs on port 80 by default. Specify additional ports similar to vsms_host.cisco.com:8080.
command	Reserved value: [disable]
name	Character class: [0-9 | A-Z | a-z | _ | -]; Reserved: (-1); Length: (1-32) Name for the event to be enabled.

Remove Event

Removing an event setup is accomplished by sending a HTTP request to VSMS with the event name to be removed. As events require a unique name even for the same device, remove requests do not need the trigger parameter. The command will remove the event setup for the trigger only on the device.

Command

```
http://<host>/event.bwt?command=remove&name=<event name>&killarchive=<[true | false]>
```

Table 4-5 Required Fields

Command	Description
host	Format: [hostname.domain | IP address] Web address of host where VSMS is running. **Note** VSMS runs on port 80 by default. Specify additional ports here similar to vsms_host.cisco.com:8080.
command	Reserved value: [remove]
name	Character class: [0-9 | A-Z | a-z | _ | -]; Reserved: (-1); Length: (1-32) Name for the event to be enabled.

Table 4-6 Optional Fields

Command	Description
killarchive	Format: [true | false] (default=true) Specifies whether the buffered event archive is removed or not when event is removed. true: Buffered event archive is stopped and removed from storage. false: Buffered event archive keeps running after the event is removed.

Return Values

A standard HTTP/1.x header followed by:

```
Content-Type: text/plain
Return Code: <1 or -1 or output>
    1 Successful completion of the URL command
    -1 Error in execution of the URL command
        Error String: <error-message>
```

Send Event Notification: from Trigger Device to VSMS

When an event occurs, the device will send an HTTP request to notify VSMS an event has occurred. The HTTP request contains parameters for the event name. This command can also be used to manually tag events. Since event names are required to be unique on a given host, input numbers are not required.

Command

```
http://<host>/event.bwt?command=event&name=<trigger name>
```

Table 4-7 Required Fields

Command	Description
host	Format: [hostname.domain | IP address] Web address of host where VSMS is running. **Note** VSMS runs on port 80 by default. Specify additional ports similar to vsms_host.cisco.com:8080.
command	Reserved value: [event]
name	Character class: [0-9 | A-Z | a-z | _ | -]; Reserved: (-1); Length: (1-32) Name for the event to be enabled.

Return Values

A standard HTTP/1.x header followed by:

```
Content-Type: text/plain
Return Code: <1 or -1 or output>
    1 Successful completion of the URL command
```

```
-1 Error in execution of the URL command
    Error String: <error-message>
```

Send Event Notification: from Motion Detection Device to VSMS

When motion above the defined threshold occurs, the device will send an HTTP request to notify VSMS an event has occurred. The HTTP request contains parameters for the event name. Since event names are required to be unique on a given host, motion window numbers are not required.

Command

```
http://<host>/event.bwt?command=event&name=<trigger name>
```

Table 4-8 Required Fields

Command	Description
host	Format: [hostname.domain | IP address] Web address of host where VSMS is running. **Note** VSMS runs on port 80 by default. Specify additional ports similar to vsms_host.cisco.com:8080.
command	Reserved value: [event]
name	Character class: [0-9 | A-Z | a-z | _ | -]; Reserved: (-1); Length: (1-32) Name for the event to be enabled.

Table 4-9 Optional Fields

Command	Description
utc	Format: [UTC milliseconds] date of post event in UTC milliseconds. **Note** VSMS runs on port 80 by default. Specify additional ports similar to vsms_host.cisco.com:8080.
archive name	Reserved value: [event]

Return Values

VSMS will respond with a HTTP NO-CONTENT response since these requests will be coming from the video server and response validation will not be handled by these devices.

Send Event Notification: from Soft Trigger to VSMS for Post Event Logging

Post log an event with a soft trigger is accomplished by sending an HTTP request to VSMS. The HTTP request contains parameters for the event name. Since event names are forced to be unique on a given host, trigger input numbers are no longer required. An event will be logged in the event database at the date/time specified by the utc parameter. No event clip archives are generated. In this way post-analytics or reviewing for items of interest can be tagged and easily looked up using VSMS event API commands.

When a soft trigger setup contains a notification URL, issuing this post log event notification command will then cause VSMS to send a post event logged notification. See Chapter 4, "Send Event Notification: from VSMS to Event-handler Notify URL Post Event Logged" section for additional information.

Command

```
http://<host>/event.bwt?command=event&name=<trigger name>&utc=<utc
milleseconds>&archivename=<archive name>
```

Table 4-10 Required Fields

Command	Description
host	Format: [hostname.domain | IP address] Web address of host where VSMS is running. VSMS runs on port 80 by default. Specify additional ports similar to vsms_host.cisco.com:8080.
command	Reserved value: [event]
name	Character class: [0-9 | A-Z | a-z | _ | -]; Reserved: (-1); Length: (1-32) Name for the event to be enabled.
utc	Format: [UTC milliseconds] date of post event in UTC milliseconds.

Table 4-11 Optional Fields

Command	Description
archive name	Character class: [0-9 | A-Z | a-z | _ | -]; Reserved: (-1); Length: (1-127) Name of the archive associated with the event. Since no archive clips are automatically extracted with post-event logging, this clip must be extracted and named by another application or handler.

Return Values

A standard HTTP/1.x header followed by:

```
Content-Type: text/plain
Return Code: <1 or -1 or output>
    1 Successful completion of the URL command
    -1 Error in execution of the URL command
```

```
Error String: <error-message>
```

Send Event Notification: from VSMS to Event-handler at Notify URL

VSMS sends a notification if the <notifyurl> is specified in the XML when the event was set up. Along with the notify URL, the following XML data, defined in the event trigger setup, is sent to a host. This URL needs to have a handler running to parse the XML data and that can respond to the notification.

Command

```
<notifyurl>?info=<xml data>
```

Note The <notifyurl> is defined in the event setup it must be fully qualified similar to http://event_host/.

Table 4-12 Parameters

Command	Description
xml	Start XML parsing tag; contains event tag and child.
TriggerNotification	Start trigger data tag; contains Host, EventUTC, Name, SrcType, TriggerInput, and ProxyList tags.
Host	Format: [hostname.domain | IP address] Web address of host where VSMS is running. VSMS runs on port 80 by default. Specify additional ports similar to vsms_host.cisco.com:8080.
EventUTC	Format: [UTC milliseconds] Date of the event in UTC milliseconds. This date is when VSMS received notification of the event from the encoder.
Name	Character class: [0-9 | A-Z | a-z | _ | -]; Reserved: (-1); Length: (1-32) Name for this event; contains no other tags.
IpDevice	Format: [hostname.domain | IP address] Web address where trigger is set up; contains no other tags.
SrcType	Reserved values: [Supported Devices] Specifies the type of video server to set up the **Note** Sending soft triggers from other devices or applications is supported by the generic <srctype>. Use input of Ø with a unique ID for IpDevice.

Table 4-12 Parameters (continued)

Command	Description
TriggerInput	Range: [0] Generic Trigger input number. Range: [1-6] Trigger input number on device. Range: [1-10] Window number for motion detection.
ProxyList	Start proxylist data tag; contains ProxyName tag.
ProxyName	Character class: [0-9 | A-Z | a-z | _ | -]; Reserved: (-1); Length: (1-64) Proxy name for event. **Note** Proxy must exist before adding an event trigger.

Send Event Notification: from VSMS to Event-handler at Notify URL After Event Clip Saved

When event archives are requested for this event setup, this notification will be sent only after each archive clip has been created. In other words, a notification is sent per event archive clip after it has been saved.

VSMS sends a notification if the <notifyurl> is specified in the XML when the event was set up. Along with the notify URL, the following XML data, defined in the event trigger setup, is sent to a host. This URL needs to have a handler running to parse the XML data and that can react to the notification.

Command

```
<notifyurl>?data=<xml data>
```

Note The <notifyurl> is defined in the event setup and must be fully qualified similar to http://event_host/.

Table 4-13 Parameters

Command	Description
xml	Start XML parsing tag; contains TriggerNotification tag and child.
TriggerNotification	Start trigger data tag; contains Host, VideoServer, TriggerInput, ProxyName, ArchiveName, StartUTC, and Duration tags.
Host	Format: [hostname.domain | IP address] Web address of host where VSMS is running.
RemoteHost	Format: [hostname.domain | IP address] Web address of remote host.
VideoServer	Character class: [0-9 | A-Z | a-z | _ | -]; Reserved: (-1); Length: (1-32) Name of this event as defined in Event Setup as <name>.

Table 4-13 Parameters (continued)

Command	Description
TriggerInput	Range: [0] Generic Trigger input number. Range: [1-6] Trigger input number on device. Range: [1-10] Window number for motion detection.
ProxyName	Character class: [0-9 | A-Z | a-z | _ | -]; Reserved: (-1); Length: (1-64) Proxy name for event archive. **Note** A proxy must exist prior to adding an event trigger.
ArchiveName	Character class: [0-9 | A-Z | a-z | _ | -]; Reserved: (-1); Length: (1-127) Archive name for this event.
StartUTC	Format: [UTC milliseconds] Start date of the archive in UTC milliseconds.
Duration	Format: [integer in seconds] Total archive time in seconds.

Send Event Notification: from VSMS to Event-handler Notify URL Post Event Logged

This notification is sent in response to the soft trigger post log event command to post an event entry into the event database.

VSMS sends a notification if the <notifyurl> is specified in the XML when the event was set up. Along with the notify URL, the following XML data, defined in the event trigger setup, is sent to a host. This URL needs to have a handler running to parse the XML data and that can react to the notification.

Command

```
<notifyurl>Info?=<xml data>
```

Note The <notifyurl> is defined in the event setup and must be fully qualified similar to http://event_host/.

Table 4-14 Parameters

Command	Description
xml	Start XML parsing tag; contains TriggerNotification tag and child.
TriggerNotification	Start trigger data tag; contains Host, EventUTC, Name, IpDevice, SrcType, TriggerInput, and ArchiveName tags.

Table 4-14 Parameters (continued)

Command	Description
Host	Format: [hostname.domain | IP address] Web address of host where VSMS is running. **Note** VSMS runs on port 80 by default. Specify additional ports similar to vsms_host.cisco.com:8080.
EventUTC	Format: [UTC milliseconds] Event date in the archive in UTC milliseconds.
Name	Character class: [0-9 | A-Z | a-z | _ | -]; Reserved: (-1); Length: (1-32) Name of this event as defined in Event Setup as <name>.
IpDevice	Format: [hostname.domain | IP address] Web address where trigger is set up; contains no other tags.
SrcType	Reserved values: [Supported Devices] Specifies the type of video server to set up the trigger. **Note** Sending soft triggers from other devices or applications is supported by the generic <srctype>. Use input of Ø with a unique ID for ipdevice.
TriggerInput	Range: [0] Generic Trigger input number. Range: [1-6] Trigger input number on device. Range: [1-10] Window number for motion detection.
ArchiveName	Character class: [0-9 | A-Z | a-z | _ | -]; Reserved: (-1); Length: (1-127) Archive name for this event.

Event Clip Stop

A clip will be created from the start command time minus the prebuffer time up to the time of the stop command. If the stop command is not issued before the post-buffer time elapses, the clip will be stopped when the post-buffer time is attained.

When the event has been setup to start the clip, use the soft-trigger:

Command

```
http://<host>/event.bwt?command=event&name=<trigger name>
```

To stop the clip, a type parameter assumes the values start and stop. The default is start.

Command

```
http://<host>/event.bwt?command=event&name=<trigger name>type=stop
```

Event Clip Error Notification

For error notification, use the current XML data tags used for notification. If there is a failure in clip creation, the ArchiveName value should be set to -1.

Command

```
<notifyurl>?data=<xml data>
```

Note The <notifyurl> is defined in the event setup it must be fully qualified similar to http://event_host/.

Get Event Information

Retrieving event information is accomplished by sending a HTTP request to VSMS. Requests can be sent to retrieve information based on query property, event name, or start and stop dates.

Command

```
http://<host>/info.bwt?type=event&property=<[setup | proxies | archives]>&name=<event
name>&startutc=<utc date in msec>&stoputc=<utc date in msec>&display=<[html | text | ssv]>
```

Table 4-15 Required Fields

Command	Description
host	Format: [hostname.domain | IP address] Web address of host where VSMS is running. **Note** VSMS runs on port 80 by default. Specify additional ports similar to vsms_host.cisco.com:8080.
command	Reserved value: [event]
property	Reserved values: [setup | proxies | archives] setup List all events set up on the VSMS host being queried. proxies: List all events, and the proxies they will use to record event archives. Events with multiple proxies will be listed per proxy. archives: List of each event trigger received, will include any event archives that were requested. In other words, an event set up with trigger tracking only will be returned without archive clip names.

Table 4-16 Optional Fields

Command	Description
name	Character class: [0-9 | A-Z | a-z | _ | -]; Reserved: (-1); Length: (1-32) Name of the event being queried. If no name is given, VSMS returns all events.
startutc	Format: [UTC milliseconds] Start date filter for archives in UTC milliseconds. **Note** Used when property=archives
stoputc	Format: [UTC milliseconds] Stop date filter for archives in UTC milliseconds. **Note** Used when property=archives
display	Reserved values: [html | text | ssv](default=html)

Return Values

A standard HTTP/1.x header followed by:

```
Content-Type: text/plain
Return Code: <[archive info] or -1 or output>
    [archive info] Successful completion of the URL command
        For property=archives, table containing the following columns: name, input/window,
        type, archive name, time
        For property=proxies, table containing the following columns: name, input/window,
        type, proxysource
        For property=setup, table containing the following columns: name, ipdevice,
        srctype, notifyURL, input/motion, type, state, notificationtype, maxevents,
        daystolive, framerate, prebuffer, postbuffer, totalevents, cliphost, action,
        acclframerate, duration
    -1 Error in execution of the URL command
        Error String: <error-message>
```

Example

The following command displays all the event archives:

```
http://host/info.bwt?type=event&property=archives
```

Example

The following command displays all the event archives for event named abc:

```
http://host/info.bwt?type=event&property=archives&name=abc
```

Example

The following command displays all the event archives for event named abc from 1018642188000 UTC to 1018642228000 UTC:

```
http://host/info.bwt?type=event&property=archives&name=abc&startutc=1018642188000&stoputc=
1018642228000
```

Example

The following command displays all event archives for events named abc from 101864218800 UTC to 1018642228000 UTC in ssv format:

```
http://host/info.bwt?type=event&property=archives&name=abc&startutc=1018642188000&stoputc=
1018642228000&display=
```

Record on Event

Motion Event Configuration and Event Handling

The following steps discuss motion event configuration and event handling.

Step 1 An event profile is added in VSOM and associated with the required feed(s) on which motion detection is to be tracked. The actions are configured to occur upon event along with the relevant parameters to perform the action such as pre-buffer, post-buffer, rate, and resolution.

Step 2 VSOM will send the event profile information to VSMS via the event.bwt apache module adding it as a software (soft) trigger (trigger input # 0, srctype generic). The event handler will parse the command and start the archives based on the actions to be performed when a motion event occurs.

Command and sample xml

```
event.bwt?command=setup&
data=<xml><event>
    <name>e_SampleEvent</name>
    <ipdevice>1207870147</ipdevice>
    <srctype>generic</srctype>    I
    <notifyurl> http://10.10.50.32/vsom/event_notify.php?</notifyurl>
    <trigger>
        <input>0</input>
        <state>rising</state>
        <type>alarm</type>
        <notificationtype>2</notificationtype>
        <maxevents>0</maxevents>
        <daystolive>30</daystolive>
        <framerate>5</framerate>
        <duration>300</duration>
        <prebuffer>30</prebuffer>
        <postbuffer>60</postbuffer>
        <proxysource>p_SampleFeed</proxysource>
    </trigger>
</event></xml>
```

Step 3 On the VSOM motion configuration page, motion windows are configured on the applicable feed and the previously setup soft-trigger event profile is registered with this configuration.

Step 4 VSOM sends the motion configuration data to VSMS through the motion.bwt handler.

Step 5 Motion.bwt parses the data, writes it into conf/motion/proxy_name.xml, and notifies the proxy.

Step 6 The proxy communicates the motion configuration information to the device including the server and URL to notify when a motion occurs.

Step 7 When motion is detected, the device sends a motion start command to VSMS via the motionrecv.bwt apache module.

Step 8 The motionrecv.bwt apache handler forwards the message onto the proxy motion driver.

Step 9 The proxy motion driver notifies VSOM using the starturl URL setup during motion configuration.

Step 10 VSOM sends a start event command to the VSMS event.bwt module. The event module will perform the necessary actions such as update properties, start recording, and mark as event.

Command

```
event.bwt?command=event&name=<e_SampleEvent>&type=start&nolog=1
```

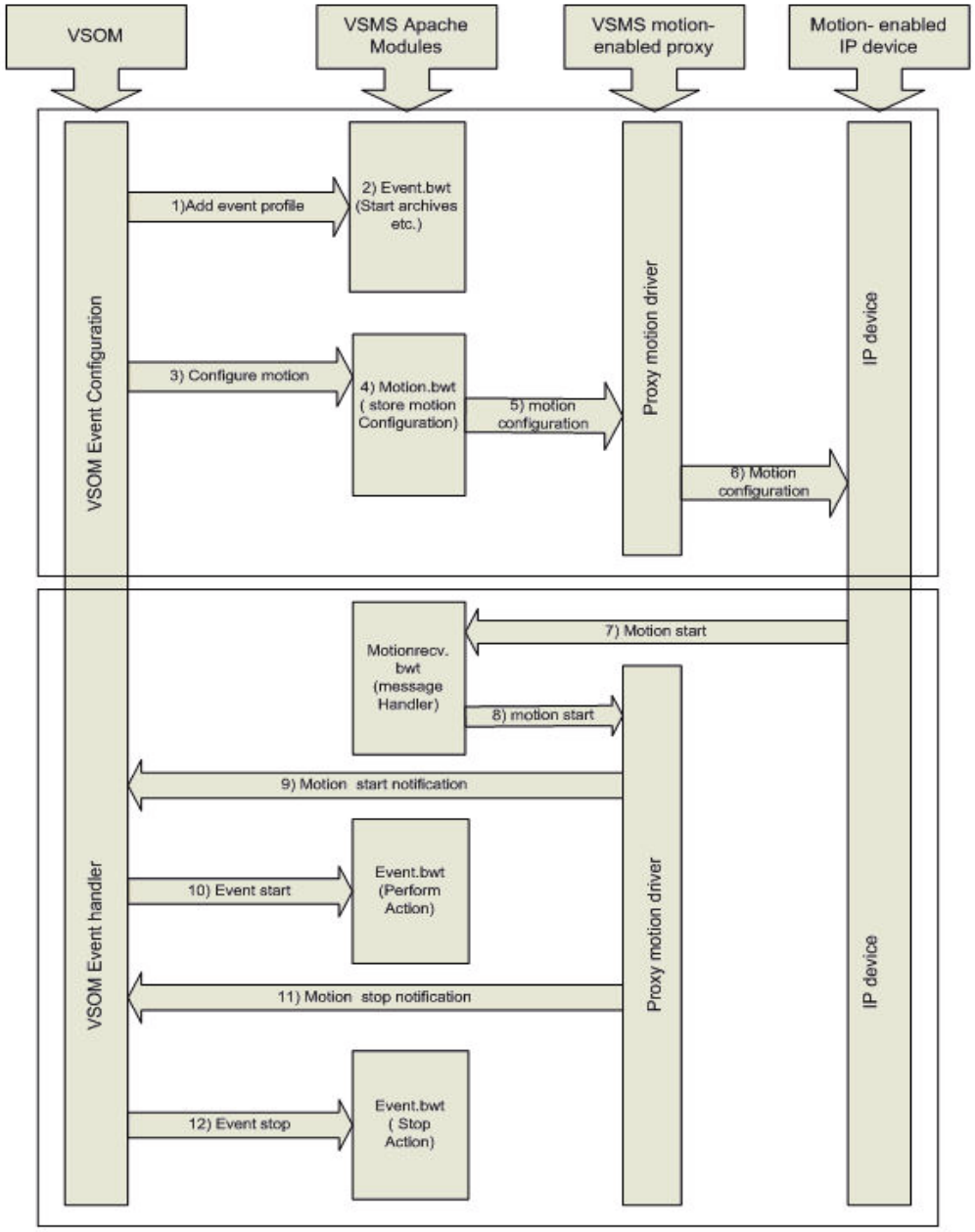
The proxy motion driver keeps track of all the windows it receives motion start commands for. It also monitors the time elapsed since the last motion start command was received for any window exceeding the ttl_motion_events. The ttl_motion_events was configured in conf/devices/Cisco-avg.xml and the default is one second. A motion stop command is sent to VSOM using the stopurl URL, setup during motion configuration.

Step 11 VSOM sends a stop event command to the VSMS event.bwt module. The event module with perform necessary actions such as set back feed properties and stop recording after post-buffer.

Command

```
event.bwt?command=event&name=<e_SampleEvent>&type=stop&nolog=1
```

Figure 4-1 Flow diagram for motion event configuration and handling



Single Alarm (trigger) Event Configuration and Handling

The following steps discuss adding alarm triggered event configurations and handling triggered events.

Step 1 An event profile is added in VSOM and associated with the required feed(s) on which motion detection is to be tracked. The actions are configured to occur upon event along with the relevant parameters to perform the action such as pre-buffer, post-buffer, rate, and resolution.

Step 2 VSOM sends the event profile information to VSMS through the event.bwt apache module. The event handler parses the command and starts the archives depending on the actions to be performed when an event occurs. For devices such as Cisco_avg, the event driver will update the device so that the device communicates with the server with the relevant information when events occur via the following command.

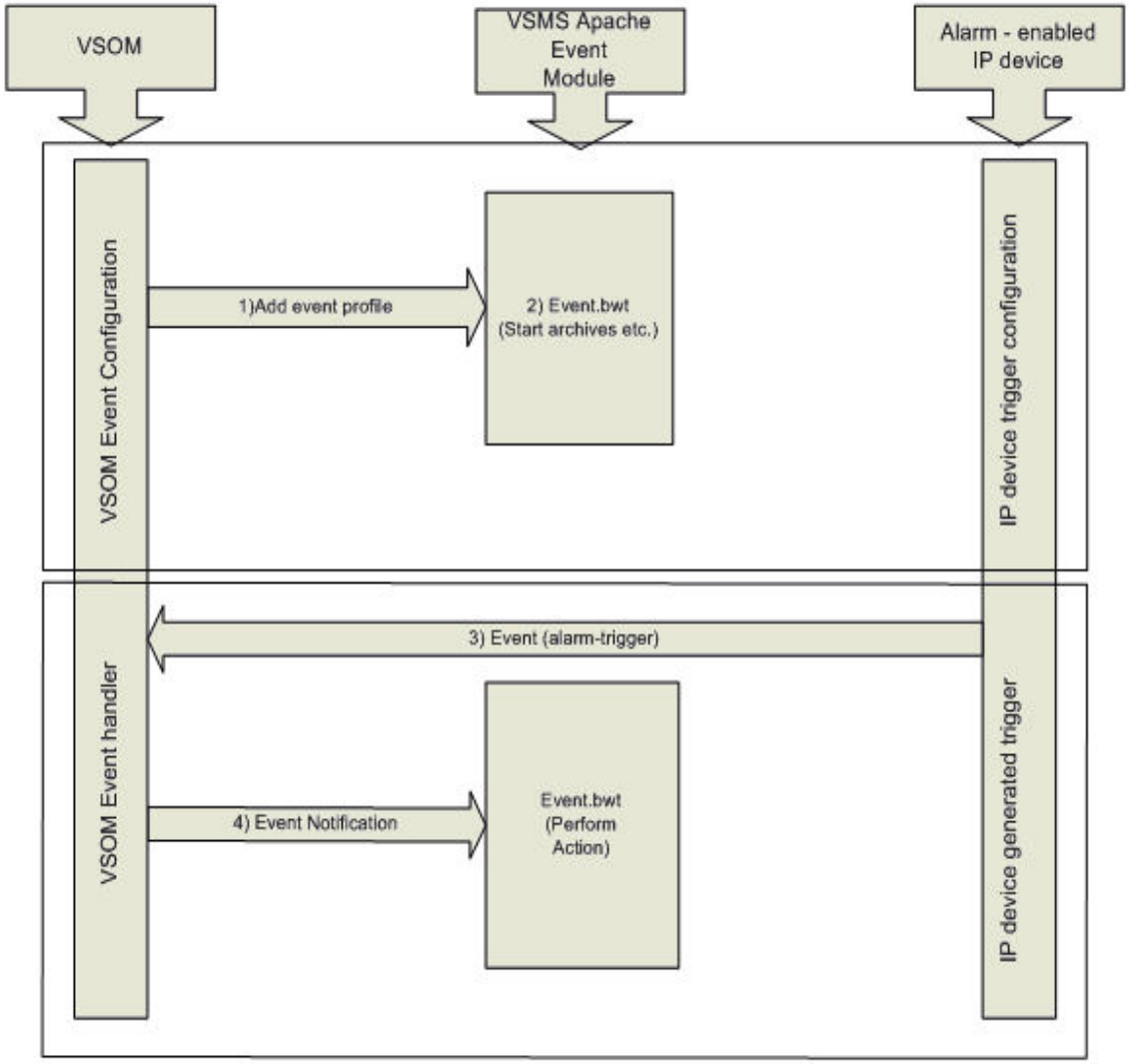
Command

```
event.bwt?command=event&name=<e_SampleEvent>
```

Step 3 When the event.bwt command is received from the IP device, the actions setup in the event profile are performed by VSMS and VSOM is notified that the event occurred.

Step 4 Once the event.bwt module finishes processing the event, it notifies VSOM with the status of the actions taken.

Figure 4-2 Single alarm flow diagram



Soft Trigger Event Configuration and Handling

Soft triggers are used when VSOM generates events in response to particular feedback. The following steps discuss adding soft triggered event configurations and handling triggered events.

Step 1 An event profile is added in VSOM and associated with the required feed(s) along with the actions to occur upon event with the relevant parameters for the action such as pre-buffer, post-buffer, rate, and resolution.

Step 2 VSOM sends the event profile information to VSMS through the event.bwt apache module. The event handler parses the command and starts the archives depending on the actions to be performed when an event occurs.

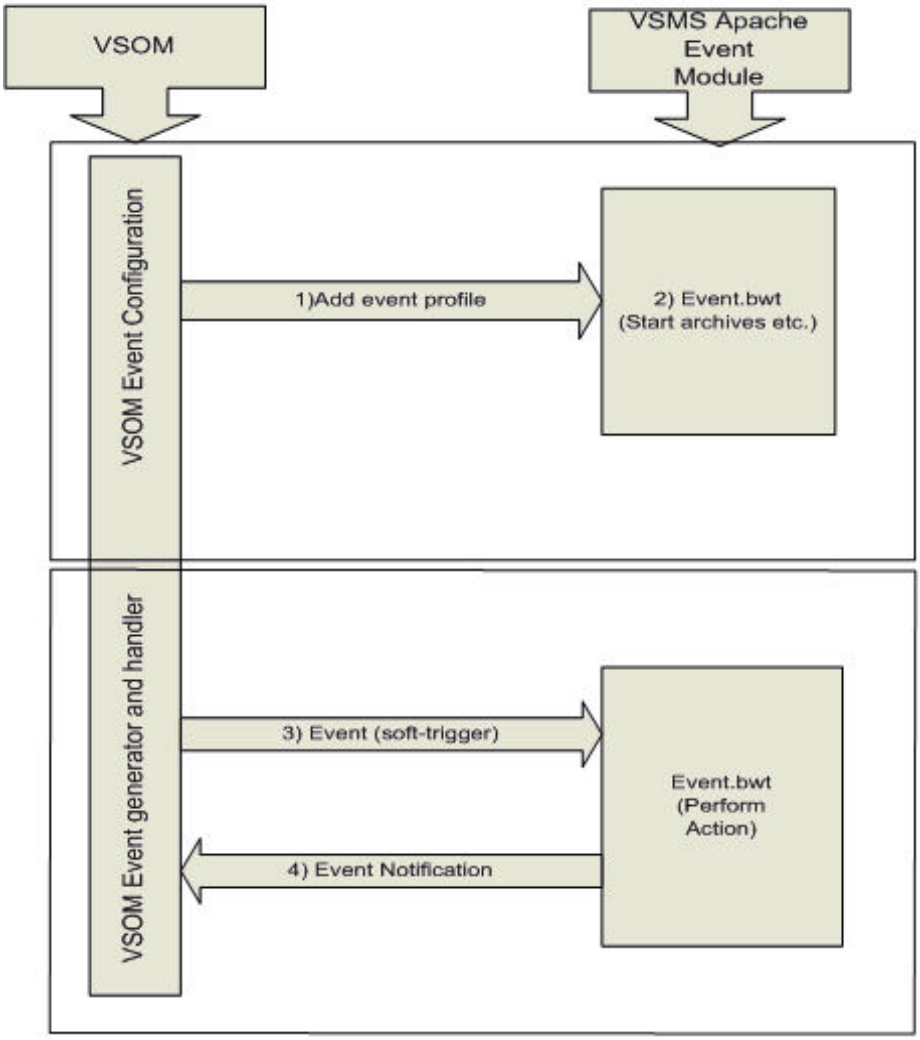
Step 3 The application sends an event.bwt command to trigger the event. When the event.bwt command is received from the IP device, the actions setup in the event profile are performed by VSMS and VSOM is notified that the event occurred.

Command

```
event.bwt?command=event&name=<e_SampleEvent>
```

Step 4 Once the event.bwt module finishes processing the event, it notifies VSOM with the status of the actions taken.

Figure 4-3 Flow diagram for soft trigger event configuration and handling



CHAPTER 5

AXClient API

Use the APIs listed in this section for archive controls such as play, pause, seek and live controls such as pan, tilt, zoom, and presets. These methods are available via the following prefixes:

- get methods query information
- is methods query state or type of information
- on methods fire events to defined handlers or call-backs
- set methods usually take arguments and pass them for a given action to take place
- setOn methods define callback functions not defined in the <APPLET> tag in addition to other methods defining simple actions such as play, pause, seek, and cameraMove.

AXClient Programming Notes

C

There are currently no C Programming Notes

C++

There are currently no C++ Programming Notes

C#

The AXClient is written in C++ and inherently has different object types than C#. Many return values are different objects than referenced in the API documentation. Specifically:

- Return values of types date will return to C# as System.DateTime.
- Return values of types int, short and long will return to C# as System.Int32.

JavaScript

The AXClient is written in C++ and inherently has different object types than JavaScript. Many return values are not native to JavaScript and require special consideration before then can be used. Specifically:

- Return values of type date are not native JavaScript Date objects. These return values will need to be recast to a JavaScript date object by passing it through the JavaScript Date constructor.
- Return values of types int, short and long will return to JavaScript as an integer.
- Return values of types float and double will return to JavaScript as a float.

- AXClient Parameter values that should be a Date must be case in JavaScript to a VT_DATE prior to passing to the AXClient. This is performed by using the Date object's getVarDate() method.

Sample Code:

```
// AXClient instantiation
<object codebase="AXClient.cab#version=5,0,17,0"
classid="clsid:41293422-93FD-443C-B848-E07EDBF866C3"
id="AXClient" name="AXClient" viewastext="true" width="350" height="280">
// Static Properties
<param name="name" value="AXClient" />
<param name="EnableDvrMode" value="false"/>
<param name="EnableVMRMode" value="false"/>
// Setup Callbacks
<param name="OnPlayerLoaded" value="userDefinedCallbackMethod"/>
// Default Values
<param name="timestamp" value="0"/>
</object>
```

Note The AXClient version number will change from this documentation depending on which version of the AXClient is being used to develop against. The sample instantiation code above does not work around a known Microsoft Internet Explorer issue involving activating ActiveX Controls. See Chapter 6, "Interactive Media Clients" originally published by Microsoft for more information.

Method Descriptions

attach

```
public void attach(string mediaSourceURL)
```

Versions Supported: 4.5+

This method attaches the specified audio source to a video source already selected via the switchTo() method. This allows an audio source to be played along with a video source. Make sure to call attach() after calling switchTo(), but before calling playForward().

Parameters:

mediaSourceURL - Media Server source URL in the form of //host/source.

See Also:

```
switchTo()
playForward()
```

clearKeyCombinationHistory()

```
public void clearKeyCombinationHistory()
```

Versions Supported: 4.9+

Clears the buffer history of key combinations from the AXClient. Any subsequent usage of "previous" or "next" buttons from a CCTV keyboard or keyboard/joystick input USB device will not fire the onKeyCombination callback until a new history has been developed.

JavaScript Sample Code:

```
<script type="text/javascript">
AXC.clearKeyCombinationHistory();
</script>
```

See Also:

`setOnKeyCombination()`

close

```
public void close()
```

Versions Supported: 4.6+

Stops streaming the feed or archive and disconnects the client from the server. Source properties are flushed and reset. After calling close(), switchTo() must be performed again to reload the stream.

Completion of this method will invoke the callback function set via setOnStateChanged().

See Also:

`setOnStateChanged()`
`switchTo()`

getContentType

```
public string getContentType()
```

Versions Supported: 4.3+

Returns the media type of the currently loaded source

Return Values:

Returns the media type of the currently loaded source. This will be one of MPEG1, MPEG2, MPEG4, JPEG, or audio.

getCurrentSource

```
public string getCurrentSource()
```

Versions Supported: 3.2+

Returns the current source loaded by the AXClient.

Return Values:

Returns the mediaSourceURL loaded by the AXClient in the form of //host/source

getCurrentTime

```
public Date getCurrentTime()
```

Versions Supported: 3.2+

Returns the current time of the AXClient's current frame or seek position for an archive. Not supported for live feeds.

Return Values:

OLE Date object which represents the current time of the frame being viewed. Live feeds return -1

JavaScript Sample Code:

```
<script type="text/javascript">
if (axc = document.applets["axc_name"]) {
 currentTime = axc.getCurrentTime();
 currentDate = new Date(currentTime); // recast
}
</script>
```

getDisplayHeight

```
public short getDisplayHeight()
```

Versions Supported: 4.8

Returns the height the source should be displayed at. This may differ from the actual source height. For example, the display height will typically be larger than the value actual source height for 2CIF sources. This method should be used in conjunction with getDisplayWidth()

Return Values:

Integer value, in pixels, that the height of the source video should be displayed.

See Also:

```
getDisplayWidth()
getVideoHeight()
```

getDisplayWidth

```
public short getDisplayWidth()
```

Versions Supported: 4.8

Returns the width the source should be displayed at. This may differ from the actual source height. This method should be used in conjunction with getDisplayHeight()

Return Values:

Integer value, in pixels, that the width of the source video should be displayed.

See Also:

```
getDisplayHeight()
getVideoWidth()
```

getErrorText

```
public string getErrorText(float errorCode)
```

Versions Supported: 4.4+

Returns the text description for the specified error code.

Return Values:

String that contains the text description for the specified error code.

getOriginalStartTime

```
public Date getOriginalStartTime()
```

Versions Supported: 4.4+

Returns the original start date/time that archive was created. This start time does not change for regular or loop archives. For non-loop archives, this value will be equal to that of getStartTime(). For loop archives, this returns the start date/time that the archive was created, and not the first frame available in the archive. Once a loop archive loops, the video data that corresponds to the original start time is no longer available, and the return value will differ from that returned by getStartTime()

Return Values:

OLE Date Object which represents the archive creation time.

See Also:

```
getStartTime()
```

JavaScript Sample Code:

```
<script type="text/javascript">
if (axc = document.applets["axc_name"]) {
 origStartTime = axc.getOriginalStartTime();
 origStartDate = new Date(origStartTime); // recast
}
</script>
```

getPlayrate

```
public long getPlayrate()
```

Versions Supported: 4.0-4.4

Returns the playrate for the current media source. Depending on the media type of the source, this can either be treated as a framerate or a bitrate. Use getContentType() to determine the media type.

This method has been superseded by the getPlayrateEx() method.

Return Values:

The float value playrate of the current source. Either framerate for JPEG sources or bitrate for MPEG sources.

See Also:

```
getContentType()
getPlayrateEx()
setPlayrate()
setPlayrateEx()
```

getPlayrateEx

```
public double getPlayrateEx()
```

Versions Supported: 4.4+

Returns the playrate for the current media source. Depending on the media type of the source, this can either be treated as a framerate or a bitrate. Use getContentType() to determine the media type.

This method replaces both the getPlayrate() and getFramerate() methods.

Return Values:

The float value playrate of the current source. Either framerate for JPEG sources or bitrate for MPEG sources.

See Also:

```
getContentType()
getPlayrate()
setPlayrate()
setPlayrateEx()
```

getProfiles

```
public object getProfiles(string mediaSourceURL)
```

Versions Supported: 4.4+

Returns the list of available profiles supported by the AXClient. CBR means constant bit rate and VBR is variable bit rate.

Parameters:

mediaSourceURL - Media Server source URL in the form of //host/source.

Return Values:

Object representation of an array of media source profiles.

See Also:

```
getProfilesSSV()
```

JavaScript Sample Code:

```
<script type="text/javascript">
var mySource = "//server/source";
if (axc = document.applets["axc_name"]) {
 var profilesVB = axc.getProfiles(mySource);
 var profileArray = profilesVB.toArray();
 for (i in profileArray ) {
    // output the profiles
    alert(profileArray[i]);
  }
}
</script>
```

getProfilesSSV

```
public string getProfilesSSV(string mediaSourceURL)
```

Versions Supported: 4.4+

Returns the list of available profiles supported by the AXClient. CBR means constant bit rate and VBR is variable bit rate.

Parameters:

mediaSourceURL - Media Server source URL in the form of //host/source.

Return Values:

Semi-colon Separated Values list containing a list of media source profiles.

See Also:

`getProfiles()`

JavaScript Sample Code:

```
<script type="text/javascript">
var mySource = "//server/source";
if (axc = document.applets["axc_name"]) {
  var profileSSV = axc.getProfilesSSV(mySource);
  alert(profileSSV);
}
</script>
```

getRecordrate

```
public long getRecordrate(string recordrate)
```

Versions Supported: 4.0+

Returns the rate at which an archive is recorded at. Depending on the media type of the source, this can either be treated as a framerate or a bitrate. Use getContentType() to determine the media type. Not applicable for live feeds.

Return Values:

String representation of record rate; frames per second for JPEG, bitrate for MPEG. Returns -1 if the source is a live feed.

See Also:

`getContentType()`

getSeekTime

```
public Date getSeekTime()
```

Versions Supported: 4.3+

Returns the date/time that the archive will seek to on the next seek() command. The seek time can be set using the setSeekTime() or setSeekTimeByPercentage() methods. Not supported for live feeds.

Return Values:

OLE Date representation of the seek time.

Exceptions:

An exception will be thrown if called against a live source.

See Also:

```
setSeekTime()
setSeekTimeByPercentage()
seek()
```

JavaScript Sample Code:

```
<script type="text/javascript">
if (axc = document.applets["axc_name"]) {
  seekTime = axc.getSeekTime();
  seekDate = new Date(seekTime); // recast
}
</script>
```

getSkipFrames

```
public long getSkipFrames()
```

Versions Supported: 4.3+

Returns the number of frames the AXClient will skip during playback of a JPEG archive. A SkipFrames of 0 indicates the client is not skipping any frames and is playing every frame in the archive. A SkipFrames equal to 1 indicates the client is skipping 1 frame, in essence playing every other frame. Not supported for live feeds or MPEG archives.

Return Values:

The number of frames that will be skipped in the playback of a JPEG archive. Live feeds or MPEG archives will return 0.

getSnapshotDIB

```
public VBArray getSnapshotDIB()
```

Versions Supported: 4.4.1+

Creates a snapshot of the current frame in standard Windows DIB format that can be saved to a .bmp file.

Return Values:

Array of bytes that is in a standard Windows DIB format.

getStartTime

```
public Date getStartTime()
```

Versions Supported: 4.3+

Returns the date/time of the archives current start time. The start time for loop archives progresses after the archive has looped at least one time. Once looped, the start time can be thought of as the stop time - duration. For non-loop archives, the start time is equivalent to the date returned by getOriginalStartTime(). If the stream is paused, the AXClient will not update the start or stop times of the archive until it is playing again. Not supported for live feeds.

Return Values:

OLE Date object representation of the start time for a given archive. Live feeds return -1.

See Also:

```
getOriginalStartTime()
getStopTime()
pause()
playForward()
```

JavaScript Sample Code:

```
<script type="text/javascript">
if (axc = document.applets["axc_name"]) {
  startTime = axc.getStartTime();
  startDate = new Date(startTime); // recast
}
</script>
```

getState

```
public string getState()
```

Versions Supported: 4.3+

Returns the current state of the source within the AXClient.

Return Values:

String representation of the playback state integer. State definitions are:

```
0 : Paused
1 : Playing Forward
2 : Playing Reverse
3 : Searching/Seeking
4 : Loading
5 : Stopped
```

getStopTime

```
public Date getStopTime()
```

Versions Supported: 4.3+

Returns the date/time of the archives current stop (end) time. The stop time for loop archives is continuously changing. If the stream is paused, the AXClient will not update the start or stop times of the archive until it is playing again. Not supported for live feeds.

Return Values:

OLE Date object representation of the stop time for a given archive. Live feeds return -1.

See Also:

```
getStartTime()
pause()
playForward()
```

JavaScript Sample Code:

```
<script type="text/javascript">
if (axc = document.applets["axc_name"]) {
  stopTime = axc.getStopTime();
  stopDate = new Date(stopTime); // recast
}
</script>
```

getVersion

```
public string getVersion()
```

Versions Supported: 4.1+

Returns the version number of the AXClient. Version is returned with 4 decimals: #.#.#.#, similar to 5.0.17.0.

Return Values:

String representation of the AXClient version number.

getVideoHeight

```
public short getVideoHeight()
```

Versions Supported: 4.6+

Returns the actual source height, in pixels. The actual source height may differ from the height that the source should be displayed. This method should be used in conjunction with getVideoWidth().

Return Values:

Integer value, in pixels, of the height of the source stream.

See Also:

```
getDisplayHeight()
getVideoWidth()
```

getVideoWidth

```
public short getVideoWidth()
```

Versions Supported: 4.6+

Returns the actual source width, in pixels. The actual source width may differ from the width that the source should be displayed. This method should be used in conjunction with getVideoHeight()

Return Values:

Integer value, in pixels, of the width of the source stream.

See Also:

```
getDisplayWidth()
getVideoHeight()
```

mute

```
public void mute(boolean mute)
```

Versions Supported: 3.2+

Turns the sound on and off for MPEG-2 video streams with embedded sound. This method is not applicable for audio proxies or audio proxies that are attached to video sources.

Parameters:

mute - TRUE turns off the sound; FALSE turns on the sound.

See Also:

```
attach()
```

mtStartStream

```
HRESULT mtStartStream(BSTR source, LONG version, VARIANT_BOOL bProxy);
```

Description

Starts a separate thread to begin filtering the graph with the stream source. This method is equivalent to calling both switchTo() and playForward; therefore, playForward() does not need to be called after mtStartStream() is invoked.

For multiple streams, there is an option in the registry (IMSClient\disableMTStartStream - default to off) to start all streams simultaneously. When this option is turned on, starting NxM panels is significantly faster than N*M times the startup time of a single stream.

Parameters:

source - String containing the URI of the feed.

version - Server version (5 or 6)

bProxy - Boolean whether source is from proxy or archive.

VARIANT_TRUE is from proxy.

VARIANT_FALSE is from archive.

JavaScript Sample Code:

```
axc= //instantiate Active X control
document.write(axc);
this.axcName = 'AX_' + this.name;
this.axc = document.applets[this.axcName];
this.axc.mtStartStream(source, version_number, !this.isArchive);
source - URL to the source
```

```
version_number - Server Version Number
true- indicates proxy
false- indicates archive
```

See Also:

```
mtStreamStarting()
```

mtStreamStarting

```
HRESULT mtStreamStarting(VARIANT_BOOL* pVal);
```

Description

Queries the system to determine if a stream is currently in the process of starting.

To handle the asynchronous nature of the video startup, VSOM will disable the video playback control buttons. When disabled, the user can still click on the video control buttons, but the JavaScript code calls mtStreamStarting first before issuing the actual command. If mtStreamStarting indicates that the video is still in the process of being loaded, VSOM will display an error to inform the user.

If a video control button can affect more than one video pane, VSOM will ensure that all video panes affected are started by calling mtStreamStarting on all the affected video streams. The actual command is issued to each video only if all of the videos are ready. If not, VSOM displays an error message.

Parameters:

pVal- Pointer to boolean whether thread has been started.

VARIANT_TRUE if a stream is in the process of starting

VARIANT_FALSE if a stream is not in the process of starting i.e. a stream already started or no attempt to start a stream was made via mtStartStream

JavaScript Sample Code:

```
if(!this.axc.mtStreamStarting()) { //call other Axclient API's}
else {//error out}
```

See Also:

```
mtStartStream()
```

pause

```
public void pause()
```

Versions Supported: 4.3+

Pauses the play back of the archive sources or stops a live source on the current frame.

If the AXClient is left in the paused state for more than 10 minutes will cause VSMS to disconnect the archive stream. This is equivalent to calling the close() method.

Completion of this method will invoke the callback function set by setOnStateChanged().

See Also:

```
setOnStateChanged()
```

playForward

```
public void playForward()
```

Versions Supported: 4.3+

Play the loaded live stream or archive forwards. Typically used on the setOnStartOfStream() callback after a switchTo() has been issued.

Completion of this method will invoke the callback function set via setOnStateChanged().

See Also:

```
playRewind()
setOnStartOfStream()
setOnStateChanged()
switchTo()
```

playRewind

```
public void playRewind()
```

Versions Supported: 4.3+

Play the loaded archive backwards. This method is the same in all regards to playForward() with the exception that it plays an archive in the opposite direction. Not applicable for live sources.

Completion of this method will invoke the callback function set via setOnStateChanged()

See Also:

```
playForward()
setOnStateChanged()
```

repeatSegment

```
public void repeatSegment(Date seektime, long startOffset, long endOffset)
```

Versions Supported: 4.6+

Starts at playback of an archive at seektime, plays (and loops) to seektime + endOffset, then jumps back to seektime - startOffset. This looping behavior repeats indefinitely.

This method seeks the specified seektime in an archive and repeatedly plays the archive segment bounded by startOffset and endOffset. Units are in seconds. endOffset is a positive number. repeatSegment can be set to false to stop the repeat mode. repeatSegment by default is set to true

Parameters:

seektime - The starting date/time in the archive to loop the segment around.
startOffset - Offset, in seconds, to play past the seektime. Must be >= 0
endOffset - Offset, in seconds, to play prior to the seektime. Must be <= 0

Java Sample Code:

```
<script type="text/javascript">
var axc = document.applets["axc_name"];
seekTime = new Date(2005, 10, 14, 14, 11, 00);
axc.repeatSegment(seekTime.getVarDate(), 60, -120); // cast to VT_DATE date
</script>
```

C# Sample Code:

```
System.DateTime seekTime = new System.DateTime(2005, 10, 14, 14, 11, 00);
IMC.repeatSegment(seekTime, 60, -120);
```

C++ Sample Code:

```
SYSTEMTIME SysDtTime;
ZeroMemory( &SysDtTime, sizeof(SysDtTime));

SysDtTime.wYear = 2005;
SysDtTime.wMonth = 10;
SysDtTime.wDay = 14;
SysDtTime.wHour = 14;
SysDtTime.wMinute = 11;
SysDtTime.wSecond = 0;

DATE SeekTime;
SystemTimeToVariantTime( &SysDtTime, &SeekTime);

HRESULT hResult = pIMC->repeatSegment( SeekTime, 60, -120);
```

save

```
public void save(
               string starttime,
               string stoptime,
               string location,
               string name,
               string saveFormat
        )
```

Versions Supported: 3.2+

Saves a clip of the media and stores it on the VSMS host. The time required to save clips is based on the length of the clip and the framerate and bitrate of the archive. Clips with 60 minutes duration take around 2 minutes. VSMS sets sleep after every 600 milliseconds when the duration of the clip exceeds 5 minutes. To avoid this, save clips with No Sleep parameter = 1.

By default, clips will be saved to the Clip Repository as defined in the VSMS configuration. Setting the parameter location overrides this default location with the following:

local - saves the clip to the local VSMS host
localandremote - saves the clip to both the local and remote VSMS hosts
user - saves the clip to the user's local PC

Completion of this method will invoke the callback function set via setOnSaveResponse().

Parameters:

starttime - start time in UTC milliseconds
stoptime - end time in UTC milliseconds to stop saving
location - one of: remote | local | localandremote | user
name - name of clip
saveFormat - type of format to save the clip.

See Also:

```
setOnSaveResponse()
```

Code Example:

```
Save a clip to the client PC. Prompt the user for clip name.
AXC.save(1116919042000, 1116966622000, "user", "", "");
```

Code Example:

Save a regular archive clip on a VSMS local host.

```
AXC.save(1116919042000, 1116966622000, "local", "outdoorClip", "regular");
```

saveInPortableFormat

```
public void saveInPortableFormat(
              string mediaSourceURL
              string starttime,
              string stoptime,
              string destination,
              string profile
         )
```

Versions Supported: 4.4+

Saves a clip in WMV format. The time required to save clips is based on the length of the clip and the frame rate and bit rate of the archive. Clips with 60 minutes duration take around 2 minutes. VSMS sets sleep after every 600 milliseconds when the duration of the clip exceeds 5 minutes. To avoid this, save clips with No Sleep parameter = 1.

Completion of this method will invoke the callback function set via setOnSaveResponse().

Parameters:

mediaSourceURL - VSMS source URL in the form of //host/source. Empty string will use currently loaded archive.
starttime - Start time in UTC milliseconds
stoptime - End time in UTC milliseconds
destination - where clip is saved on local client or empty string to prompt user for destination.
profile - One of the profiles returned by getProfiles() or getProfilesSSV()

See Also:

```
getProfiles()
getProfilesSSV()
setOnSaveResponse()
```

Code Example:

Save a clip portable clip and do not specify the source and use the loaded archive in the AXClient as the source. Prompt the user to provide the clip destination.

```
AXC.saveInPortableFormat("",1116919042000,1116966622000,"","WMV CBR 500Kbits");
```

saveWithPasskey

```
public int saveWithPasskey(
          string starttime,
          string stoptime,
          string location,
          string name,
          string saveFormat,
          string passPhrase
     )
```

Versions Supported: 4.6+

Save a clip of the media. If the archive source media type is JPEG or MPEG-4, the clip will be saved in AVI format. If the archive source media type is MPEG-2, the clip will be saved in MP2 format. The time required to save clips is based on the length of the clip and the framerate and bitrate of the archive. Clips with 60 minutes duration take around 2 minutes. VSMS sets sleep after every 600 milliseconds when the duration of the clip exceeds 5 minutes. To avoid this, save clips with No Sleep parameter = 1.

By default, clips will be saved to the Clip Repository as defined in the VSMS configuration. Setting the parameter location overrides this default location with the following:

local - saves the clip to the local VSMS host
localandremote - saves the clip to both the local and remote VSMS hosts
user - saves the clip to the user's local PC

Completion of this method will invoke the callback function set via setOnSaveResponse().

Parameters:

starttime - Start time in UTC milliseconds
stoptime - End time in UTC milliseconds
location - one of: remote | local | localandremote | user
name - name of clip
saveFormat - can be regular, bwm, or bwx where the option bwx is for saving a clip in a tamper-proof bwx file.

passPhrase - minimum length of six characters

See Also:

`setOnSaveResponse()`

seek

```
public void seek ()
```

Versions Supported: 3.2+

Seek an archive to the date/time specified by setSeekTime() or setSeekTimeByPercentage() methods. Seeking should not be performed prior to the onStartOfStream callback has been received.

Completion of this method will invoke the callback function set via setOnSeekTimeChanged().

See Also:

`setOnSeekTimeChanged()`
`setOnStartOfStream()`
`setSeekTime()`
`setSeekTimeByPercentage()`

seekToPercentage

```
public void seekToPercentage(double percent)
```

Versions Supported: 4.6+

Specifies the seek position and performs the seek within an archive all in one method. The archive will jump to by the percentage (factor * 100) of the total length of the archive. This method replaces using both the setSeekTimeByPercentage() and seek() methods in order to issue a seek by percentage.

Seeking should not be performed prior to the onStartOfStream callback has been received. Completion of this method will invoke the callback function set via setOnSeekTimeChanged().

Example:

```
0.5 will set the seek time to be 50% of the archive. The factor must be between 0.0 and
1.0.
```

Parameters:

percent - Percentage factor to seek the archive. Between 0.0 and 1.0

See Also:

```
seek()
setSeekTimeByPercentage()
setOnSeekTimeChanged()
setOnStartOfStream()
```

seekToTime

```
public void seekToTime(Date time)
```

Versions Supported: 4.6+

Seek to a certain point in the archive. Seeking should not be performed prior to the onStartOfStream callback has been received. Completion of this method will invoke the callback function set via setOnSeekTimeChanged().

Parameters:

time - Date of which to seek an archive to.

See Also:

```
seek()
setSeekTime()
setOnSeekTimeChanged()
setOnStartOfStream()
```

JavaScript Sample code:

```
<script type="text/javascript">
if (axc = document.applets[imc_name]) {
// seek to 10 minutes ago
var myDate = new Date();
myDate.setMinutes( myDate.getMinutes() - 10 );
axc.seekToTime( myDate.getVarDate() );
}
</script>
```

setMotionStartEvent

```
public void setMotionStartEvent (string URL)
```

Versions Supported: 6.0

Used to set the motion start event.

setMotionStopEvent

```
public void setMotionStopEvent (string URL)
```

Versions Supported: 6.0

Used to set the motion stop event.

setOnAsyncMethodCompleted

```
public void setOnAsyncMethodCompleted(string callbackFunction)
```

Versions Supported: 4.6

The AXClient uses the callback function supplied to report back the result of invoking asynchronous methods.

Parameters:

callbackFunction - User defined function name.

Event Callback API:

```
void callbackFunction(
string name,
object asyncMethodDispId,
boolean succeeded,
integer errorNumber,
string errorDescription
);
```

This user defined function will be called when the AXClient has completed an asynchronous method call.

Callback Parameters:

callbackFunction - AXClient name where the original method was invoked
asyncMethodDispId - Asynchronous method Id
succeeded - true if the method worked, false otherwise
errorNumber - errorNumber of problem if succeeded == false
errorDescription - description of errorNumber

JavaScript Sample Code:

```
<script type="text/javascript">
function onAsyncMethodCompleted(name, asyncMethodDispId, succeeded, errorNumber,
errorDescription) {
if (asyncMethodDispId == DISPID_ASYNC_METHOD_switchTo) {
if (succeeded) {
alert("switchTo: SUCCEEDED");
} else {
alert("switchTo: FAILED");
```

```
}
}
}
AXC.setOnAsyncMethodCompleted("onAsyncMethodCompleted");
</script>
```

setOnEndOfStream

```
public void setOnEndOfStream(string callBackFunction)
```

Versions Supported: 4.5+

The AXClient uses the callback function supplied to notify when the source archive reaches the end or when a close() command is issued.

Parameters:

callbackFunction - User defined function name

Event Callback API:

```
void callbackFunction(string name);
```

This user defined function will be called when the source that an AXClient is playing has reached its end of the stream.

Callback Parameters:

name - AXClient name of the source that reached its end of stream

See Also:

```
close()
```

JavaScript Sample Code:

```
<script type="text/javascript">
function onEndOfStream(name) {
alert(name + " has reached the end of its stream.");
}
AXC.setOnEndOfStream("onEndOfStream");
</script>
```

setOnFrameChanged

```
public void setOnFrameChanged(string callBackFunction)
```

Versions Supported: 4.3-4.6

The AXClient uses the callback function supplied using this method to notify when JPEG archive frames change.

Parameters:

callbackFunction - User defined function name

Event Callback API:

```
void callbackFunction(string name, Date currenttime);
```

This user defined function will be called when the AXClient has trapped a mouse wheel event.

Callback Parameters:

name - AXClient name where the mouse wheel event was trapped
currenttime - The date/time of the frame being displayed

setOnKeyCombination

```
public void setOnKeyCombination(string callBackFunction)
```

Versions Supported: 5.0

The AXClient uses the callback function supplied using this method to notify when a CCTV Keyboard or Keyboard/Joystick key combination has been received.

Parameters:

callbackFunction - User defined function name

Event Callback API:

```
void callbackFunction(int monitorId,
                      int viewId,
                      int paneNum,
                      int cameraId);
```

This user defined function will be called when the loaded keyboard key combination has been received.

Callback Parameters:

If any one parameter is not provided by the CCTV keyboard or keyboard/joystick USB input device, an integer value of -1 will be returned.

monitorId - The monitor # as selected by the keyboard.

viewId - The view # as selected by the keyboard.

paneNum - The monitor pane # as selected by the keyboard.

cameraId - The camera/feed # as selected by the keyboard.

JavaScript Sample Code:

```
<script type="text/javascript">
function onKeyCombination(monitorId, viewId, paneNum, cameraId) {
   // perform action
}
AXC.setOnKeyCombination("onKeyCombination");
</script>
```

See Also:

```
clearKeyCombinationHistory()
```

setOnMouseButtonDown

```
public void setOnMouseButtonDown(string callBackFunction)
```

Versions Supported: 4.4.1+

The AXClient uses the callback function supplied using this method to notify mouse button down (mouse-click) events.

Parameters:

callbackFunction - User defined function name

Event Callback API:

```
void callbackFunction(string name, long button, integer posX, integer posY);
```

This user defined function will be called when the AXClient has trapped a mouse button down event.

Coordinate values are relative to the upper-left of the client window (0,0).

Button values are a single or a combination of values. Examples:

Ox0004 - (Ctrl key down)
0x0008 - (Shift key down)
0x0001 - (left mouse button down)
0x0002 - (right mouse button down)
0x0005 - combination of 0x0004 Ctrl key down and 0x0001 left mouse button down.

Callback Parameters:

name - AXClient name where the mouse button down event was trapped
button - Button ID of any mouse button or key combinations that were depressed at the time of the event
posX - Horizontal coordinate of the cursor within the AXClient window.
posY - Vertical coordinate of the cursor within the AXClient window.

JavaScript Sample Code:

```
<script type="text/javascript">
function onMouseButtonDown(name, button, posX, posY) {
// perform action
}
AXC.setOnMouseButtonDown("onMouseButtonDown");
</script>
```

setOnMouseButtonUp

```
public void setOnMouseButtonUp(string callBackFunction)
```

Versions Supported: 4.4.1+

The AXClient uses the callback function supplied using this method to notify mouse button up (mouse-release) events.

Parameters:

callbackFunction - User defined function name

Event Callback API:

void callbackFunction(string name, long button, integer posX, integer posY);

This user defined function will be called when the AXClient has trapped a mouse button up event.

Coordinate values are relative to the upper-left of the client window (0,0).

Button values are a single or a combination of values. Examples:

Ox0004 - (Ctrl key down)
0x0008 - (Shift key down)

Callback Parameters:

name - AXClient name where the mouse button up event was trapped
button - Button ID of any key combinations that were depressed at the time of the event
posX - Horizontal coordinate of the cursor within the AXClient window.
posY - Vertical coordinate of the cursor within the AXClient window.

JavaScript Sample Code:

```
<script type="text/javascript">
function onMouseButtonUp(name, button, posX, posY) {
// perform action
}
AXC.setOnMouseButtonUp("onMouseButtonUp");
</script>
```

setOnMouseButtonDblClick

```
public void setOnMouseButtonDblClick(string callBackFunction)
```

Versions Supported: 4.4.1+

The AXClient uses the callback function supplied using this method to notify mouse button double-click events.

Parameters:

callbackFunction - User defined function name

Event Callback API:

```
void callbackFunction(string name, long button, integer posX, integer posY);
```

This user defined function will be called when the AXClient has trapped a mouse button double-click event.

Coordinate values are relative to the upper-left of the client window (0,0).

Button values are a single or a combination of values. Examples:

Ox0004 - (Ctrl key down)
0x0008 - (Shift key down)

Callback Parameters:

name - AXClient name where the mouse button down event was trapped
button - Button ID of any mouse button or key combinations that were depressed at the time of the event
posX - Horizontal coordinate of the cursor within the AXClient window.
posY - Vertical coordinate of the cursor within the AXClient window.

JavaScript Sample Code:

```
<script type="text/javascript">
function onMouseButtonDblClick(name, button, posX, posY) {
// perform action
```

```
}
AXC.setOnMouseButtonDblClick("onMouseButtonDblClick");
</script>
```

setOnMouseMove

```
public void setOnMouseMove(string callBackFunction)
```

Versions Supported: 4.4.1+

The AXClient uses the callback function supplied using this method to notify mouse movement events.

Parameters:

callbackFunction - User defined function name

Event Callback API:

```
void callbackFunction(string name, long button, integer posX, integer posY);
```

This user defined function will be called when the AXClient has trapped a mouse move event occurs.

Coordinate values are relative to the upper-left of the client window (0,0).

Button values are a single or a combination of values. Examples:

Ox0004 - (Ctrl key down)
0x0008 - (Shift key down)
0x0001 - (left mouse button down)
0x0002 - (right mouse button down)
0x0005 - combination of 0x0004 Ctrl key down and 0x0001 left mouse button down.

Callback Parameters:

name - AXClient name where the mouse button down event was trapped
button - Button ID of any mouse button or key combinations that were depressed at the time of the event
posX - Horizontal coordinate of the cursor within the AXClient window.
posY - Vertical coordinate of the cursor within the AXClient window.

JavaScript Sample Code:

```
<script type="text/javascript">
function onMouseMove(name, button, posX, posY) {
// perform action
}
AXC.setOnMouseMove("onMouseMove");
</script>
```

setOnMouseWheelRotated

```
void setOnMouseWheelRotated(string callbackFunction);
```

Versions Supported: 4.6+

The AXClient uses the callback function supplied using this method to notify mouse wheel rotation events.

Parameters:

callbackFunction - User defined function name

Event Callback API:

```
void callbackFunction(string name, long deltaZ, long button, integer posX, integer posY);
```

This user defined function will be called when the AXClient has trapped a mouse wheel event.

Coordinate values are relative to the upper-left of the client window (0,0).

Button values are a single or a combination of values. Examples:

Ox0004 - (Ctrl key down)
0x0008 - (Shift key down)
0x0001 - (left mouse button down)
0x0002 - (right mouse button down)
0x0005 - combination of 0x0004 Ctrl key down and 0x0001 left mouse button down.

Callback Parameters:

name - AXClient name where the mouse wheel event was trapped
deltaZ - How far the mouse wheel moved
button - Button ID of any mouse button or key combinations that were depressed at the time of the event
posX - Horizontal coordinate of the cursor within the AXClient window.
posY - Vertical coordinate of the cursor within the AXClient window.

JavaScript Sample Code:

```
<script type="text/javascript">
function theMouseWheelRotated(name, deltaZ, button, posX, posY) {
// perform action
}
AXC.setOnMouseWheelRotated("theMouseWheelRotated");
</script>
```

setOnPlayerClicked

```
void setOnMouseWheelRotated(string callbackFunction);
```

Versions Supported: 4.4+

The AXClient uses the callback function supplied using this method to notify mouse click events.

Parameters:

callbackFunction - User defined function name

Event Callback API:

```
void callbackFunction(string name, integer posX, integer posY);
```

This user defined function will be called when the AXClient has trapped a mouse click event.

Coordinate values are relative to the upper-left of the client window (0,0).

Callback Parameters:

name - AXClient name where the mouse click event was trapped
posX - Horizontal coordinate of the cursor within the AXClient window.
posY - Vertical coordinate of the cursor within the AXClient window.

JavaScript Sample Code:

```
<script type="text/javascript">
function onPlayerClicked(name, deltaZ, button, posX, posY) {
// perform action
}
AXC.setOnPlayerClicked("onPlayerClicked");
</script>
```

setOnPlayerLoaded

Note This API method has been purposely left out of the IMC due to the fact that IMC must be loaded and defined before JavaScript call-backs can be defined. Therefore, a JavaScript function must define the Event Callback function at instantiation time, using the <OBJECT> property tags:

```
<param name="OnPlayerLoaded" value="userDefinedCallbackMethod"/>
```

Event Callback API:

```
void callbackFunction(string name);
```

This user defined function will be called when the AXClient has loaded.

Callback Parameters:

name - AXClient name where the mouse wheel event was trapped

JavaScript Sample Code:

```
<script type="text/javascript">
function onPlayerLoaded(name) {
// perform action
}
</script>
```

setOnPlayrateChanged

```
public void setOnPlayrateChanged(string callBackFunction)
```

Versions Supported: 4.3+

The AXClient uses the callback function supplied using this method to notify when the archive play rate has changed.

Parameters:

callbackFunction - User defined function name

Event Callback API:

```
void callbackFunction(string name, long rate);
```

This user defined function will be called when the archive playrate has changed.

Callback Parameters:

name - AXClient name playing the affected archive
rate - New play rate of archive. Frames per second for JPEG source and bit rate for MPEG source

See Also:

```
getPlayrate()
getPlayrateEx()
setPlayrate()
setPlayrateEx()
```

JavaScript Sample Code:

```
<script type="text/javascript">
function onPlayrateChanged(name, rate) {
// perform action
}
AXC.setOnPlayrateChanged("onPlayrateChanged");
</script>
```

setOnSaveResponse

```
public void setOnSaveResponse(string callBackFunction)
```

Versions Supported: 4.3+

The AXClient uses the callback function supplied using this method to notify that a previously initiated save clip has finished. The server response is the result of the initial save methods.

Parameters:

callbackFunction - User defined function name

Event Callback API:

```
void callbackFunction(string name, bool success, bool confirm, string location, string
message);
```

This user defined function will be called when a previously initiated save clip has finished.

The confirm parameter will always return false if clipping was saved to a local client PC. The client PCs can not return a confirmation message.

Callback Parameters:

name - AXClient name which processed the save clip command
success - true or false (boolean); whether clip was saved successfully
confirm - true or false (boolean); whether or not a clip confirmation was received
location - location clip was saved; one of: user, remote, local, localandremote
message - string representation of message returned by VSMS

See Also:

```
save()
saveInPortableFormat()
saveWithPasskey()
```

JavaScript Sample Code:

```
<script type="text/javascript">
function onSaveResponse(name, success, confirm, location, message) {
if (axc = document.applets[name]) {
source = axc.getCurrentSource();
if (success) {
alert("Clipping of "+ source +" was successful.");
} else {
alert("Clipping of "+ source +" failed.");
}
}
}
AXC.setOnSaveResponse("onSaveResponse");
</script>
```

setOnSeekTimeChanged

```
public void setOnSeekTimeChanged(string callBackFunction)
```

Versions Supported: 4.3+

The AXClient uses the callback function supplied using this method to notify when the seek time of an archive has changed. This callback is initiated when a seek(), seekToTime() or seekToPercentage() has completed.

Parameters:

callbackFunction - User defined function name

Event Callback API

```
void callbackFunction(string name, Date seektime);
```

This user defined function will be called when the archive seek time has changed.

Callback Parameters:

name - AXClient name which processed the save clip command
seektime - Current date/time of the archive after the performed seek operation

See Also:

```
getCurrentTime()
getSeekTime()
seek()
seekToTime()
seekToPercentage()
```

JavaScript Sample Code:

```
<script type="text/javascript">
function onSeekTimeChanged(name, seektime) {
var seekDate = new Date(seektime);
// perform action
}
```

```
AXC.setOnSeekTimeChanged("onSeekTimeChanged");
</script>
```

setOnSourceChanged

```
public void setOnSourceChanged(string callBackFunction)
```

Versions Supported: 4.3+

The AXClient uses the callback function supplied using this method to notify when the loaded source has changed.

Parameters:

callbackFunction - User defined function name

Event Callback API:

```
void callbackFunction(string name, string sourceID);
```

This user defined function will be called when the loaded source has changed.

Callback Parameters:

name - AXClient name which processed the save clip command
sourceID - The mediaSourceUrl provided in the SwitchToEx that caused the source to change

See Also:

```
getCurrentSource()
switchTo()
```

JavaScript Sample Code:

```
<script type="text/javascript">
function onSourceChanged(name, mediaSourceUrl) {
// perform action
}
AXC.setOnSourceChanged("onSourceChanged");
</script>
```

setOnStartOfStream

```
public void setOnStartOfStream(string callBackFunction)
```

Versions Supported: 4.3+

The AXClient uses the callback function supplied using this method to notify when the source has loaded. This callback is the primary notification method for when it is permitted to begin using get() method calls and handling other events. Typically this occurs after a switchTo() followed by a playForward(). Methods such as seekToTime() should wait until the onStartOfStream callback before use.

Parameters:

callbackFunction - User defined function name

Event Callback API:

```
void callbackFunction(string name);
```

This user defined function will be called when the source has started streaming.

Callback Parameters:

name - AXClient name which processed the save clip command

See Also:

```
playForward()
seekToTime()
seekToPercentage()
switchTo()
```

JavaScript Sample Code:

```
<script type="text/javascript">
function onStartOfStream(name) {
axc = document.applets[name];
// seek to 10 minutes ago
var myDate = new Date();
myDate.setMinutes( myDate.getMinutes() - 10 );
axc.seekToTime( myDate.getVarDate() );
}
AXC.setOnStartOfStream("onStartOfStream");
</script>
```

setOnStateChanged

```
public void setOnStateChanged(string callBackFunction)
```

Versions Supported: 4.3+

The AXClient uses the callback function supplied using this method to notify when the AXClient has changed state.

Parameters:

callbackFunction - User defined function name

Event Callback API:

```
void callbackFunction(string name, long state);
```

This user defined function will be called when the AXClient has changed state. State definitions are:

0 : Paused
1 : Playing Forward
2 : Playing Reverse
3 : Searching/Seeking
4 : Loading
5 : Stopped

Callback Parameters:

name - AXClient name which processed the save clip command
state - String representation of the playback state integer.

See Also:

```
getState()
```

setOnStartTimeChanged

```
public void setOnStartTimeChanged(string callBackFunction)
```

Versions Supported: 4.3+

The AXClient uses the callback function supplied using this method to notify when the archive source start time changes. Occurs approximately every second when VSMS updates the archive properties or when a new archive source is loaded. When the stream is paused, information is not being passed to AXClient so the start and stop time updates will not trigger.

Parameters:

callbackFunction - User defined function name

Event Callback API:

```
void callbackFunction(string name, Date starttime);
```

This user defined function will be called when the archive start time has changed.

Callback Parameters:

name - AXClient name which processed the save clip command
starttime - Date representation of the new archive start time

See Also:

```
getStartTime()
getStopTime()
setOnStopTimeChanged()
```

JavaScript Code Example:

```
<script type="text/javascript">
function onStartTimeChanged(name, starttime) {
startDate = new Date(starttime); // recast
// perform action
}
AXC.setOnStartTimeChanged("onStartTimeChanged");
</script>
```

setOnStopTimeChanged

```
public void setOnStartTimeChanged(string callBackFunction)
```

Versions Supported: 4.3+

The AXClient uses the callback function supplied using this method to notify when the archive source stop/end time changes. Occurs approximately every second when VSMS updates the archive properties or when a new archive source is loaded. When the stream is paused, information is not being passed to AXClient so the start and stop time updates will not trigger.

Parameters:

callbackFunction - User defined function name

Event Callback API:

```
void callbackFunction(string name, Date stoptime);
```

This user defined function will be called when the archive stop/end time has changed.

Callback Parameters:

name - AXClient name which processed the save clip command
starttime - Date representation of the new archive stop/end time

See Also:

```
getStartTime()
getStopTime()
setOnStartTimeChanged()
```

JavaScript Code Example:

```
<script type="text/javascript">
function onStopTimeChanged(name, stoptime) {
stopDate = new Date(stoptime); // recast
// perform action
}
AXC.setOnStopTimeChanged(onStopTimeChanged);
</script>
```

setPlayrate

```
public void setPlayrate(double rate)
```

Versions Supported: 4.0+

Sets the playrate for a JPEG archive or JPEG archive file (.bwm). Does not work for MPEG archive sources.

This method has been superseded by the setPlayrateEx() method.

Completion of this method will invoke the callback function set via setOnPlayrateChanged().

Parameters:

rate - Frames per second for a JPEG archive

See Also:

```
getPlayrate()
getPlayrateEx()
setPlayrateEx()
setOnPlayrateChanged()
```

setPlayrateEx

```
public void setPlayrate(double rate)
```

Versions Supported: 4.4+

Sets the playrate for a JPEG archive or JPEG archive file (.bwm). Does not work for MPEG archive sources.

This method replaces the setPlayrate() method.

Completion of this method will invoke the callback function set via setOnPlayrateChanged().

Parameters:

rate - Frames per second for a JPEG archive

See Also:

```
getPlayrate()
getPlayrateEx()
setPlayrateEx()
setOnPlayrateChanged()
```

setSeekTime

```
public void setSeekTime(date time)
```

Versions Supported: 3.2+

Specify the position in the archive that the method seek() will jump to. Seeking should not be performed until the onStartOfStream callback has been received.

This method has been superseded by the seekToTime() method.

Parameters:

time - Date object representation of the time to seek an archive to

See Also:

```
seek()
seekToTime()
setSeekTimeByPercentage()
setOnStartOfStream()
```

JavaScript Code Example:

```
<script type="text/javascript">
myDate = new Date().getVarDate();
AXC.setSeekTime(myDate);
AXC.seek();
</script>
```

setSeekTimeByPercentage

```
public void setSeekTime(string factor)
```

Versions Supported: 3.2+

Specifies the seek position in the archive that the method seek() will jump to. The archive will jump to by the percentage (factor * 100) of the total length of the archive. For example, 0.5 will set the seek time to be 50% of the archive. The factor must be between 0.0 and 1.0.

This method has been superseded by the seekToPercentage() method.

Parameters:

factor - Date object representation of the time to seek an archive to

See Also:

```
seek()
seekToPercentage()
setSeekTime()
setOnStartOfStream()
```

JavaScript Code Example:

```
<script type="text/javascript">
AXC.setSeekTimeByPercentage("0.5"); // seek to 50%
AXC.seek();
</script>
```

setSkipFrames

```
public void setSkipFrames(long skipframes)
```

Versions Supported: 4.3+

Set the number of frames the IMC should skip during JPEG archive playback. Skip Frames equals 1 means the IMC will play every other frame.

This method has been superseded by the seekToPercentage() method.

Parameters:

skipframes - Integer value of the number of frames to skip.

See Also:

```
getSkipFrames()
```

setVmrDisplayMode

```
public void setVmrDisplayMode
```

Versions Supported: 5.1

Parameters:

SHORT displayMode - [0] or [1]

C++ Sample Code:

```
setVmrDisplayMode(0);
setVmrDisplayMode(1);
```

JavaScript Sample Code:

```
<script type="text/javascript">
var params1 = new Array();
params1['EnableDvrMode'] = 1;
params1['EnableVMRMode'] = 1;
params1['showTimestamp'] = 0;
new IMC('IMC1', '100%', '100%', null, params1);
IMC1.setOnMouseButtonDown('mouseButtonDown');
IMC1.setOnMouseMove('mouseMove');
IMC1.setOnMouseButtonUp('mouseButtonUp');
IMC1.setOnMouseWheelRotated('mouseWheelRotated');
```

```
IMC1.setOnStateChanged('onStateChanged');
IMC1.setOnStartOfStream('onStartOfStream');
IMC1.setOnStopTimeChanged('onStopTimeChanged');
IMC1.setVmrDisplayMode(1);
params1 = null;
</script>
```

showTimestamp

```
public void showTimestamp(boolean show)
```

Versions Supported: 4.4+

Show or hide the timestamp overlay when playing an MPEG-2 source. Timestamps are OFF by default.

Parameters:

show - Boolean: TRUE to turn on timestamps, FALSE to turn off.

snapshot

```
public variant snapshot()
```

Versions Supported: 4.3+

Save the current frame of video to the viewing client computer. Opens a windows dialog box where users can choose to save in a number of image formats, including bmp, gif, jpg and png.

stepForward

```
public void stepForward()
```

Versions Supported: 4.3+

Instruct the media to play forward one frame. Operates in the same direction to the current play direction. Thus issuing a playRewind() then a stepForward() will actually step reverse. Only valid for JPEG archive sources.

See Also:

```
playForward()
playRewind()
stepRewind()
```

stepRewind

```
public void stepRewind()
```

Versions Supported: 4.3+

Instruct the media to rewind one frame. Operates in reverse to the current play direction. Thus issuing a playRewind() then a stepRewind() will actually step forward. Only valid for JPEG archive sources.

See Also:

```
playForward()
playRewind()
stepForward()
```

stop

```
public void stop()
```

Versions Supported: 4.6+

Stops streaming the feed or archive. The video data buffer is flushed, however, properties are not unloaded, and get() methods will still return valid information for the stream that is loaded.

switchTo

```
public void switchTo(String mediaSourceURL)
```

Versions Supported: 3.2+

Play the specified source

Parameters:

mediaSourceURL - VSMS protocol in the form of //host/source.

Note The mediaSourceURL has the following optional attribute: framerate - Range: [0|0.001-30](default=5) Maximum number of frames per seconds transmitted to view proxy.

Code Examples:

```
AXC.switchTo("//host/source");
AXC.switchTo("//host/source?framerate=10");
```

CHAPTER 6

Interactive Media Clients

Interactive Media Client (IMC) is a group of ActiveX Controls that display video and control cameras. Multiple controls comprise IMC as follows:

AXClient

IMC features an ActiveX Control called AXClient. The AXClient plays live and archived JPEG, MPEG, and audio sources via a single control. A single AXClient renders video in a single video panel. By including multiple client objects in a page, multiple sources can be played simultaneously. With the IMC Public Methods API, it is possible to perform many features via the AXClient. The following features are available through the Public Methods API:

- Switch between multiple media sources in any format. For example, users can switch between live and archive JPEG, MPEG and audio sources.
- Dynamically define play lists using client-side scripts; including playing back multiple archives successively.
- Control archive playback with play, pause, stop, and seek.

Note VSMS will disconnect if the archive stream is paused more than 10 minutes.

- Extract video clips from an archive and save these clips in a variety of formats. These formats are AVI, MP2, WMV, and other archive and bwm formats.
- Capture snapshots from live or archived sources.

Similar to the IMC 4.2 applet, the AXClient can be used in web pages to display live and archived video. Developers can build custom, Windows-based applications with standard application building packages such as Visual C++, Visual Basic, Visual C#, etc. The AXClient also exposes IMC Public Class Methods API called by client-side script that affects how AXClient works. Use the IMC public methods to control video streaming and playback.

Note See the AXClient API, page 5-1 chapter to learn how the AXClient has the changed Public Class Methods.

AXClient Tag

A sample display using the AXClient <OBJECT> tag via a web page is shown below. The parameters also display the minimum required tags to automatically view the video. A single AXClient renders video in a single video panel. By including multiple AXClients in a page, multiple sources can be played simultaneously. With the IMC Public Methods API, it is possible to perform various functions features using playback/playforward type methods via the AXClient.

```
/*
 * Script to play video automatically
 */
var Play_List = new Array; //use associated array to map to each client
//add to list here
Play_List["IMC1"] = "//vsms.cisco.com/Source_Name";

function loadPlayer(axc_name){
<script language="javascript" type="text/javascript">  if(document.applets[axc_name]) {
    var player = document.applets[axc_name];
    try { //throws exception for invalid sources
      player.switchTo(Play_List[axc_name]);
      player.playForward();
    } catch(ex) {
      //exception logic
    }
  }
}
</script>

<!-- Video Panel -->
<object border="0" width="352" height="240"
codebase="/bwt/sources/AXClient.cab#version=4,5,0,6"
classid="clsid:41293422-93FD-443C-B848-E07EDBF866C3"
id="IMC1" name="IMC1" viewastext="viewastext">
  <param name="name"            value="IMC1"/>
  <param name="timestamp"       value="0"/>
  <!-- call script defined above-->
  <param name="OnPlayerLoaded"  value="loadPlayer"/>
</object>
```

Table 6-1 General Parameters

Field	Description
`display`	Reserved values: [html | text | ssv](default=html)
`border`	Range:[0-1000+]Number of pixels to border video.
`width`	Range: `[1-1000+ | 1-100%]` Width of object in pixels. The AXClient can stretch and scale video as requirements dictate. The full width defined in the <OBJECT> tag is used to render video. It is also possible to define a percentage value instead of hard pixels. The percentage will take standard DOM containers (tables, frames, etc.) into account, but it is possible to render video to 100% of the screen.

Table 6-1 General Parameters (continued)

Field	Description
height	Range: [1-1000+ | 1-100%] Height of object in pixels. The AXClient can stretch and scale video as requirements dictate. The full height defined in the <OBJECT> tag is used to render video. It is also possible to define a percentage value instead of hard pixels. The percentage will take standard DOM containers (tables, frames, etc.) into account, but it is possible to render video to 100% of the screen.
name	Character class: [0-9 | A-Z | a-z | _] The name the AXClient will use to identify itself when firing event.
id	Character class: [0-9 | A-Z | a-z | _] A unique ID for this AXClient.
onplayerloaded	Character class: [A-Z | a-z | 0-9 | _] User defined script function name. Called when the AXClient loads and is passed name value. Use this call-back to initialize each instance of AXClient with other script call-backs and to automate actions.
timestamp	0 = no timestamp displayed 1 = timestamp displayed

AXClient with Slider Tag

The AXClient can be paired with other ActiveX controls for a feature-rich application. The VSMS host pages incorporate an ActiveX Slider Control as a scrollbar to simplify archive viewing. Script libraries, scroll_bar.js and api_methods.js are available with the pages that can be incorporated into custom developed applications.

```
<OBJECT
    ID="<unique ID>"
    name="<unique Name>"
    width="<in pixels>"
    height="<in pixels>"
    classID="clsid:41293422-93FD-443C-B848-E07EDBF866C3">
    <param name="Min"           value="1">
    <param name="Max"           value="100">
    <param name="TickFrequency" value="0">
    <!-- other possible parameters
    //<param name="BorderStyle"   value="0">
    //<param name="MousePointer"  value="0">
    //<param name="Enabled"       value="1">
    //<param name="OLEDropMode"   value="0">
    //<param name="Orientation"   value="0">
    //<param name="LargeChange"   value="0">
    //<param name="SmallChange"   value="0">
    //<param name="SelectRange"   value="0">
    //<param name="SelStart"      value="0">
    //<param name="SelLength"     value="0">
    //<param name="TickStyle"     value="2">
    //<param name="TickFrequency" value="0">
```

```
    //<param name="Value"         value="0">
    //<param name="TextPosition"  value="0">
    //<param name="Captions"      value="0">
     -->
</OBJECT>
```

Use standardized naming conventions and include the following libraries in the web page:

```
<script language="javascript"
    src="/bWT/sources/scripts/api_methods.js"></script>
<script language="javascript"
    src="/bWT/sources/scripts/scroll_bar.js"></script>
```

Set the slider properties which are expressed as parameters using scripts. This permits properties to be updated dynamically.

```
<script language="javascript" type="text/javascript">
    //define parameters after OBJECT is initialized
    document.applets['scrollbar_IMC< axc_suffix >'].Min = 1;
    document.applets['scrollbar_IMC< axc_suffix >'].Max = 1000;
    document.applets['scrollbar_IMC< axc_suffix >'].TickFrequency = 10;
</script>
```

The following scripts capture DOM events for the specified Slider. Use < axc suffix > to match the AXClient instances with the appropriate Slider Control.

```
//by passing only AXClient name it simplifies DOM calls in scripts
<!-- a change is sent at the end of a drag -->
<script for="scrollbar_IMC< axc suffix >"
   event="Change()">
   onChangeScrollbar('IMC< axc suffix >');
</script>
<script for="scrollbar_IMC< axc suffix >"
   event="MouseDown(mouse_button,shift_key,x,y)">
   onMouseDownScrollbar('IMC< axc suffix >',mouse_button,shift_key,x,y);
</script>
<script for="scrollbar_IMC< axc suffix >"
   event="MouseUp(mouse_button,shift_key,x,y)">
   onMouseUpScrollbar('IMC< axc suffix >',mouse_button,shift_key,x,y);
</script>
<!-- scroll messages are sent while dragging -->
<script for="scrollbar_IMC< axc suffix >"
   event="Scroll()">
   onMoveScrollbar('IMC< axc suffix >');
</script>
```

AXClient with DHTML Timestamps

The AXClient does not include an attached timestamp area. An MPEG-2 feeds overlay is the only timestamp the AXClient includes. The BWT pages use a simple script to display the current frame timestamp. It is also possible to adjust the interval that a custom script uses to update the timestamp displayed. This same interval can also be used to adjust the position displayed by the associated Slider control.

```
<script language="javascript" type="text/javascript">
form.elements['TIMESTAMP_' + theIMC].value =
    formatUTCToString(current_utc);
</script>
```

ActiveX Camera (PTZ) Control

The AX Camera Controls permits users to send camera control directions to a PTZ camera. AX Camera Controls are objects that required by the application program.

Camera Controls

```
<script language="javascript" type="text/javascript">
//
// Instantiate the AX Camera Controls Object.
//
Camera_Control = new ActiveXObject("axcamcontrol.CiscoCameraControl");
//
// Set the object members.
//
Camera_Control.cameraType      = "<camera model type>"
Camera_Control.source          = "<camera source>"
Camera_Control.comPort         = "<comport>"
Camera_Control.sourcetype      = "<srctype>"
Camera_Control.controlprotocol = "<protocol>"
Camera_Control.controlpriority = "<priority>"
Camera_Control.chainNumber     = "<daisy chain number>"
//
// Pan-Tilt-Zoom the camera using setRelativePosition member function.
//
Camera_Control.setRelativePosition(<PTZcommand>,<speed>,<amplitude>);
//
// Pan-Tilt-Zoom to a preset using swithToPreset member function.
//
Camera_Control.switchRelativePosition(<presetNumber>);
</script>
```

Table 6-2 Required Members

Field	Description
source	Format: [name@address:port] name (required): The camera input number. address (optional): IP address/hostname.domain.extension of the IP Device the camera is connected to. port (optional): The port default is 80.
sourcetype	Reserved values: Type of device control Pan-Tilt-Zoom cameras.
comport	Reserved values: [COM1 | COM2] This specifies the COM port of the video encoder used for Pan-Tilt-Zoom and other camera controls. **Note** A value must be provided even though it may not be applicable.

Table 6-2 Required Members (continued)

Field	Description
`cameratype`	Reserved values. The camera brand and model used for Pan-Tilt-Zoom and other camera controls. **Note** <device>_driver_v1 uses the <device> 1.0 HTTP API. <device>_driver_v2 uses the <device> 2.0 HTTP API.
`controlprotocol`	Reserved values: [P | D] This specifies the camera protocol. **Note** A value must be provided even though it may not be applicable.
`controlpriority`	Range: [1-100](100) Camera control priority policy implemented to handle contention between multiple Pan-Tilt-Zoom commands to a single camera, where 100 is the highest possible priority. The user with the highest priority gets exclusive use of the camera for CAMERA_CONTROL_PRIORITY seconds, as defined on the VSMC Console. Any concurrent requests with a lower priority will be rejected during this time interval.
`chainnumber`	Range: [0-64](0) The address of the target Pan-Tilt-Zoom camera in the case of multiple addressable PTZ cameras connected in a daisy chain to the same serial port. **Note** A value must be provided even though it may not be applicable.
`presets`	Range: [-100 to 100] Used to start "continuous" camera movement. The camera will continue to move and/or zoom until a subsequent command is issued (unless an ms parameter was supplied with the URL command). • For pan, negative values indicate pan left and positive values indicate pan right. • For tilt, negative values indicate tilt down and positive values indicate tilt up. • For zoom, negative values indicate zoom out and positive values indicate zoom in. • For all three <device>, 0 means stop (so "command=_0,0,0" will stop all camera PTZ movement).

LoopBack

The LoopBack property controls archive behavior upon reaching its end. The archive will loop back to the beginning if the property if set to true or pause upon reaching its end. The default value for this property is false.

JavaScript

```
LoopBack
```

Sample Code

```
<script language=javascript>
<!--
function test()
{
IMC.LoopBack = true;
IsLoopBackOn = IMC.LoopBack;
}
//-->
</script>
<html>
<body onload="test()">
<OBJECT height=300 width=400 classid="CLSID:41293422-93FD-443C-B848-E07EDBF866C3"
viewastext id="IMC" name="IMC"> <PARAM name="LoopBack" VALUE="true"/>
</OBJECT>
</body>
</html>
```

C#

```
bool LoopBack [ get, set ]
```

Sample Code

```
bool fLoopBack;
fLoopBack = IMC.LoopBack;
IMC.LoopBack = true;
```

C++

```
HRESULT get_LoopBack(VARIANT_BOOL* pVal);
HRESULT put_LoopBack(VARIANT_BOOL newVal);
```

Sample Code

```
VARIANT_BOOL fLoopBack;
pIMC->get_LoopBack( fLoopBack);
pIMC->put_LoopBack( VARIANT_TRUE);
```

AsynchronousMode

The AXClient API methods can be invoked either synchronously or asynchronously. By default, all invocations are synchronous. An AXClient host application has option to invoke certain methods asynchronously by setting the property AsynchronousMode to true. Note that not all methods can be invoked asynchronously. All methods for retrieving information e.g. getRecordrate, getPlayrate etc. are synchronous in both modes.

The following methods can be executed asynchronously:

- pause
- mute
- stepForward
- stepRewind
- playForward
- playRewind
- setSeekTime
- setSeekTimeByPercentage
- seek
- SwitchToEx
- stop
- setSkipFrames
- playResume
- seekToTime
- seekToPercentage
- put_Timestamp
- close
- setPlayrateEx
- attach
- put_LoopBack
- SetACK(BOOL value) The default value is false. The AXClient will throw an exception if the server detects an error.

JavaScript

```
AsynchronousMode
```

Sample Code

```
<script language="javascript">
<!--
function test()
{
IMC.AsynchronousMode = true;
IsAsynchronousModeOn = IMC.AsynchronousMode;
}
//-->
</script>
```

```
<html>
<body onload="test()">
<object height="300" width="400" classid="CLSID:41293422-93FD-443C-B848-E07EDBF866C3"
viewastext id="IMC" name="IMC">
<param name="AsynchronousMode" value="true"/>
</object>
</body>
</html>
```

C#

```
bool AsynchronousMode [ get, set ]
```

Sample Code

```
bool fAsynchronousMode;
fAsynchronousMode = IMC.AsynchronousMode;
IMC.AsynchronousMode = true;
```

C++

```
HRESULT get_AsynchronousMode(VARIANT_BOOL* pVal);
HRESULT put_AsynchronousMode(VARIANT_BOOL newVal);
```

Sample Code

```
VARIANT_BOOL fAsynchronousMode;
pIMC->get_AsynchronousMode(fAsynchronousMode);
pIMC->put_AsynchronousMode( VARIANT_TRUE);
```

ActiveX Joystick Control

The AX Joystick Controls permits users to use joystick camera controls for a PTZ camera. AX Joystick Controls are objects required by the application program. AX Joystick Controls must be used with the AX Camera Controls as the AX Camera Controls describe the necessary PTZ camera information.

ActiveX Joystick Object

```
<script language="javascript" type="text/javascript">
//
// Instantiate the camera control object.
// Need the object members to define the vsms host and camera related data
//
Camera_Control = new ActiveXObject("axcamcontrol.CiscoCameraControl");
//
// Set the camera control object members.
//
Camera_Control.sourceHost      = "<VSMS host>"
Camera_Control.cameraType      = "<camera model type>"
Camera_Control.source          = "<camera source>"
Camera_Control.comPort         = "<comport>"
Camera_Control.sourcetype      = "<srctype>"
Camera_Control.controlprotocol = "<protocol>"
Camera_Control.controlpriority = "<priority>"
```

```
Camera_Control.chainNumber     = "<number>"
//
// Instantiate the joystick controls object.
//
Joystick_Control = new ActiveXObject("axcamcontrol. CiscoJoystickControl");
//
// These object members are auto detected and initialized
// when joystick control object is instantiated.
//
Joystick_Control.axes
Joystick_Control.buttons
Joystick_Control.povs
Joystick_Control.sliders
//
// Set the joystick control object members.
//
Joystick_Control.pollingInterval = <milliseconds>;
Joystick_Control.cameraControl = Camera_Control; //defined above
//
// Map an <device> using map<device> member function.
// Each <device> needs to mapped with map<device>()
//
Joystick_Control.map<device>(<device>Number, <device>Command);
//
// Map a button using mapButton member function.
// Each button needs to be mapped with mapButton()
//
Joystick_Control.mapButton(buttonNumber, buttonCommand);
//
// Map a POV using mapPov member function.
//
Joystick_Control.mapPov(povNumber, povCommand);
//
// Map a slider using mapSlider member function
//
Joystick_Control.mapSlider(sliderNumber, sliderCommand);
</script>
```

Table 6-3 Required Fields

Field	Description
axes	Format: [integer] Number of axes detected when object is instantiated.
buttons	Format: [integer] Number of buttons detected when object is instantiated.
povs	Format: [integer] Number of POVs detected when object is instantiated.
sliders	Format: [integer] Number of sliders detected when object is instantiated.
pollinginterval	Format: [integer milliseconds](default=200) Polling time interval to check if the joystick has moved.
cameracontrol	Camera control PTZ object.

Method Descriptions

setRelativePosition

```
setRelativePosition(int PTZcommand, int speed, int amplitude | zoom)
```

Send a command to move PTZ camera.

Table 6-4 General Parameters

Field	Description
PTZcommand	Range [0-9] Pan-Tilt-Zoom command. 0=zoom out; 1=left down; 2=down; 3=down right; 4=left; 5=zoom in; 6=right; 7=up left; 8=up; 9=up right **Note** PTZcommand values map to directions implied by matching values from a keyboard 10-key. For example, 7 is up left.
speed	Range [1-100] Camera movement speed, where 100 is the fastest.
amplitude | zoom	Range [1-360 | 1-100] Distance for pan and tilt movements where 1 is the smallest value and 360 is the largest value. Or amount a Pan-Tilt-Zoom camera zooms where 1 is the smallest amount and 100 is the largest.
enable	enable(bool enable) Permits the joystick command to be sent to the camera.

Example

```
//
// The following function causes a Pan-Tilt-Zoom camera,
// to move slightly to the left as fast as possible.
//
Camera_Control.setRelativePosition(4,100,5);
```

switchToPreset

```
switchToPreset(int presetNumber)
```

Send a command to move to the specified Preset Position stored in the camera.

Parameters:

presetNumber - the number of the preset stored in the camera

Example

```
//
// The following function causes a Pan-Tilt-Zoom camera,
// to move to preset 10.
```

```
//
Camera_Control.switchToPreset(10);
```

Shared Semi-colon Separated Value (SSV) Commands for Mapping Joystick Controls

For <SSV command>, combine options from a single line to configure the behavior of a given joystick control--<device>, button, POV, or slider.

```
continuous pan | tilt | zoom; (base options); relative pan | tilt | zoom; (base options);
[always enabled=true | false;] | [enabled=true | false;] switch to preset; preset=<number>
send button; button=<camera.bwt "button" parameter> send url; url=<fully qualified URL>
enable relative movement (base options): scale=<decimal percentage like 1.0 - 0.01 >;
speed=<decimal percentage like 1.0 - 0.01 >; invert=true | false;
```

clearMap

clearMap()

Reset the mappings for a Joystick_Control. It is possible to map multiple Camera_Control to a single Joystick_Control to permit multiple cameras to be moved simultaneously via a single USB joystick. To avoid overlapping controls, clear the mappings before setting a new one.

getMode

getMode()

Returns a string displaying the currently connected joystick.

- If empty string—no joystick is attached
- If "Plug and Play Joystick (USB)"—for a DirectX joystick

map<device>

```
map<device>(int <<device> number>, string <<device> SSV command>)
```

Map a joystick <device> to a command.

Parameters:

<device> number - The number corresponding to a joystick <device>. Depending on the type of joystick, typically 1/4 = x <device>, 2/5 = y <device>, 3/6 = z <device>.

Note Some joysticks map <device> to translational (Cartesian x-y-z to 1-2-3) or rotational (roll, pitch, yaw to 4-5-6) or a combination of these. Some typical examples are 1-2-3, 1-2-6, and 1-2.

<device> SSV command - <device> command string. The string command is made up of a command and optional parameters separated by semicolons.

Note The `speed` parameter is no longer used. Movement speed is translated proportionally from joystick displacement.

mapButton

```
mapButton(int <button number>, string <button SSV command>))
```

Map button to a command.

Parameters:

button number - The number corresponding to a button on a joystick.

button SSV command - The button command string. The string command is made up of a command and optional parameters, separated by semicolons.

mapPov

```
mapPov(int <POV number>, string <POV SSV command>))
```

Map POV to a command.

Parameters:

POV number - The number corresponding to a POV button on a joystick. POV SSV command - POV command string. The string command is made up of a command and optional parameters, separated by semicolons.

mapSlider

```
mapSlider(int <slider number>, string <slider SSV command>))
```

Map slider to a command.

Parameters:

slider number - The number corresponding to a slider on a joystick.

slider SSV command - The slider command string. The string command is made up of a command and optional parameters, separated by semicolons.

Example

```
if( Joystick_Control.axes != null && Joystick_Control.axes > 0 ) {
    // reset/clear settings first
    // now more than a single camera can be mapped to
    // one joystick.
    Joystick_Control.clearMap();
    //
    // Map the X-<device> to continuous pan
    //
    var x_<device>_cmd= "continuous pan;scale=0.90";
    try { /// X ///
        Joystick_Control.map<device>(1, x_<device>_cmd);
    } catch(ex) {
        //may be translated or twist control; 4.4
        Joystick_Control.map<device>(4, x_<device>_cmd);
    }
    //
    // Map the Y-<device> to continuous tilt
    //
    var y_<device>_cmd= "continuous tilt;scale=0.90";
    try { /// Y ///
        Joystick_Control.map<device>(2, y_<device>_cmd);
    } catch(ex) {
```

```
    //may be translated or twist control; 4.4
    Joystick_Control.map<device>(5, y_<device>_cmd);
}
//
// Map the Z-<device> to continuous zoom
//
if( Joystick_Control.axes > 2 ) {
    var z_<device>_cmd= "continuous zoom;scale=0.90;invert=true";
    try { /// Z ///
        Joystick_Control.map<device>(3, z_<device>_cmd);
    } catch(ex) {
        //may be translated or twist control; 4.4
        Joystick_Control.map<device>(6, z_<device>_cmd);
    }
}
//
// The following function maps button 1 to continuous pan.
//
Joystick_Control.mapButton(1,"continuous pan;scale=0.90;speed=0.55");
//
// The following function maps POV to a relative position.
//
Joystick_Control.mapPov(1,"relative pan;scale=0.77;speed=0.55");
//
// The following function maps the slider to a continuous zoom command.
//
Joystick_Control.mapSlider(1,"continuous zoom;scale=1.0;speed=0.30");
    }
```

CHAPTER 7

Camera Control API

VSMS supports the following two mechanisms for controlling pan-tilt-zoom (PTZ) -capable cameras via URL-based commands:

- camera.bwt (camera control) module—This module supports a standard set of high-level camera control commands. It works by translating these high-level commands into low-level commands specific to particular models of cameras and forwarding them to the encoders in which they are attached. In most cases, this is the module to use for controlling cameras.
- devcontrol.bwt (device control) module—This module provides a "pass through" function that forwards commands to encoders without interpreting them. It is primarily useful for tweaking encoder parameters in ways not supported by VSMS and when a direct path between the client and encoder is not available.

Camera Control Module (camera.bwt)

The purpose of the VSMS camera control module is to permit clients to configure and control different types of cameras via the network without having to know the low-level control protocols for those cameras. It is designed to support two common configurations:

1. One or more analog cameras connected to a network video server (encoder). Each camera has a video output and serial ports. The video output for each camera is attached to the encoder. The encoder digitizes and encodes the video feeds received from the cameras and transmits those encoded feeds over the network to VSMS proxy servers.

   The camera serial ports are hooked up in a daisy chain fashion, with each camera in the chain assigned a unique "chain number". The whole chain is then plugged into a serial port on the encoder. Device-specific camera control commands (which contain the chain number) are received by the encoder from VSMS and then sent to the camera chain via the encoder's serial port. Commands are passed along the chain until they reach the identified camera. These digital network cameras should support all the functions of a camera and an encoder in a single device.
2. An IP camera with integrated video, PTZ, and controls.

Camera control flows only from client to camera as discussed below:

- The client sends the URL commands to VSMS. The commands are then passed to the camera control module.
- The camera control module interprets and translates the URL commands into camera-specific, low-level commands and sends them to the encoder.
- The encoder forwards the commands it receives to the applicable camera.

- The encoder responds back to VSMS with a simple "acknowledge" method (the actual method depends upon the type of encoder).
- VSMS will send an HTTP 204 "No Content" response to the client if ack is no, indicating that it received and processed the URL command (HTTP 204 permits URL commands to be sent from a browser without forcing that browser to reload the page after every command).

Note If ack is yes, then an HTTP 200 code will be returned indicating the command was successfully executed.

Basic camera control commands (URLs) forms are as follows:

```
http://<host>/camera.bwt?<device parameters><operation parameters><session parameters>
```

Where:

- <host>—The fully qualified host name (with domain) or an IP address of the camera control server, followed by an optional ":port". If not supplied, the port number will default to 80 (this is the default port for VSMS). Examples are shown below: myhost.mycompany.com
  myhost.mycompany.com:8080
  123.45.67.89
  123.45.67.89:8000
- <device parameters> —used to communicate information about the camera and encoder. See Device Parameters, page 7-2 for additional information.
- <operation parameters>—used to identify the specific operations the camera is to execute. See Operation Parameters, page 7-4 for additional information.
- <session parameters>—used to change how the command is handled by the VSMS server. See Session Parameters, page 7-9 for additional information.

Device Parameters

The core functions of the camera control module are supported internally by two types of "drivers":

- camera drivers—These are responsible for converting the standardized high-level URL commands supported by the module into the (usually binary) control protocol understood by a particular model of camera.
- encoder drivers—These are responsible for sending the output of a camera driver to the encoder to which the camera is attached (via the network protocol supported by a particular type of encoder).

For the server to successfully translate and forward a camera control command, the following is required:

- The encoder driver to be used for communicating with the encoder.
- The IP address of that encoder.
- The user name and password if authentication is required for that encoder.

Note Authentication parameters are not provided via camera.bwt URL commands. Instead, the server obtains this information by reading the appropriate "run" file. Whenever a proxy to an encoder (direct proxy) is started on a VSMS host, it writes the authentication information for that encoder in a "run" file whose name is the IP address of the encoder. Therefore, before a VSMS host can successfully operate as a camera control server for a particular encoder, users must:

- Start a proxy on the VSMS host for that encoder, or
- Transfer a copy of the "run" file for that encoder from another host on which such a proxy is running. Run files are located in /usr/BWhttpd/root/run.
- The camera driver used for converting the URL commands into the low-level control protocol.
- If the camera is attached to the encoder via a serial port, the port number that it is attached to and its chain number.
- If the camera is not attached to the encoder via a serial port (e.g., network camera), the numeric identifier the encoder uses to identify that camera.

This parameter serves two purposes:

1. Instructs the server the IP address of the encoder. This is used both for establishing network communication with the encoder and for finding the "run" file that contains the encoder's authentication data.
2. If the encoder uses a numeric id to identify a camera (instead of serial port and chain number), num is used.

Command

```
http://<host>/camera.bwt?model=<[&srctype=<[Supported
Devices]>&source=<name@address:port>&number=<0-64>&speed=<1-100>&command=<ptz
command>&button=<preset command>&protocol=<[P | D]>&priority=<1-100>&ms=<20-20000000>
```

Table 7-1 Device Parameters

Parameter	Description
srctype	Reserved Values:[Supported Devices] This identifies the type of encoder which the module uses to select the appropriate encoder driver. Provides the same value as setting up a proxy for the encoder. Required unless the proxy is provided.
source	Format:[id@ip_address]
model	Reserved Values:[Supported Devices] This identifies the camera model which the module uses to select the appropriate encoder driver. **Note** For network cameras, the model will usually be the same as the srctype. – <device>_driver_v1 uses <device> 1.0 HTTP API. <device>_driver_v2 uses <device> 2.0 HTTP API. Required unless proxy is provided.
protocol	Reserved Values:[P | D] This identifies the particular variant of camera control protocol for VSMS to employ.

Table 7-1 Device Parameters (continued)

Parameter	Description
`comport`	Reserved Values:[COM1 | COM2] This parameter identifies the encoder's serial port to which the camera is connected as applicable. Required for analog cameras unless a proxy is provided.
`number`	Range:[0 - 64] This parameter identifies the chain number for the camera as applicable. Required for analog cameras unless a proxy is provided.
`proxy`	Format:[name] Rather than sending all the device parameters individually in the camera.bwt URL command, users can add "named" configurations to a ptz.conf configuration file on VSMS and reference these configurations via a single "proxy" name. See Creating PTZ Configurations, page 7-10 for additional information.

Operation Parameters

The following parameters are used to control camera operations:

Table 7-2 Operation Parameters

Parameter	Description
`command`	Format:[<c><operands>(;<c><operands>;)] where: • <c>— a single letter that identifies the command to be executed. • <operands>—the operands appropriate for that command. Required unless the button is provided.
`button`	Format:[<name>] where <name> is the name of a macro defined for that camera. This field is optional.

Table 7-2 Operation Parameters (continued)

Parameter	Description
speed	Range:[1 - 100] This parameter is used to set the pan and/or tilt speed for "momentary" PTZ movement commands. This field is optional.
ms	Range:[20 - 20000000] This parameter is used to transform a "continuous" PTZ movement command into a "momentary" PTZ movement command. This is accomplished by having the server automatically issue a "stop movement" command to the camera an X number of milliseconds after the movement command is sent. This field is optional.

Configuration Operations

Configuration operations affect the configuration of a camera rather than its current position and are described in the following table:

Table 7-3 Configuration Parameters

Command	Parameter	Description
*	(command)	Format:[<operand>] This command operates as "pass through". Each camera driver supports it, but the interpretation of the operand differs from driver to driver. To display the syntax for a particular model, find the appropriate driver entry in the camera.prof file.
backlight_off	(button)	Disable backlight compensation
backlight_on	(button)	Enable backlight compensation
dzoom_off	(button)	Disable digital zoom
dzoom_on	(button)	Enable digital zoom
focus_auto	(button)	Auto focus
focus_manual	(button)	Manual focus
iris_auto	(button)	Auto iris
iris_manual	(button)	Manual iris
night_auto	(button)	Auto night mode
night_off	(button)	Turn off night mode
night_on	(button)	Turn on night mode
init	(button)	Initialize the default settings for PTZ control
reset	(button)	Reset/restart the settings for PTZ control
wb_auto	(button)	Auto white balance

Table 7-3 Configuration Parameters (continued)

Command	Parameter	Description
wb_indoor	(button)	Indoor white balance
wb_outdoor	(button)	Outdoor white balance
wb_manual	(button)	Manual white balance

Focus Operations

Focus operations affect the current focus state of the camera and are described in the following table:

Table 7-4 Focus Parameters

Command	Parameter	Description
F	(button)	Range:[0 - 9] This command shifts focus of the camera farther away (0 = short distance, 9 = long distance).
R	(command)	Range: [0 – 9] This command shifts focus of the camera closer in (0 = short distance, 9 = long distance).
far	(button)	Focus a little farther
near	(button)	Focus a little nearer

Iris Operations

Iris operations affect the current iris state of the camera and are described in the following table:

Table 7-5 Iris Parameters

Command	Parameter	Description
D	(command)	Range:[0 - 9]This command closes (dims) the camera iris (0 = small amount, 9 = large amount).
E	(command)	Range:[0 - 9]This command opens (brightens) the camera iris (0 = small amount, 9 = large amount).
bright	(button)	Brighten a little
dim	(button)	Dim a little

PTZ Operations

PTZ operations are those that involve panning (left/right), tilting (up/down), and zooming (in/out) the camera and are described in Table 7-6 on page 7-7:

Table 7-6 PTZ Parameters

Command	Parameter	Description
B	(command)	Range: [1 – 360] This "momentary" movement command moves the camera down and left a given relative distance (1 = short distance, 360 = long distance).
H	(command)	Range: [1 – 360] This "momentary" movement command moves the camera left a given relative distance (1 = short distance, 360 = long distance).
J	(command)	Range: [1 – 360] This "momentary" movement command moves the camera down a given relative distance (1 = short distance, 360 = long distance).
K	(command)	Range:[1 - 360] This "momentary" movement command moves the camera up a given relative distance (1 = short distance, 360 = long distance).
L	(command)	Range:[1 - 360] This "momentary" movement command moves the camera right a given relative distance (1 = short distance, 360 = long distance).
N	(command)	Range: [1 – 360] This "momentary" movement command moves the camera down and right a given relative distance (1 = short distance, 360 = long distance).
U	(command)	Range: [1 – 360] This "momentary" movement command moves the camera up and right a given relative distance (1 = short distance, 360 = long distance).
W	(command)	Range: [1 – 100] This "momentary" zoom command zooms the camera out a given relative distance (1 = short amount, 100 = long amount).
Y	(command)	Range: [1 – 360] This "momentary" movement command moves the camera up and left a given relative distance (1 = short distance, 360 = long distance).
Z	(command)	Range: [1 – 100] This "momentary" zoom command zooms the camera in a given relative distance (1 = short amount, 100 = long amount).

Table 7-6 PTZ Parameters (continued)

Command	Parameter	Description
_	(command)	Format:[<pan_speed>,<tilt_speed>,<zoom_speed>] command is used to start "continuous" camera movement. The camera will continue to move and/or zoom until a subsequent command is issued (unless an ms parameter was supplied with the URL command). Speed values range from -100 to 100: • For pan, negative values indicate pan left and positive values indicate pan right. • For tilt, negative values indicate tilt down and positive values indicate tilt up. • For zoom, negative values indicate zoom out and positive values indicate zoom in. • For all three <device>, 0 means stop (so "command=_0,0,0" will stop all camera PTZ movement).
down	(button)	Move down a little
downleft	(button)	Move down and left a little
downright	(button)	Move down and right a little
left	(button)	Move left a little
right	(button)	Move right a little
stop	(button)	Stop all PTZ movement
tele	(button)	Zoom in a little
up	(button)	Move up a little
upleft	(button)	Move up and left a little
upright	(button)	Move up and right a little
wide	(button)	Zoom out a little

Presets Operations

Presets operations involve the definitions of preset camera locations and movement to those positions. They are described in Table 7-7 on page 7-9:

Table 7-7 Preset Parameters

Command	Parameter	Description
D	(command)	Format: [<num>(<label>)] This command is used to assign a preset number (and an optional text label) to the current camera position. Preset numbering starts from 1 and most cameras support at least 10.
G	(command)	Format: [<num>] This command is used to move the camera to a previously defined preset number. Preset numbering starts from 1 and most cameras support at least 10.

Session Parameters

The following parameters affect how the camera control module processes a URL command:

Table 7-8 Session Parameters

Parameter	Description

ack	Reserved Values:[yes | no]
	This parameter controls how the status is returned to the client. If ack is no, then an HTTP 204 ("No Content") code will be returned. If ack is yes, then an HTTP 200 code will be returned and the body of the response will contain additional information in the following format:
	`<result>`
	`<code>#</code>`
	`<text>text of error if code is not 0</text>`
	`</result>`
	The default is no. This field is optional.

camera	Format:[http://<host>/camera.bwt] This parameter instructs the camera control module to forward this command to another VSMS server (as identified by the <host> parameter). This field is optional.
priority	Range:[1 - 100] This parameter assigns a priority for the current command. Each time a command is received, it compares the priority of the command to that of the last one. If the priority of the current command is lower than that of the previous command, it is rejected until a sufficient duration of time has passed since the previous command was executed. The default "exclusive access" is five minutes. This value can be modified in the PTZ Configuration section of the management console. This parameter is required unless a proxy is provided.

Creating PTZ Configurations

As discussed in Device Parameters, page 7-2, device parameters may be recorded for specific PTZ configurations. Users can reference them in commands via the proxy parameter. PTZ configurations are stored in /usr/BWhttpd/conf/ptz.conf. Each line in ptz.conf is formatted as follows:

```
proxy;presets;model;srctype;source;comport;number;speed;protocol;priority
```

Table 7-9 PTZ Configurations

Parameter	Description
proxy	The name of this configuration.
presets	List of camera presets. Use number/value pairs separated by commas. This field is not used by VSMS. Users should use the Interactive Media Transcoder (IMT) Client for defining the preset list.
model	The value for the model parameter described in Device Parameters.
srctype	The value for the srctype parameter described in Device Parameters.
source	The value for the source parameter described in Device Parameters.
comport	The value for the comport parameter described in Device Parameters.
number	The value for the number parameter described in Device Parameters.
speed	The default value for the speed parameter described in Operation Parameters.
protocol	The value for the protocol parameter described in Device Parameters.
priority	The value for the priority parameter described in Session Parameters.

A sample ptz.conf file appears as follows:

```
<device>;<device>;1@10.10.1.100;COM1;0;10;P;100
<device>;1=pre1,2=pre2,3=pre3;<device>;<device>;1@10.10.2.100;COM1;0;5;P;100
proxy<device>;1=hallway;2=window;3=door;<device>;1@10.10.2.110;COM2;0;25;P;100
```

Using camera.bwt

Using camera.bwt involves configuring various pieces of the camera control components, device parameters, operations parameters, and session parameters to formulate a camera control command.

The API command has two forms as specified in the camera control API and "named" proxy configuration.

Command with Required and Optional Parameters

```
camera.bwt?>&source=<name@address:port>&command=<ptz command>&model=<[ camera model
]>&number>=<0-64>&comport=<[ COM1 | COM2 ]>&speed=<value>&command=<ptz
command>&protocol=<[ P | D ]>&priority=<value>&ms=<value>
```

Command with proxy parameter where "proxy name" is defined in /usr/BWhttpd/conf/ptz.conf

```
http://<host>/camera.bwt?&proxy=<proxy name>&command=<ptz command>&button=<button command>
```

Example

The following example stops continuous movement for a <device>:

```
http://host/camera.bwt?command=_0,0,0&source=1@192.168.1.109&model=<device>&srctype=<devic
e>&comport=COM1
```

Example

The following example sends a pass through command to enable digital zoom for a <device> camera:

```
http://host/camera.bwt?command=*"120"650000001&source=1@192.168.1.112&model=<device>&srcty
pe=<device>&comport=COM1
```

Example

The following example sends n iris focus, using the button macro "near" command, to an <device> PTZ Network camera:

```
http://host/camera.bwt?button=near&source=1@192.168.1.125&model=<device>&srctype=<device>&
comport=COM1
```

Example

The following example sets preset 1 for a <device> camera:

```
http://host/camera.bwt?command=S1,Front_Door&source=1@192.168.1.100&model=<device>&srctype
=<device>&comport=COM1
```

Example

The following example goes to preset 1 for a <device> camera:

```
http://host/camera.bwt?command=G1,Front_Door&source=1@192.168.1.100&model=<device>&srctype
=<device>&comport=COM1
```

Example

The following example chains multiple PTZ commands into one API call that will go to preset 1, focus near, and brighten the iris:

```
http://host/camera.bwt?command=G1;R0;E5&source=1@192.168.1.100&model=<device>&srctype=<dev
ice>&comport=COM1
```

Example

The following example moves a <device> camera up:

```
http://host/camera.bwt?command=K10,Front_Door&source=1@192.168.1.100&model=<device>&srctyp
e=<device>&comport=COM1
```

Example

The following example moves a <device> camera down using the proxy name parameter:

```
http://host/camera.bwt?command=J50&proxy=Outside_<device>
```

The ptz.conf entry for Outside<device> is as follows:

```
Outside_<device>;;<device>;<device>;192.168.1.103;COM1;0;10;P;100
```

Device Control Module (devcontrol.bwt)

Users can bypass the VSMS camera.bwt module and control a device directly using the pass through mode module devcontrol.bwt.

Command

```
http://<host>/devcontrol.bwt?source=<[device hostname | IP address]>&comport=<[COM1 |
COM2]>&srctype=<[Supported Devices]>&datatype=<[ascii |
hex]>&username=<username>&password=<password>&data=<data string>
```

Table 7-10 Required Fields

Command	Description
host	Format: [hostname.domain | IP address] Web address of host where VSMS is running. **Note** VSMS runs on port 80 by default. Specify additional ports similar to vsms_host.cisco.com:8080.
source	Format: [address:port] address (required): IP address/hostname.domain.extension of the controlled device. port (optional): Displays the port number. The port default is 80.
comport	Reserved values: [COM1 | COM2] Specifies the COM port of the Pan-Tilt-Zoom controls that are connected to the controlling device (a supported srctype video server).
srctype	Reserved values. [Supported Devices] Specifies the type of video encoder to control the device.
data	Data string used to control the device.

Table 7-11 Optional Fields

Command	Description
`datatype`	Reserved Values: [ascii | hex] (hex) Format that the command being passed is expected to be in by the device receiving it.
`username`	Format: [user name] Administration user name for encoding device. **Note** If a username and password is enabled for a device, this parameter is required.
`password`	Format: [password] Administration password for encoding device. **Note** This parameter is required if a username and password is enabled for a device.

Example

The following command causes a device with protocol P, Pan-Tilt-Zoom camera, connected to a <device> to move right:

```
http://host/devcontrol.bwt?source=<device>&comport=COM1&srctype=<device>&data=a00000023f3f
af0d&datatype=hex
```

Example

The following command causes a device, with protocol P, Pan-Tilt-Zoom camera, connected to a <device> to stop movement:

```
http://host/devcontrol.bwt?source=<device>&comport=COM1&srctype=<device>&data=a00000000000
af0f&datatype=hex
```

CHAPTER 8

DVR Integration

This gateway can support up to10 associated DVRs.

Features:

- Integrated 3rd party DVR devices into the Video Surveillance Manager suite
- Access live and recorded media from 3rd party DVRs
- Control of PTZ cameras connected to the DVR

Media Flow

DVRs are considered devices providing media feeds into the Media Acquisition subsystem

Step 1 Acquisition from the devices via drivers.

Step 2 Normalized.

Step 3 Fed into the shared memory.

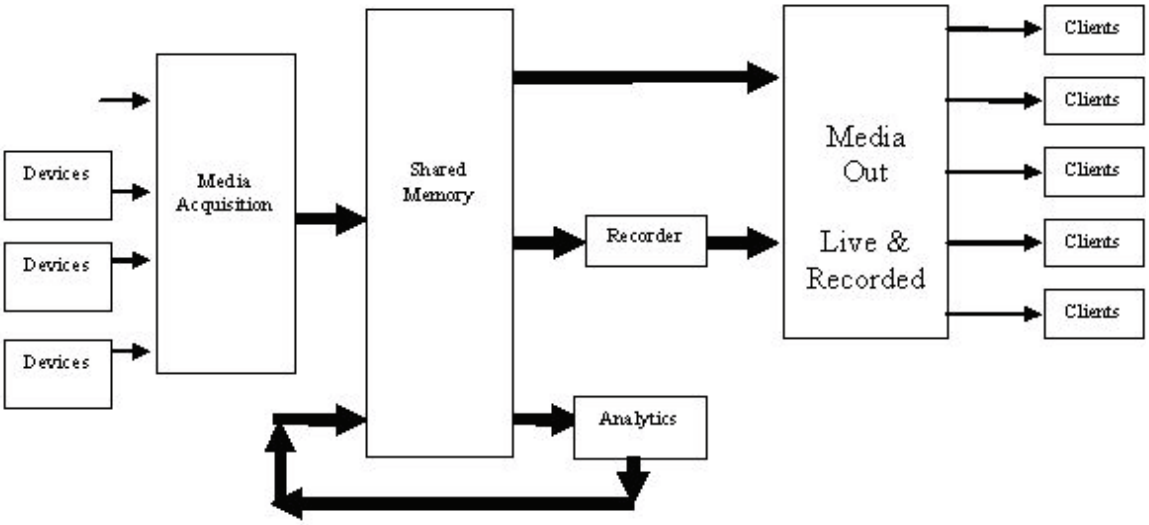
Step 4 The media is fanned to multiple consumers.

Step 5 Media is obtained by the recorder via the Capture interface.

Step 6 Media out repackages the media into multiple packets and the network transports to them to various clients.

Step 7 Analytics process the video and may modify the media stream and reinsert it into shared memory.

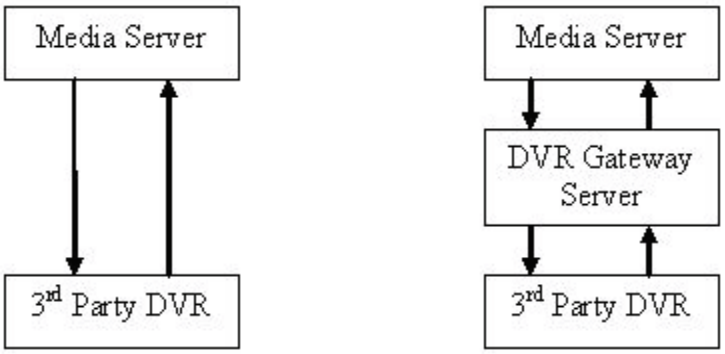
Figure 8-1 Media Flow



There are two variations of the DVR integration:

1. Direct access to video streams and control paths of the DVR when a network interface is present
2. When a Windows SDK is the only interface to the DVR, the DVR Gateway server will provide the network API for the Media Server to access the video streams and control paths of the DVR

Figure 8-2 DVR Integration Scenarios



The following APIs are used to contact VSMS through the current BWT commands and info calls. These are then passed to the DVR GateWay for retrieval through an encapsulated XML in a HTTP Post message.

ADD NVR

Command

```
http://<host>/command.bwt?command=start&type=nvr&name=my_nvr&source=10.10.55.153&srctype=nice_dvr&username=root&password=psbu
```

This command sends a message to the bwt_commander running as an Apache loadable module. The bwt_commander uses this information to create a .12 DVR configuration file for later use by the Discovery API. The information in the XML file will consist of the parameters passed in through the command.bwt URI by VSOM.

Note A new directory will be created named /usr/BWhttpd/conf/dvr to differentiate DVRs from Proxies. DVR objects are unique from DVR channels and contain only a subset of information.

REMOVE NVR

Command

```
http://<host>/command.bwt?command=remove&type=nvr&name=my_nvr
```

This command sends a message to the bwt_commander. The bwt_commander removes the .12 DVR Media configuration file and all associated channel configuration .10 files and shuts down the Stream connections so that these particular streams and NVRs are no longer viewed in VSOM.

LIST NVRs

Command

```
http://<host>/info.bwt?type=nvr&display=[html|text|ssv]
```

This command displays a list of NVRs.

START DISCOVERY

Command

```
http://<host>command.bwt?command=discover&type=nvr
```

This command initiates discovery to the DVR Gateway from VSOM. Once a message is sent to the BWT commander, it will be passed through the DVR Gateway to retrieve all the channels connected to the DVR. For each of these unique channels, there is a media configuration .10 file which will represent an active DVR channel. These channels are represented as Camera Feeds in VSOM and as Proxies in VSMS.

CHAPTER 9

Legacy Client Applets

Note Java applets will no longer be supported. It is highly recommended that developers do not use the following applets in new development.

Interactive Media Client (IMC) is a Java applet that plays live and archive JPEG, MPEG-2, and audio sources via a single client. It also supports media types from IMS 4.2 and earlier. IMC API features are as follows:

- switch between multiple media sources and formats (i.e. live and archive JPEG, MPEG-2 and audio sources, but not transcoded MPEG-4)
- continuously play back multiple archives using user-defined play lists
- pan-tilt-zoom supported cameras
- set and go to camera presets
- control archive playback with play, pause, and seek

Note VSMS will disconnect if the archive stream is paused more than 10 minutes.

- capture live or archive snapshots from the current source
- extract video clips from the current archive being played

The IMC can be customized to enhance a user's media experience. Multiple sources can be declared using NAME/VALUE pairs embedded in an IMC <APPLET> tag. Use the SourceList drop down menu of the IMC to switch between multiple sources or use the API methods. See Public Class Methods API section for additional information.

Note For XP clients unable to view full screen MPEG-2 video, run dxdiag, the DirectX Diagnostic Tool. Click on the Display tab. The DirectX Features: DirectDraw Acceleration, Direct3D Acceleration, and AGP Texture Acceleration must be set to Enabled.

Interactive Media Client

The parameter examples and descriptions in this module are the minimum required NAME/VALUE pairs to view videos. See the Code-block example sections for additional information.

Note This code sample includes a directly embedded defined SOURCELOADER VALUE and a set of source NAME/VALUE pairs. IMC will use a shared source handler declared in the SOURCELOADER VALUE to define sources or it must contain at least one complete set of source NAME/VALUE pairs to display video. See the code-block examples below to view how the IMC works.

```
<APPLET NAME="[IMC name]"  CODE="com.cisco.client.IMC"
    WIDTH="[width]" HEIGHT="[height]" CODEBASE="/java" MAYSCRIPT>
<PARAM NAME=CABBASE             VALUE="IMC.cab">
<PARAM NAME=ARCHIVE             VALUE="IMC.jar">
<PARAM NAME=AUTORUN             VALUE="[true | false]">
<PARAM NAME=SOURCELISTVISIBLE   VALUE="[true | false]">
<PARAM NAME=TIMESTAMPVISIBLE    VALUE="[true | false]">
<PARAM NAME=PLAYLIST            VALUE="[ID1;ID2;...;IDn]">
<PARAM NAME=LOOPINGBEHAVIOR     VALUE="[repeat | next | stop]">
<PARAM NAME=SCROLLBAR           VALUE="[ScrollBar name]">
<PARAM NAME=USEJOYSTICK         VALUE="[true | false]">
<PARAM NAME=HASJOYSTICKZ        VALUE="[true | false]">
<PARAM NAME=LOADINGTEXT         VALUE="[text displayed before video renders]">
<!-- define SOURCELOADER parameter or embed source NAME/VALUE pairs directly -->
<PARAM NAME=SOURCELOADER        VALUE="[SourceLoader name]">
<!-- source NAME/VALUE pairs embedded directly -->
<PARAM NAME=NUMBEROFSOURCES     VALUE="<n>">
<PARAM NAME=SOURCE[n]%ID        VALUE="[Unique Identifier]">
<PARAM NAME=SOURCE[n]%SOURCE    VALUE="[name@host:port]">
<PARAM NAME=SOURCE[n]%MEDIATYPE VALUE="[jpeg | mpeg2-v | mpeg4-v | audio]">
<PARAM NAME=SOURCE[n]%STORAGE   VALUE="[live | archive]">
<PARAM NAME=SOURCE[n]%LABEL     VALUE="[source description or label]">
<PARAM NAME=SOURCE[n]%AUTHORIZATION VALUE="[username,password]">
<PARAM NAME=SOURCE[n]%FRAMERATE VALUE="[frames per second]">
<PARAM NAME=SOURCE[n]%TIMEOUT   VALUE="[time out]">
<!-- AUDIO ONLY -->
<PARAM NAME=SOURCE[n]%AUDIOENCODING   VALUE="[mulaw | gsm | g726_40 | g726_24 | g726_16]">
<!-- PAN-TILT-ZOOM, MOTION DETECTION or DIRECT VIEW -->
<PARAM NAME=SOURCE[n]%DIRECTTOCAMERA        VALUE="[true | false]">
<PARAM NAME=SOURCE[n]%CAMERATYPE            VALUE="[<device>2130 | <device> | canonvcc3 |
cohu
   | cohu_1 | <device>2120 | <device>2420 | autodome]">
<PARAM NAME=SOURCE[n]%CAMERASOURCE          VALUE="[name@address:port]">
<PARAM NAME=SOURCE[n]%CAMERASOURCETYPE      VALUE="[ Supported Devices]  ">
<PARAM NAME=SOURCE[n]%CAMERAPORT            VALUE="[COM1 | COM2]">
<PARAM NAME=SOURCE[n]%CAMERACONTROLPROTOCOL VALUE="[P | D]">
<PARAM NAME=SOURCE[n]%CAMERACONTROLPRIORITY VALUE="[1-100]">
<PARAM NAME=SOURCE[n]%PRESETS               VALUE="[1=desc1;2=desc2;...;n=descn]">
<PARAM NAME=SOURCE[n]%CHAINNUMBER           VALUE="[n]">
</APPLET>
```

Table 9-1 Required Fields

Command	Description
NAME	Character class: [0-9 | A-Z | a-z | _] Name IMC will use to identify itself when firing events, as well as VALUE defined for SOURCELOADER parameter to handle external sources for other IMC applets. **Note** PLAYERNAME is no longer supported.
CODEBASE	Reserved value: [/java] Default directory applet is installed to /usr/bWhttpd/root/htdocs/java.
WIDTH	Range: [1-1000+ | 1-100%] Width of applet in pixels. IMC can stretch and scale video as requirements dictate. The full WIDTH defined in the <APPLET> tag is used to render video. It is also possible to define a percentage value instead of hard pixels. The percentage will take typical DOM containers (tables, frames, etc.) into account, but it is possible to render video to 100% of the screen. The SourceList and Timestamp each require 20 pixels of the WIDTH defined in the <APPLET> tag; the remaining amount is used to render video.
HEIGHT	Range: [1-1000+ | 1-100%] Height of applet in pixels. IMC can stretch and scale video as requirements dictate. The full HEIGHT defined in the <APPLET> tag is used to render video. It is also possible to define a percentage value instead of hard pixels. The percentage will take typical DOM containers such as tables and frames into account, but it is possible to render video to 100% of the screen. The SourceList and Timestamp each require 20 pixels of the HEIGHT defined in the <APPLET> tag; the remaining amount is used to render video.
CODE	Reserved value: [com.cisco.client.IMC]
ARCHIVE	Reserved value: [IMC.jar] Netscape Communicator JAR archive
CABBASE	Reserved value: [IMC.cab] Microsoft IE CAB archive

Table 9-2 Optional Fields

Command	Description
AUTORUN	Boolean values: [false | true](default=true) This specifies whether to stream media as soon as IMC loads or wait for the user to select a source. Sources can be selected using the SourceList drop down menu available for each IMC or the Public API switchTo(ID) method.
SOURCELISTVISIBLE	Boolean values: [false | true](true) This specifies whether the SourceList drop down menu is visible. The SourceList drop down menu displays the LABEL VALUE for the sources available to IMC and defined in the PLAYLIST parameter. If the LABEL parameter is not set for a given source in the PLAYLIST, IMC will populate the SourceList drop down menu with the source name, VSMS host, media type, and storage for that source similar to proxy@vsms_host. The SourceList requires 20 pixels of the HEIGHT defined in the <APPLET> tag. **Note** To view SOURCELISTVISIBLE with the parameter set to false, use the Public API switchTo(ID) method or set the AUTORUN parameter to true.
TIMESTAMPVISIBLE	Boolean values: [false | true](true) This specifies whether IMC displays Timestamp for the current archive. It is possible for the Timestamp to display the current time or the seek time of an archive. The Timestamp is blank for live sources. Also, the Timestamp will take up 20 pixels of the HEIGHT defined in the <APPLET> tag. See Public Class Methods API for additional information. **Note** The timestamp format displayed by the IMC is determined by the client machine's time format setting.
PLAYLIST	Format: [ID1;ID2;...;IDn] This specifies which SOURCE and LABEL will be displayed in the video panel and SourceList drop down menu. The values in the PLAYLIST are the same values specified in the source ID parameters, and should be separated by semi-colons.

Table 9-2 Optional Fields (continued)

Command	Description
LOOPINGBEHAVIOR	Reserved values: [repeat | next | stop](stop) When the current source in the PLAYLIST is an archive, specify what IMC will do when it reaches the end of the archive. • repeat Replay the current archive. • next Go to the next source in the PLAYLIST. • stop Do not replay the archive or go to the next source in the PLAYLIST.
SOURCELOADER	Character class: [0-9 | A-Z | a-z | _] Name of the applet used as an external shared source handler. The applet declared in the SOURCELOADER VALUE is used by the IMC to pull information for sources defined in its PLAYLIST. Only sources declared in the applet defined by the SOURCELOADER VALUE and in the IMC PLAYLIST can be displayed. Starting with IMS-4.0, any instance of IMC can be used as an external shared source handler. Simply specify the name of the IMC in this parameter value. For backwards compatibility, the Stand-alone SourceLoader applet is still available, but it is recommended that developers use the first IMC painted to the page to handle sources for the rest of the IMC applets. **Note** This parameter is required if there are no sources embedded in the IMC, however, IMC does not support both embedded source information and a defined SOURCELOADER parameter. IMC either uses a shared source handler declared in the SOURCELOADER VALUE to define sources or it must contain at least one complete set of source NAME/VALUE pairs to display video.
SCROLLBAR	Character class: [0-9 | A-Z | a-z | _] Name of a ScrollBar applet included in the page. **Note** A single ScrollBar applet can be shared between multiple IMC applets by using the Public Class Methods API to attach a ScrollBar with an IMC.

Table 9-2 Optional Fields (continued)

Command	Description
USEJOYSTICK	Boolean values: [false | true](true) This specifies whether joystick controls are enabled (true) or disabled (false). Joystick controls will only work if the USB joystick is installed on a Windows client machine. **Note** Joystick button mappings. Button 2: camera zoom in; Button 3: camera zoom out; Button 5: Increment preset number (1-60); Button 6: Decrement preset number (60-1)
HASJOYSTICKZ	Boolean values: [false | true](false) This indicates that the joystick can rotate on the Z <device>.
LOADINGTEXT	Character Class:[0-9 | A-Z | a-z | _]("Cisco") This specifies what text is displayed before video renders, AUTORUN parameter is false.

Table 9-3 IMC Source Parameters

Parameter	Description
NUMBEROFSOURCES	Format: [integer] Specifies the number of media sources.
ID	Character class: [0-9 | A-Z | a-z | _ | -] (default=source number [n] from SOURCE[n]%) A unique ID for this media source. If no ID is specified, IMC will use the number [n] from the SOURCE[n]%ARGUMENT to identify the source. To assure there are no conflicts, it is highly recommended developers define source IDs and use globally unique IDs throughout all applications.
SOURCE	Format: [name@address:port] name (required): The source name as it would be declared when starting the proxy or archive. address (optional): IP address/hostname.domain.extension of the source, if no address given, the address defaults to localhost. port (optional): Port number, if no port is given the port defaults to 80.
MEDIATYPE	Reserved values: [jpeg | mpeg2-v | mpeg4-v | audio] CODEC used to encode the media. • mpeg2-v: Video-only MPEG-2 Stream. • mpeg4-v: Video-only MPEG-4 Stream.

Table 9-3 IMC Source Parameters (continued)

Parameter	Description
STORAGE	Reserved values: [live | archive] This specifies whether the media source is live or archived.
LABEL	Character class: [0-9 | A-Z | a-z | _ | -] This specifies a label for this media source. When SOURCELISTVISIBLE is true this value will appear in IMC SourceList drop down menu. If the LABEL parameter is not set for a given source, IMC will populate the SourceList drop down menu with the source name, VSMS host, media type, and storage for that source (e.g. "proxy@vsms_host (live, jpeg)").
AUTHORIZATION	Format: [<username,password>]; Character class: [0-9 | A-Z | a-z] Specify username and password for direct view to devices with authorization enabled.
FRAMERATE	Range: [0.001-30] Specifies the number of frames per second of JPEG video to be displayed. If attempting to display a higher number of frames per second than is possible within the client's JVM and available bandwidth, IMC will display as many frames per seconds as possible.
TIMEOUT	Format: [integer in seconds] Specifies the number of seconds the media streams prior to disconnecting. To stream media continuously set TIMEOUT to Ø.
AUDIOENCODING	Reserved Values: [mulaw | gsm | g726_40 | g726_24 | g726_16](mulaw) This specifies the audio encoding mode. mulaw: µ-law 8-bit 64 kbps g726_40: G.726 5-bit 40 kbps g726_24: G.726 3-bit 24 kbps g726_16: G.726 2-bit 16 kbps gsm: GSM 13 kbps
DIRECTTOCAMERA	Boolean values: [false | true](false) Specifies whether this view is direct view [true] or proxy view [false]. **Note** This parameter is included for completeness, but it is recommended that it either be set to false or not be included. Viewing a feed directly from the encoder will affect all other connections to the device including, other clients and archives.
CAMERATYPE	Reserved values. The camera brand and model used for Pan-Tilt-Zoom and other camera controls.

Table 9-3 IMC Source Parameters (continued)

Parameter	Description
CAMERASOURCE	Format: [name@address:port] name (required): The camera input number. address (optional): IP address/hostname.domain.extension of the IP Device the camera is connected to. port (optional): Port number, if no port is given the port defaults to 80.
CAMERASOURCETYPE	Reserved Values. Type of device to either control Pan-Tilt-Zoom cameras or used with direct viewing.
CAMERACOMPORT	Reserved values: [COM1 | COM2] This specifies the COM port of the video encoder used for Pan-Tilt-Zoom and other camera controls.
CAMERACONTROLPROTOCOL	Reserved values: [P | D] This specifies the camera protocol.
CAMERACONTROLPRIORITY	Range: [1-100](100) Camera control priority policy implemented to handle contention between multiple Pan-Tilt-Zoom commands to a single camera, where 100 is the highest possible priority. The user with the highest priority gets exclusive use of the camera for CAMERA_CONTROL_PRIORITY seconds. So any concurrent requests with a lower priority will be rejected in this time interval.
PRESETS	Format: [<integer>=<preset description>;...] List of camera presets. Use number/value pairs separated by semi-colons such as . 1=Front Door and 2=Back Door.
CHAINNUMBER	Range: [0-64](0) The address of the target Pan-Tilt-Zoom camera in the case of multiple PTZ cameras connected in a daisy chain to the same serial port.

Stand-alone SourceLoader <APPLET>

IMC has a SOURCELOADER parameter to allow developers to easily share sources between IMC applets. One of the applets that can be defined in the SOURCELOADER parameter is called the Stand-alone SourceLoader. SourceLoader applets contain data shared among multiple instances of IMC which eliminates redundant information. This enables quicker development of complex, custom configurations while reducing the amount of code being maintained, parsed and queried for.

All of the source NAME/VALUE pairs for the Stand-alone SourceLoader applet are the same as defined for embedding them into an IMC applet. To use a Stand-alone SourceLoader applet with IMC instead of embedding them directly, simply include the SourceLoader <APPLET> tag above IMC, and set the

SOURCELOADER parameter VALUE to the NAME of the Stand-alone SourceLoader. Once a SourceLoader is declared for a particular IMC instance, only sources defined in the SourceLoader are available for that IMC. Sources embedded directly in the IMC will be ignored.

```
<APPLET NAME="[SourceLoader name]" CODE="com.cisco.client.SourceLoader"
    WIDTH=0 HEIGHT=0 CODEBASE="/java" MAYSCRIPT>
<PARAM NAME=cabbase             VALUE="IMC.cab">
<PARAM NAME=archive             VALUE="IMC.jar">
<!-- source NAME/VALUE pairs included here (see IMC sample above) -->
</APPLET>
```

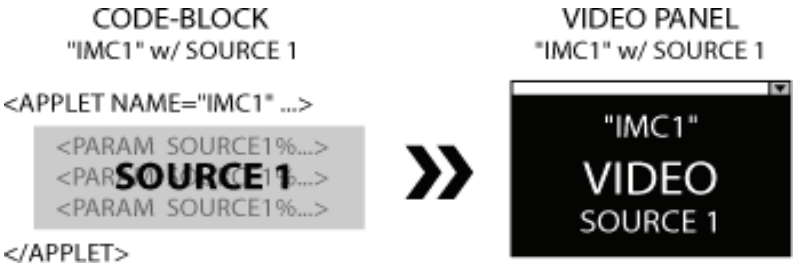
Note A simple single video panel does not require an external shared source handler. However, when developing a viewing template that contains several IMC applets, it is easier to handle sources externally by defining them once and sharing them. For backward compatibility it is still possible to use the Stand-alone SourceLoader applet, but starting with IMS-4.0, sharing sources using IMC applets only is recommended. This is illustrated in the IMC code-block examples below:

IMC Applet Code-block Examples

The following examples illustrate various configurations using IMC. Each IMC instance is displayed in <APPLET> code-blocks and the resulting video panel. The output contains separate blocks of source NAME/VALUE pairs which represent embedded or referenced video sources which are then available to be displayed.

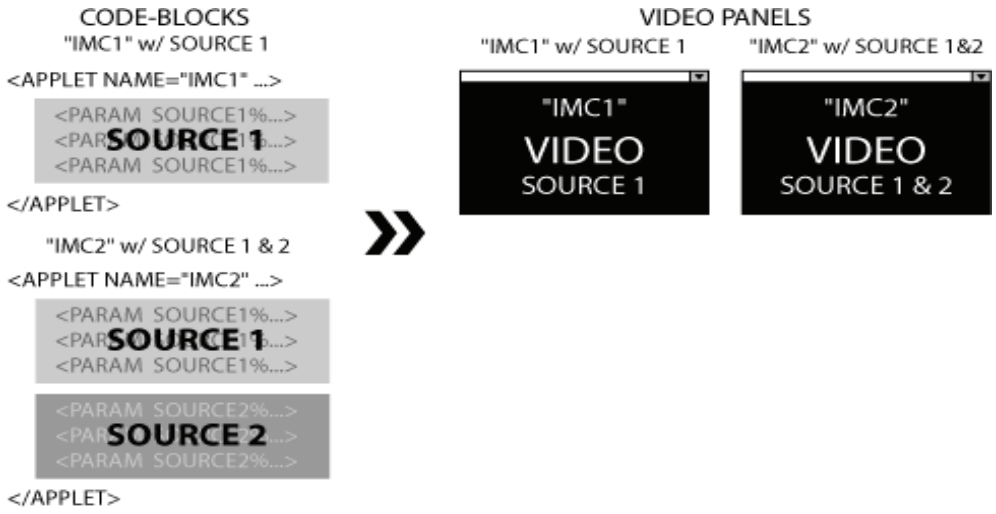
Example: Embed Source Directly in an IMC Applet

The following diagram illustrates that the IMC code-block has a single set of source NAME/VALUE pairs directly embedded.

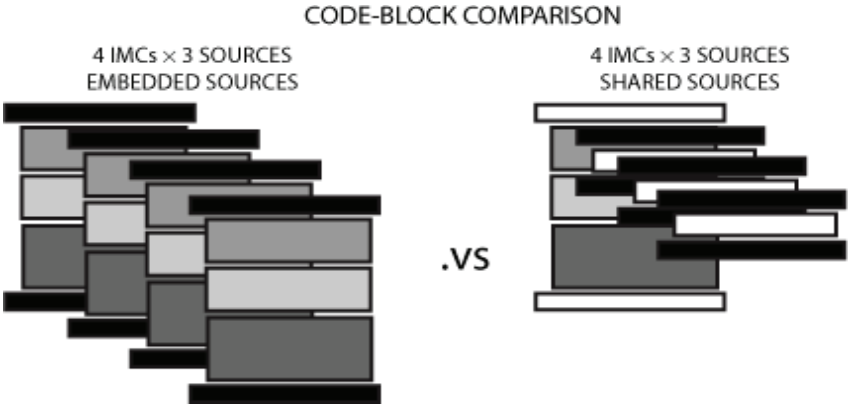


Example: Embed Sources Directly in Each IMC Instance

The following diagrams illustrate two IMC instances where each source is embedded directly into each instance. The instance of IMC named "IMC1" shows the simplest version, where a single video source is embedded directly in IMC and is displayed. Notice that the instance of IMC named "IMC2" has the same source as "IMC1" as well as an additional one. In other words the code-block for "SOURCE1%" is repeated in each IMC. This example also demonstrates that multiple IMC instances can be paired to display video from two separate sources simultaneously.



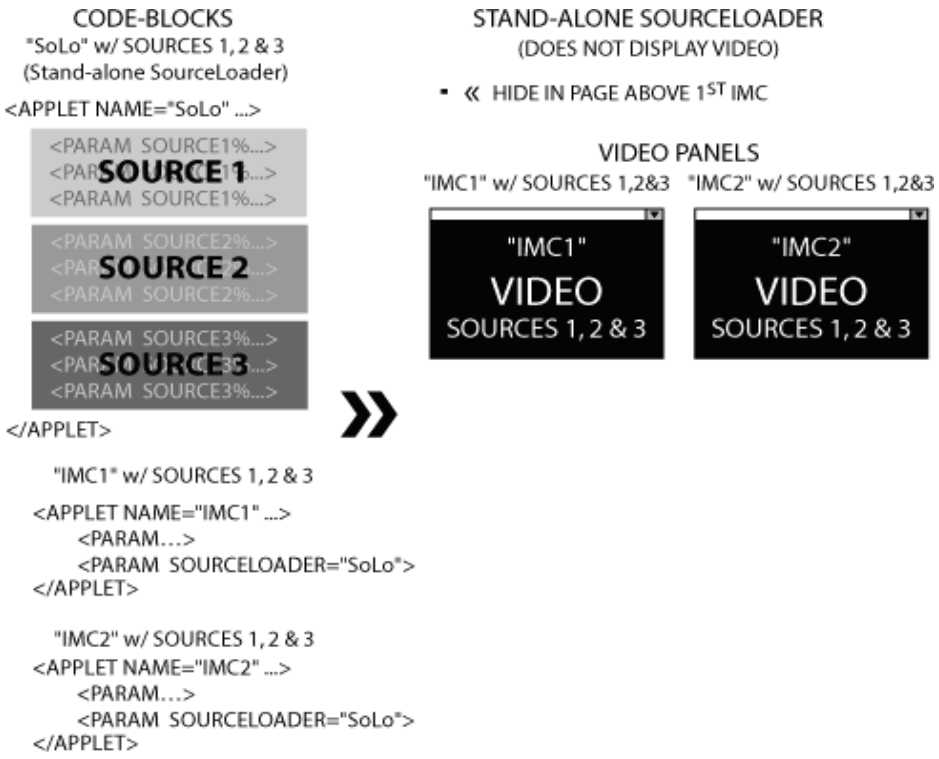
As viewing templates get more complex it is recommended to share sources instead of embedding sources directly in each IMC. The following diagram illustrates this point. Examples of different shared source handler options follow.



Example: Share Sources between Stand-alone SourceLoader and Multiple IMC Instances

The following diagram illustrates two IMC instances where sources are loaded from a Stand-alone SourceLoader applet. This example demonstrates the advantage of loading sources from a shared source handler, like a Stand-alone SourceLoader. Each source appears only once, so there is less code to maintain and parse, and only one query (or set of queries) is necessary to paint the sources to the page. Also note that the Stand-alone SourceLoader does not display any video so even though it must be included at the top of the page, it is best to hide it as well as possible.

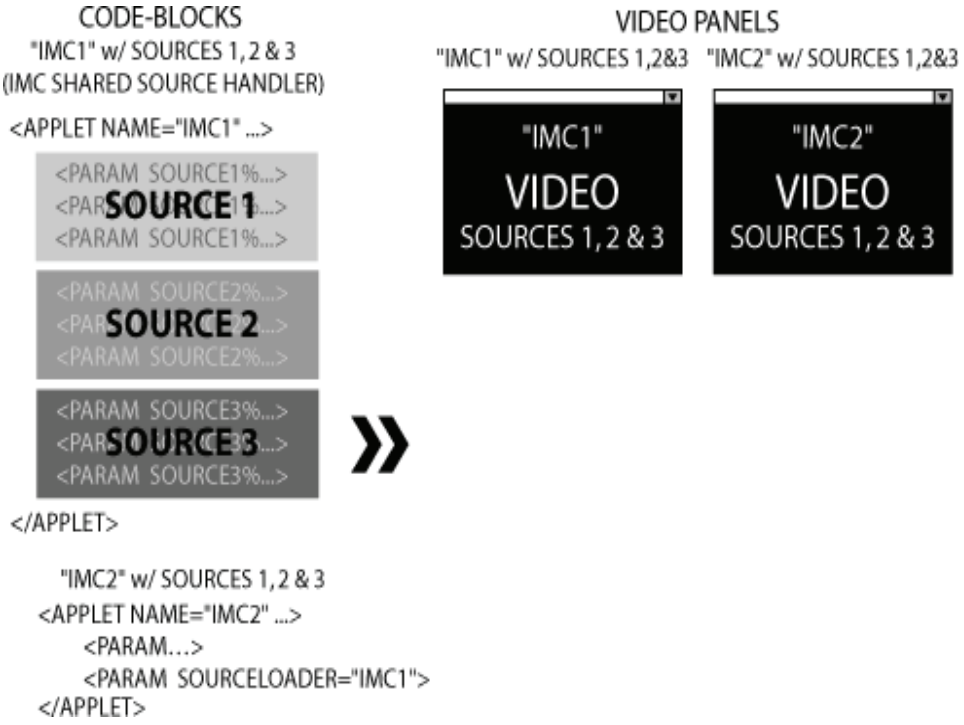
Note Starting with IMS-4.0, it is recommended that developers use the first IMC painted to the page as a shared source handler instead of a Stand-alone SourceLoader.



Example: Share Sources between One IMC and Other IMC Instances

The following diagram illustrates two IMC instances where sources are shared between one instance of IMC and another. This example demonstrates the advantage of loading sources from a shared source handler like IMC. Each source appears only once, so there is less code to maintain and parse, and only one query (or set of queries) is necessary to paint the sources to the page. And, since IMC can display video, it is not necessary to hide it as it is with a Stand-alone SourceLoader.

Note Starting with IMS-4.0, it is recommended that developers use the first IMC painted to the page as a shared source handler instead of using a Stand-alone SourceLoader.



Custom IMC Applet JavaScript Parameters

IMC also has optional Parameters affecting how IMC communicates with JavaScript. Add one or more of the following to the IMC parameters to declare what user-defined JavaScript function should be called for the given event. See Chapter 6, "Interactive Media Clients" for additional information.

Table 9-4 Custom Parameters

Parameter	Description
`onframechanged`	<User defined function name> Called when the video frame changes. **Note** Fires once for JPEG live feeds when the applet is first loaded. Does not fire when source changes. Returns: <VSMS name> and "-1". **Note** Fires continuously for playing JPEG archive feeds. Returns: <VSMS name> and <UTC frame timestamp> similar to 1054937153907.
`onplayratechanged`	<User defined function name> Called when the source play rate changes, as a result of the setPlayrate() method.
`onplayerloaded`	<User defined function name> Called when the applet is loaded.

Table 9-4 Custom Parameters (continued)

Parameter	Description
onsaveresponse	<User defined function name> Called after the archive clip is saved or an error is returned by save clip.
onseektimechanged	<User defined function name> Called when the seek time changes.
onsourceschanged	<User defined function name> Called when the Source is changed.
onstartofstream	<User defined function name> Called when source begins to stream. Use this call-back as the primary notification method to begin handling other events for this IMC applet.
onstarttimechanged	<User defined function name> Called when an archive's start time changes. Fires when the source changes and for loop archives as their properties are updated by VSMS (approximately every 30 seconds).
onstatechanged	<User defined function name> Called when the playback state changes.
onstoptimechanged	<User defined function name> Called when an archive's stop time changes. Fires when the source changes and for loop archives as their properties are updated by VSMS (approximately every 30 seconds).

Note When the applet calls the JavaScript functions, it passes information into the function. This information includes the NAME of the applet calling the function and may include additional information.

The following displays information passed to the JavaScript for each function. All arguments are passed as Strings. If an argument has a numerical value, the String must be converted to a number using functions such as parseInt() or parseFloat();

Table 9-5 JavaScript Information

Function	Argument
OnFrameChanged	name, frameTime
OnFramerateChanged	name, newFrameRate
OnPlayerLoaded	name
OnPlayrateChanged	name, newPlayRate (frame or bit rate)
OnSaveResponse	name, successful, confirmed, location, message
OnSeekTimeChanged	name, newSeekTime
OnSourceChanged	name, SourceId
OnStartOfStream	name

Table 9-5 JavaScript Information (continued)

Function	Argument
OnStartTimeChanged	name, newStartTime
OnStateChanged	name, state
OnStopTimeChanged	name, newStopTime

Example

Example function for onplayerloaded call-back:

```
<script>
function IMC_setLoaded( name ) // [name of the applet]
{
    logic here;
}
</script>

<APPLET NAME="IMC2"  CODE="com.cisco.client.IMC"
    WIDTH=352 HEIGHT=280 CODEBASE="/java" MAYSCRIPT>
<PARAM NAME=CABBASE             VALUE="IMC.cab">
<PARAM NAME=ARCHIVE             VALUE="IMC.jar">
<PARAM NAME=AUTORUN             VALUE="true">
<PARAM NAME=SOURCELISTVISIBLE   VALUE="true">
<PARAM NAME=TIMESTAMPVISIBLE    VALUE="true">
<PARAM NAME=PLAYLIST            VALUE="ID1;ID2;ID3">
<!-- Sources pulled from other IMC applet named, "IMC1" -->
<PARAM NAME=SOURCELOADER        VALUE="IMC1">
<PARAM NAME=SCROLLBAR           VALUE="Bar">
<PARAM NAME=ONPLAYERLOADED      VALUE="IMC_setLoaded">
</APPLET>
```

IMC 4.2 with Automatic Install and Update

Cisco IMC requires several DLLs and other items to be installed on the viewing client PC. By including this <OBJECT> tag in the pages that include IMC <APPLET> tags, the required software will automatically be installed. It is also possible to create a centralized page to permit users to install or update instead of including the <OBJECT> tag on every page.

Automatic Update Tag

```
<OBJECT CLASSID="clsid:A1E70050-43BC-4f38-B647-DBDBF46DE371"
    CODEBASE="/java/IMC.cab#Version=1,0,0,137">
</OBJECT>
```

Note This tag enables the automatic updates of the IMC related decoder(s) and DLLs.

IMC 4.2 with ScrollBar <APPLET>

Another applet that can be paired with IMC is the ScrollBar applet. Use this applet when playing back recorded media to move quickly to different segments of the archive. ScrollBar can have custom colors or graphics to seamlessly integrate it into any application. Associate ScrollBar applets with an IMC instance by including the SCROLLBAR parameter with the name of the ScrollBar applet. It is also possible to use the Public Class Methods API to attach a ScrollBar with an IMC.

ScrollBar Applet Tag

```
<APPLET NAME="[ScrollBar name]" CODE="com.cisco.client.Scrollbar"
    WIDTH=[width in pixels] HEIGHT=[height in pixels] CODEBASE="/java" MAYSCRIPT>
<PARAM NAME=cabbase     VALUE="IMC.cab">
<PARAM NAME=archive     VALUE="IMC.jar">
<!-- PARAMETERS to customize ScrollBar -->
<PARAM NAME=BARIMAGE    VALUE="[absolute path to GIF]">
<PARAM NAME=BARCOLOR    VALUE="[color in hexadecimal]">
<PARAM NAME=CARETIMAGE  VALUE="[absolute path to GIF]">
<PARAM NAME=CARETCOLOR  VALUE="[color in hexadecimal]">
<PARAM NAME=CARETSIZE   VALUE="[width in pixels]">
</APPLET>
```

Table 9-6 General ScrollBar Applet Parameters

Parameter	Description
NAME	Character class: [0-9 | A-Z | a-z | _] Name of a ScrollBar applet included in the page to be attached to IMC applets via the SCROLLBAR parameter or attachScrollbar() API method. **Note** A single ScrollBar applet can be shared between multiple IMC applets by using the Public Class Methods API to attach a ScrollBar with an IMC.
CODEBASE	Reserved value: [/java] Default directory applet is installed to /usr/bWhttpd/root/htdocs/java.
WIDTH	Range: [1-1000+] Width of applet in pixels. ScrollBar can scale as requirements dictate. **Note** The ScrollBar will stretch custom images.
HEIGHT	Range: [1-1000+] Height of applet in pixels. ScrollBar can scale as requirements dictate. **Note** The ScrollBar will stretch custom images.
CODE	Reserved value: [com.cisco.client.ScrollBar]
ARCHIVE	Reserved value: [IMC.jar] Netscape Communicator JAR archive
CABBASE	Reserved value: [IMC.cab] Microsoft Internet Explorer CAB archive

Table 9-7 Custom ScrollBar Parameters

Parameter	Description
BARIMAGE	Format: [absolute path to GIF] Specifies the absolute path to the GIF image used by ScrollBar for the background. For example, /BWT/sources/skin/scroll_bar.gif. **Note** The ScrollBar will stretch custom images and supports GIF transparency.
BARCOLOR	Format: [color in hexadecimal] Specifies the RGB color in hexadecimal format used by ScrollBar for the background. For example, 0x0000FF for blue.
CARETIMAGE	Format: `[absolute path to GIF]` Specifies the absolute path to the GIF image used by ScrollBar for the caret. For example, /BWT/sources/skin/caret.gif. **Note** The ScrollBar will stretch custom images and supports GIF transparency.
CARETCOLOR	Format: [color in hexadecimal] Specifies the RGB color in hexadecimal format used by ScrollBar for the caret. For example, 0xE0E0E0 for light grey.
CARETSIZE	Format: [width in pixels] Specifies the width in pixels used by ScrollBar for the caret. **Note** The ScrollBar will stretch custom images and supports GIF transparency.

VideoClient

Cisco VideoClient is a Java applet that requests and processes motion JPEG video streams. VideoClient can be embedded in a web page and viewed using a standard browser. Like the video.jpg thin-client, VideoClient is a simple way to view a JPEG proxy. It has no visible frames and can be resized as applicable.

VideoClient Applet Tag

```
<APPLET CODEBASE="/java" WIDTH="[applet width]" HEIGHT="[applet height]"
    CODE="com.cisco.videoclient.VideoClient">
<PARAM NAME="ARCHIVE"      VALUE="VideoClient.jar">
<PARAM NAME="CABBASE"      VALUE="VideoClient.cab">
<PARAM NAME="VIDEOWIDTH"   VALUE="[width in pixels]">
<PARAM NAME="VIDEOHEIGHT"  VALUE="[height in pixels]">
<PARAM NAME="FRAMERATE"    VALUE="[frames per second]">
<PARAM NAME="MESSAGE"      VALUE="[text message]">
<PARAM NAME="VIDEO"        VALUE="/video.jpg?source=[name@address:port]&timeout=0">
<PARAM NAME="CONNECTTYPE"  VALUE="[TCP | HTTP]">
<PARAM NAME="AUTOPLAY"     VALUE="[true | false]">
<PARAM NAME="CLICKCONTROL" VALUE="[true | false]">
<PARAM NAME="DEBUG"        VALUE="[true | false]">
<PARAM NAME="REPORT"       VALUE="[true | false]">
<IMG BORDER="0" SRC="/video.jpg?Source=[name@address:port]&framerate=0"
    WIDTH="[width in pixels]" HEIGHT="[height in pixels]" HSPACE="0" VSPACE="0"
```

```
    ALT="Java must be enabled to view this applet.">
</APPLET>
```

Table 9-8 General Applet Parameters

Parameter	Description
CODEBASE	Reserved value: [/java]
WIDTH	Range: [1-1000+ | 1-100%] Displays the width of applet in pixels. VideoClient can stretch and scale as requirements dictate. It is also possible to define a percentage value instead of hard pixels. The percentage will take typical DOM containers (tables, frames, etc.) into account, but it is possible to render video to 100% of the screen.
HEIGHT	Range: [1-1000+ | 1-100%] Height of applet in pixels. VideoClient can stretch and scale as requirements dictate. It is also possible to define a percentage value instead of hard pixels. The percentage will take typical DOM containers (tables, frames, etc.) into account, but it is possible to render video to 100% of the screen.
CODE	Reserved value: [com.cisco.videoclient.VideoClient]
ARCHIVE	Reserved value: [VideoClient.jar] Netscape Communicator JAR archive
CABBASE	Reserved value: [VideoClient.cab] Microsoft Internet Explorer CAB archive

Table 9-9 Customer VideoClient Parameters

Parameter	Description
AUTOPLAY	Boolean values: [true | false](default=true) If this value is true, the video clip will start the playback immediately after loading. If false, the video clip starts or stops the playback when the user clicks the video screen.
CLICKCONTROL	Boolean values: [true | false](false) If this value is true, the video clip starts or stops the playback when the video screen is clicked. If false, the video clip starts the playback when the HTML page containing the applet is loaded and stops when the HTML page is exited.
FRAMERATE	Range: [0.001-30] Specifies the number of frames per second of video to be displayed. If attempting to display a higher number of frames per second than is possible within the client's JVM, VideoClient will display as many frames per seconds as possible.

Table 9-9 Customer VideoClient Parameters (continued)

Parameter	Description
MESSAGE	Character class: [0-9 | A-Z | a-z | _ | <space> | ? | ! | -] The supplied message displays in the Java console and status bar of the browser when VideoClient is loaded.
REPORT	Boolean values: [true | false](false) If true, video statistics are displayed in the browser's Java Console. The statistics displayed are frames per second, kilobytes per second, and the total running time of the video clip. If false, the statistics are not displayed.
VIDEO	Reserved value: [/video.jpg] The specified value is the source of the video clip. A value of /video.jpg specify requests to the Video Proxy Source in the default server configuration. The VIDEO parameter is a required VideoClient parameter.
VIDEOWIDTH	Range: [160-720](352) Specifies the width in pixels of the video feed.
VIDEOHEIGHT	Range: [112-576](240) Specifies the height in pixels of the video feed.
CONNECTTYPE	Reserved values: [TCP | HTTP](TCP) This specifies connection type for VideoClient.
DEBUG	Boolean values: [true | false](false) If true, VideoClient will send debugging information to the Java console.

Cisco CamControl

CamControl is a Java applet that sends Pan-Tilt-Zoom commands to VSMS to control supported PTZ cameras. Many camera models complex administration options have been included in CamControl. See Chapter 7, "Camera Control API" or the Public Class Methods API section for additional information.

CamControl Applet Tag

```
<APPLET CODEBASE="/java" WIDTH="[width in pixels]" HEIGHT="[height in pixels]"
    CODE="com.cisco.cameracontrol.CamControl">
<PARAM NAME="ARCHIVE"    VALUE="CamControl.jar">
<PARAM NAME="CABBASE"    VALUE="CamControl.cab">
<PARAM NAME="ADMIN"      VALUE="[true | false]">
<PARAM NAME="BGCOLOR"    VALUE="[#RGB color]">
<PARAM NAME="FGCOLOR"    VALUE="[#RGB color]">
<PARAM NAME="CAMERA"     VALUE="[http://<ims host>/camera.bwt]">
<PARAM NAME="MODEL"      VALUE="[<device>2130 |<device> | <device> | <device> |
<device>]">
<PARAM NAME="IMAGESTRIP" VALUE="[image strip path]">
<PARAM NAME="PROXY"      VALUE="[proxy hostname]">
<PARAM NAME="SOURCE"     VALUE="[name@address:port]">
<PARAM NAME="DEVICE"     VALUE="[COM1 | COM2]"
```

```
<PARAM NAME="SRCTYPE"    VALUE="[<device>2130 | <device>2400 | <device>2400L |
<device>2401
   | <device>2401L]">
Java must be enabled to view this applet.
</APPLET>
```

Table 9-10 General Applet Parameters

Parameter	Description
CODEBASE	Reserved value: [/java]
WIDTH	Range: [120-1000+] Width of applet in pixels. CamControl can stretch and scale as requirements dictate, as long as controls are visible.
HEIGHT	Range: [120-1000+] Height of applet in pixels. CamControl can stretch and scale as requirements dictate, as long as controls are visible.
CODE	Reserved value: [com.cisco.cameracontrol.CamControl]
ARCHIVE	Reserved value: [CamControl.jar] Netscape Communicator JAR archive.
CABBASE	Reserved value: [CamControl.cab] Microsoft Internet Explorer CAB archive.

Table 9-11 Custom CamControl Parameters

Parameter	Description
ADMIN	Boolean values: [true | false](default=false) Enables [true] or disables [false] advanced camera features.
BGCOLOR	Format: [#RRGGBB](#C0C0C0) Sets background color of the CamControl client to the specified RGB hexadecimal value. Use this parameter to match the CamControl color to the corresponding color palette of the HTML page.
FGCOLOR	Format: [#RRGGBB](#000000) This sets foreground or text color of the CamControl client to the specified RGB hexadecimal value. Use this parameter to match the CamControl color to the corresponding color palette of the HTML page.
CAMERA	Format: [http://<vsms host>/camera.bwt](/camera.bwt) Sets the URL of the server-side camera module to pass the command from one VSMS host to another, typically to control cameras through firewalls. **Note** VSMS runs on port 80 by default. Specify additional ports here similar to vsms_host.cisco.com:8080.

Table 9-11 Custom CamControl Parameters (continued)

Parameter	Description
DEVICE	Reserved values: [COM1 | COM2] This specifies the COM port of the video source.
IMAGESTRIP	Format: [URL | path](/BWT/sources/camcontrol.gif) Specifies the image that CamControl uses to illustrate the Pan-Tilt-Zoom controls. Override this to provide an image which matches the color palette and design.
MODEL	Reserved values. The camera brand and model to control.
PROXY	Format: [http://IP address/hostname:port.domain.extension/camera.bwt](localhost) sets the remote address of the camera to control through the Remote Device Control Server proxy.
SOURCE	Format: [name@address:port] name (required): The camera input number. address (required): IP address/hostname.domain.extension of the video source. port (optional): Port number, if no port is given the port defaults to 80.
SRCTYPE	Reserved values. Specifies the type of video server to control the camera.

Pan-Tilt-Zoom cameras have the ability to store "preset" positions. See Chapter 7, “Presets Operations” section for additional information.

Cisco AudioClient

audio.bwt URL

Used in conjunction with the AudioClient applet.

Command

```
http://<host>/audio.bwt?server=<audio proxy name>&segment=<n>&features=<n>
```

Table 9-12 Required Fields

Field	Description
`host`	Format: [hostname.domain | IP address] Web address of host where VSMS is running. **Note** VSMS runs on port 80 by default. Specify additional ports similar to vsms_host.cisco.com:8080.
`server`	Format: [name] name (required): The source name. This is a proxy name or the camera input number on a media source such as a video server, network camera, or encoder. address (optional): IP address/hostname.domain.extension of the source, if no address given, the address defaults to localhost. port (optional): Port number, if no port is given the port defaults to 80. **Note** Remote address:port is not supported.
`segment`	Reserved value: [0 | 1 | 2 | 3 | 4] Audio encoding mode for an audio archive. • 0: µ-law 8-bit 64 kbps • 1: G.726 5-bit 40 kbps • 2: G.726 3-bit 24 kbps • 3: G.726 2-bit 16 kbps • 4: GSM 13 kbps

Table 9-13 Optional Fields

Field	Description
`features`	Reserved Values [0 | 1] • 0: raw audio data • 1: encoded data (not supported for IMS-3.0)

AudioClient Applet Tag

```
<APPLET codebase="/java" width="0" height="0" code="com.cisco.client.AudioClient.class">
   <PARAM name="cabbase" value="AudioClient.cab">
   <PARAM name="archive" value="AudioClient.jar">
   <PARAM name=SOURCE value="http://host/audio.bwt?server=[server name]&segment=[segment
value]&features=0">
</APPLET>
```

Table 9-14 General Applet Parameters

Parameter	Description
CODEBASE	Reserved value: [/java] Path to where applet is installed.
CODE	Reserved value: [com.cisco.client.AudioClient.class]
ARCHIVE	Reserved value: [AudioClient.jar] Enables Netscape Communicator JAR archive support.
CABBASE	Reserved value: [AudioClient.cab] This enables Microsoft IE CAB archive support.
SOURCE	Format http://host/audio.bwt?server=[server name]&segment=[0 | 1 | 2 | 3 | 4]&features=[0 | 1] **Note** Must be a fully qualified URL.

video.jpg Thin-client URL

The easiest way to view a JPEG proxy is to use the thin-client URL shown below. Additionally, this thin-client URL can be used with an HTML <IMG> tag.

Command

```
http://<host>/video.jpg?source=<proxy name>&framerate=<n>&timeout=<n>
```

Table 9-15 Required Fields

Field	Description
host	Format: [hostname.domain | IP address] Web address of host where VSMS is running. **Note** VSMS runs on port 80 by default. Specify additional ports here.
source	Format: [proxy name] proxy name (required): The source name. This is a proxy name or the camera input number on a media source such as a video server, network camera, or encoder. **Note** Remote address:port is not supported. To view a JPEG frame from another VSMS host, specify it in the <host> portion of the URL.

Table 9-16 Optional Fields

Field	Description
framerate	Range: [0 | 0.001-30](default=5) Maximum number of frames per second transmitted to view proxy. For a single frame snapshot use framerate=Ø. If the proxy frame rate is running lower than five frames per second, the proxy frame rate is the default frame rate.
timeout	Format: [integer in seconds](600) Time in seconds of when to disconnect the client. For no time out use timeout=Ø.

Example

```
http://vsms_host/video.jpg?source=camera1&framerate=0
```

Example

Using the HTML <IMG> tag included in a web page, the following displays the proxy "camera1" at a frame rate of 1 frame per second and a time out of 900 seconds:

```
<IMG SRC="http://vsms_host/video.jpg?source=camera1&framerate=1&timeout=900" WIDTH="160"
HEIGHT="120" HSPACE="0" VSPACE="0" ALT="View Proxy with HTML Image Tag">
```

Note Microsoft IE does not support frame rates >Ø).

audio.bwt URL

```
http://<host>/audio.bwt?server=<audio proxy name>&segment=<n>&features=<n>
```

CHAPTER 10

Advanced Configurations

Overview

VSMS contains a powerful security system enabling VSMS, VSVM and VSOM to work together to restrict API commands to trusted hosts and limit access to video streams to validated clients. The security features are optional and are easily enabled and configured through a simple text file.

Configuring Security

There are three parts to enabling and configuring security in a system:

1. API access control—Permits blocking of certain VSMS and VSVM URL commands so that they are only accepted from servers at specific IP addresses.
2. Stream access validation—Stream access authentication is used to limit access to video and audio streams to specific clients. This is done by authenticating each stream access with an application server such as VSOM. Working with a VSOM server, access to individual video streams can be controlled using the Rights on VSOM for a specific User and Role. Together, these features can be used to prevent BWT pages, browsers, the AXClient and Review Player from issuing API commands to VSMS and permitting the viewing of video streams unless authorization is granted.
3. password protection—Apache can be used to password protect access to the BWT pages on the VSMS system.

API Access Control and Stream Access Authentication

VSMS and VSVM use a trusted_ip file for access control. It resides in /usr/BWhttpd/conf/trusted_ip It is a text file with the following format: <IP Address> <Port> [<URL >] When this file exists, API access control on VSMS or VSVM hosts where it exists is enabled. The IP address is the address of a trusted host. The port number and URL are optional and specify the path to an application server Stream Access Authenticator. This is used for permitting stream level access control to a specific video stream. The location of this file is the same even if VSMS and VSVM are installed on the same host, and in this case, they share the file.

VSMS API Access Control

VSMS limits access to critical URL commands using the trusted_ip file. If the trusted_ip file is present on a VSMS host, the URL commands with the roots command.bwt or cgi-bin/smanager are blocked if they do not originate from an IP address listed in the trusted_ip file. Commands that use these VSMS API URL roots include critical commands for starting, stopping, updating and deleting proxies and archives and managing storage.

VSVM API Access Control

VSVM limits access to all VSVM URL commands using the trusted_ip file. If the trusted_ip file is present on a VSVM host, all URL commands are blocked unless they originate from an IP address listed in the trusted_ip file.

Stream Access Authentication

When the trusted_ip file is present on a VSMS host, VSMS will also authenticate the access to video and audio streams. If the IP address of the client requesting access to a stream is listed in the trusted_ip file as a trusted host, access to all streams on the VSMS server will be allowed to that client. If this is not the case, VSMS will look for stream access authenticator URLs in the trusted_ip file and will request authentication for the stream access for the client. In a system, this authenticator URL will typically point to a VSOM or VSVM server. VSMS will request validation from each stream authenticator listed in the trusted_ip file until the access is either authenticated or no additional authenticators remain. Both VSOM and VSVM can act as authenticators.

VSOM Authenticator URL

The format of the VSOM authenticator URL including the port # is 80 /vsom/service/security.php

VSVM Authenticator URL

The format of the VSVM authenticator URL including the port # is`8086 /security.bwt`

Example trusted_ip files

Here is an example of a trusted_ip file that would be installed on a VSMS server:

```
10.10.50.140 80 /vsom/service/security.php #VSOM server and authenticator url
10.10.50.58 8086 /security.bwt # VSVM server and authenticator URL
```

The first line in this example serves several purposes. The IP address and port indicates that the VSOM host at 10.10.50.140 is allowed to issue all URL commands to this VSMS host. Secondly, the URL tells VSMS that this host can also act as a stream Authenticator. Therefore, when clients attempt to access a video stream, VSMS will send a validation request to this VSOM host using this URL.

The second line provides a VSVM host IP address and authenticator URL. This tells VSMS that any client stream access not authenticated by the first line should be sent to this host and URL. This will permit VSVM clients of the VSVM host to have their stream access requests authenticated. The 8086 port number is needed because VSVM communications are handled on this port.

Unix style comments are supported, ignoring everything on a line after a '#'.

In this example, there would likely be a trusted_ip file installed on the VSVM host as well. The trusted_ip would look similar to:

```
10.10.50.140 #VSOM server
```

This tells the VSVM server to accept URL commands from the VSOM server at 10.10.50.140. In this example, the BWT pages on the VSMS host would be accessible to clients, but it would not be possible to view video streams through them at a client unless that client was also logged into VSOM and had rights to streams on the VSMS host. Only those streams would be viewable. Additionally, critical API commands could not be executed from the BWT pages or the browser.

Co-Installation Special Cases

When VSMS, VSOM and VSVM are co-installed on the same host, in any combination of two or all of them, there are some special circumstances for configuring the trusted_ip file. Both VSMS and VSVM automatically accept all URL commands that originate from the same host. An IP address is not required in the trusted_ip file to permit this. However, when stream authentication is to be done by a co-installed application, the IP address of the host (even though it is the same host) must be provided along with the authenticator URL to use. When VSMS, VSOM and VSVM are all co-installed on the same host, it is likely that both VSOM and VSVM will be used as authenticators for stream accesses. This will require the IP address of the host to appear twice with the two different authenticator URLs.

Example: VSMS and VSOM Co-Installed

VSMS and VSOM are installed on the same host. VSVM is installed on a third. The VSMS/VSOM host IP address is 10.10.20.20. The VSVM host IP address is 10.10.20.25. URL commands from VSOM to VSMS are automatically allowed, but users need to indicate the stream authenticators on the co-installed host. On the VSMS/VSOM host, the trusted_ip file looks similar to:

```
10.10.20.20 80 /vsom/service/security.php # Use co-installed VSOM for stream authenticator
10.10.20.25 8086 /security.bwt # Use remote VSVM for stream authenticator
```

On the VSVM host, authorize URL commands from the VSOM server. So the trusted_ip file looks similar to:

```
10.10.20.20#Allow URL commands from the VSMS/VSOM host.
```

The port number is not required because it is only used for issuing stream authentication requests.

Example: VSMS and VSVM Co-Installed

VSMS and VSVM are installed on the same host. VSOM is installed on a third. The VSMS/VSVM host IP address is 10.10.20.20. The VSOM host IP address is 10.10.20.25. On the VSMS/VSVM host, the trusted_ip file looks similar to:

```
10.10.20.20 8086 /security.bwt # Use co-installed VSVM for stream authenticator
10.10.20.25 80 /vsom/service/security.php # Use remote VSOM for stream authenticator
```

On the VSOM host, there is no need for a trusted_ip file. VSOM does not need or support this file.

Example: VSOM and VSVM Co-installed

VSVM and VSOM are installed on the same host. VSMS is installed on a third. The VSVM/VSOM host IP address is 10.10.20.20. The VSMS host IP address is 10.10.20.25. URL commands from VSOM to VSVM are automatically allowed. Since VSMS is not installed, on this host, there is no stream authentication in the trusted_ip file. To block URL commands from other hosts, the trusted_ip file must be present but can be empty. On the VSMS host, authorization is required for the URL commands from the VSOM server and list VSOM and VSVM as stream authenticators. The trusted_ip file on the VSMS server looks similar to:

```
# Allow URL commands from remote VSOM and use for stream authenticator 10.10.20.20 80
/vsom/service/security.php
# Use remote VSVM for stream authenticator 10.10.20.20 8086 /security.bwt
```

Example: All Installed

VSMS, VSVM and VSOM are all co-installed on the same host at IP address 10.10.20.20. VSMS and VSVM will automatically support URL commands from VSOM. Only stream authentication needs to be defined.

```
10.10.20.20 80 /vsom/service/security.php # Use co-installed VSOM for stream authenticator
10.10.20.20 8086 /security.bwt # Use co-installed VSVM for stream authenticator
```

Multiple IP Addresses on a Single Host

It is possible to have multiple IP addresses on a single host because of the presence of multiple Ethernet ports or NICs. When VSMS, VSOM and VSVM are installed on separate hosts, this will have a negligible effect. The trusted_ip files on the VSMS and VSVM hosts should have the proper IP addresses and if multiple IP addresses will be used on the same host, they need to be properly reflected in the trusted_ip file. In the case of co-installed applications, VSMS and VSVM automatically allow URL commands originating from the first IP address on the host. If URL commands need to originate from one of the additional IP addresses on the host, they must be listed in the trusted_ip file in order to be allowed.

CHAPTER 11

On Demand Viewer/Media Out

The Media Out streaming solution has been designed to increase scalability and performance, build a control framework for large-scale live and recorded streaming using standard protocols, and simplify multimedia presentation creations by isolating the client applications from the complexities of data transport and synchronization. Backward compatibility with the AXClient has been maintained.

How It Works

The media out subsystem relies on lightweight HTTP and RTSP handlers. These are necessary to reduce memory and CPU footprints, as well as to provide scalability. The subsystem will employs multi-threaded model where the main framework opens listening ports for RTSP, HTTP, and HTTPS connections. When a connection arrives, the framework produces a new HTTP or RTSP thread to handle the incoming request.

Major media out components include:

- Framework — responsible for setting up the sockets for the clients to connect to the media out subsystem.
- RTSP Handler — handles the RTSP control session with the client. The RTSP control session handles requests as follows:
  - DESCRIBE—calls the appropriate module to query the specified media stream for its SDP information.
  - SETUP — causes the handler to create or update a session object, create a subsession object for the specified media stream, and add it to the session object. For each media stream that uses UDP as a transport, the UDP socket handler of the media out library will be called to allocate UDP sockets for the RTP and RTCP transport.
  - PLAY — requests for subsessions which are in the SETUP state causes a thread to be produced to start play back of the media stream. For subsessions which are in the PLAY stat, the request is passed to the associated thread so it can determine new play time range criteria. For subsessions which are in the PAUSE state, a message is sent to the associated thread to start playing again. Requests for subsessions in any other state are rejected.
  - PAUSE — requests for subsessions which are in the PLAY state are sent to the associated thread to pause play back. Requests for subsessions in any other state are rejected.
  - TEARDOWN — requests are sent to the applicable subsession thread to cause the subsession to stop playback and terminate.

A RTSP control session with a client can be terminated by the client at any time, however, all active media streaming sessions with the client MUST remain active and will only terminate upon receipt of an associated TEARDOWN message or the non-receipt of RTCP Receiver Report messages.

- HTTP Handler — the HTTP handler is requested by the info.bwt to call the applicable media module to retrieve, format, and return the requested information.
- HTTP Proxy — the HTTP proxy permits one HTTP server to interface to outside networks. The proxy acts as a pass-through handler.
- Media Out Library —
  - Live Media Reader
  - Archived Media Reader
  - Media Encoders and Packetizers
  - RTP/RTCP Handlers
  - Transport Handlers

RTSP Methods

The standards-based communication between the server, the client, and media players (QT, VLC) will follow RTSP protocol.

OPTIONS

The OPTIONS method is sent by a client to determine what methods are supported by a Server.

Client to Server:

```
OPTIONS rtsp://audio.example.com/foobar/audio.en RTSP/1.0
    CSeq: 3
```

Server to Client:

```
RTSP/1.0 200 OK
    CSeq: 3
Public: DESCRIBE, SETUP, PLAY, PAUSE, TEARDOWN
```

DESCRIBE

The DESCRIBE method requests the Server to return information about a media stream or media presentation. The RTSP Server will only accept requests that expect SDP as the returned format.

Client to Server:

```
DESCRIBE rtsp://server.example.com/demo/548/foobar RTSP/1.0
CSeq: 1
Accept: application/sdp
```

Server to Client:

```
RTSP/1.0 200 1 OK
```

```
Content-type: application/sdp
Content-Length: 44

v=0
o=- 2890844526 2890842807 IN IP4 192.16.24.202
s=RTSP Session
m=audio 0 RTP/AVP 0
a=control:rtsp://audio.example.com/foobar/audio.en
m=video 0 RTP/AVP 31
a=control:rtsp://video.example.com/foobar/video
```

SETUP

The SETUP method requests the Server to setup and reserve resources for the play back of a media stream or presentation. The RTSP Server will support all the transport protocols: TCP (with interleaved data), unicast, and multicast UDP.

Client to Server:

```
SETUP rtsp://audio.example.com/foobar/audio.en RTSP/1.0
CSeq: 1
Transport: RTP/AVP/UDP;unicast;client_port=3056-3057
```

Server to Client:

```
RTSP/1.0 200 OK
CSeq: 1
Session: 12345678
Transport: RTP/AVP/UDP;unicast;client_port=3056-3057;
                  server_port=5000-5001
```

Client to Server:

```
SETUP rtsp://video.example.com/foobar/video RTSP/1.0
CSeq: 1
Transport: RTP/AVP/UDP;unicast;client_port=3058-3059
```

Client to Server:

```
RTSP/1.0 200 OK
CSeq: 1
Session: 23456789
Transport: RTP/AVP/UDP;unicast;client_port=3058-3059;
                  server_port=5002-5003
```

PLAY

The PLAY method initiates the start of play back of a media stream or presentation. Multiple PLAY requests for the same session can be sent by a client. If the URL specifies an existing media stream, the play requests are executed in the order they are received and this may result in the coalescing of time ranges, See RFC 2326 for details.

If the URL specifies different media streams then the specified media stream to be started, a setup must have been provided and received for that stream. The RTSP Server will accept the following headers/parameters as part of the PLAY request as defined in RFC 2326:

```
"Range: npt or clock (absolute) time ranges, but NOT the time parameter
"Scale
```

Client to Server:

```
PLAY rtsp://video.example.com/foobar/video RTSP/1.0
    CSeq: 2
    Session: 23456789
    Range: npt=0:10:00-
```

Server to Client:

```
RTSP/1.0 200 OK
CSeq: 2
Session: 23456789
Range: npt=0:10:00-0:20:00
RTP-Info: url=rtsp://video.example.com/foobar/video;
         seq=12312232;rtptime=78712811
```

Client to Server:

```
PLAY rtsp://audio.example.com/foobar/audio.en RTSP/1.0
CSeq: 2
Session: 12345678
Range: npt=0:10:00-
```

Server to Client:

```
RTSP/1.0 200 OK
CSeq: 2
Session: 12345678
Range: npt=0:10:00-0:20:00
RTP-Info: url=rtsp://audio.example.com/foobar/audio.en;
         seq=876655;rtptime=1032181
```

PAUSE

The PAUSE method requests the play back of the specified media stream or presentation be paused. Outstanding play requests for this stream or presentation will be deleted. Playback will be resumed when a new PLAY request is received for the media stream or presentation.

```
Client->Server:
    PAUSE rtsp://audio.example.com/foobar/audio.en RTSP/1.0
        CSeq: 3
      Session: 12345678
```

Server to Client:

```
RTSP/1.0 200 OK
    CSeq: 3
```

Client to Server:

```
PAUSE rtsp://video.example.com/foobar/video RTSP/1.0
   CSeq: 3
    Session: 23456789
```

Server to Client:

```
RTSP/1.0 200 OK
     CSeq: 3
```

TEARDOWN

The TEARDOWN method terminates the media stream or presentation and the Server reclaims all resources associated with session.

Client to Server:

```
TEARDOWN rtsp://audio.example.com/foobar/audio.en RTSP/1.0
     CSeq: 3
   Session: 12345678
```

Server to Client:

```
RTSP/1.0 200 OK
     CSeq: 3
```

Client to Server:

```
TEARDOWN rtsp://video.example.com/foobar/video RTSP/1.0
    CSeq: 3
     Session: 23456789
```

Server to Client:

```
RTSP/1.0 200 OK
     CSeq: 3
```

CHAPTER 12

Command Line Tools

Command Line Tools are utilities used to manage the event database and archive repositories. Storage Manager related tools are used via URL-based APIs. These tools are located at /usr/BWhttpd/bin.

Remove Triggered Events: remove_events

Event related tool permits users to remove event entries from the event database. This tool is located at /usr/BWhttpd/bin. Remove the specified events entries from the event database. The event entries can be removed by event name and/or by event time range.

Note If there is an event archive corresponding to an event entry, the archive will not be removed.

Command

```
remove_events [name] [startUTC] [stopUTC]
```

Example

Remove all the event entries from the event database for the event named, EVENT1.

```
/usr/BWhttpd/bin/remove_events EVENT1
```

Example

Remove all the event entries for EVENT1, created between the start UTC time of 1045183402 and stop UTC time of 1045185500.

```
/usr/BWhttpd/bin/remove_events EVENT1 1045183402 1045185500
```

Example

Remove all the event entries for all events created between the start UTC time of 1045183402 and stop UTC time of 1045185500.

```
/usr/BWhttpd/bin/remove_events 1045183402 1045185500
```

Cut-out Archive Clip: coutar

This command extracts a clip from an existing archive and adds the clip to the Storage Manager repository. Save the archive clip to a remote or local host. This command requires several arguments which are parsed by position.

Command

```
coutar source_id startUTC endUTC Days-to-Live "archive description" target_id@address
savemode [repository] [saveformat] [URL to notification handler]
```

Table 12-1 Required Fields

Position	Description
1	Format: [source_id] source_id (required): Source archive name. This is the parent archive required to create a clip from.
2	Format: [Start in UTC milliseconds] Start date of the child clip in UTC milliseconds. Make sure the parent archive contains data for this date.
3	Format: [Stop in UTC milliseconds] Stop date of the child clip in UTC milliseconds. Make sure the parent archive contains data for this date.
4	Format: [integer in days] Number of days that the archive will be stored before system removal. For permanent storage set daystolive=Ø.
5	Character class: [0-9 | A-Z | a-z | _ | <space> | -]; Length: (0-20) Brief description for the new archive clip. Wrap descriptions in double-quotes (e.g. archive description). **Note** For an archive clip to have type clip in the archive listing (http://<host>/info.bwt?type=archive), description must be clip.
6	target_id@address target_id: The new archive clip name. address (optional): IP address/hostname.domain.extension of target host to where clip is saved. If no address is provided, then clip will be saved on the local host or where it has been configured on VSM Console.
7	Format: [local] Indicates where to save the archive clip. local: Save clip to this VSMS host. remote: If name parameter (in form of target_id@address:port) is specified, then VSMS will save the clip to the specified host. If no host is specified, see the VSM Console pages localandremote: Save clip to this VSMS host and to a remote host.

Table 12-2 Optional Fields

Position	Description
8	Format: [/repository_mount] Repository location where this clip will reside. If no repository is specified for a local clip, see the VSM Console pages.
9	Format: [regular | bwm | bwx] (default=regular) Indicates what type of archive clip to generate.
10	Format: [http://<host>/handler_path] URL to send on completion of a clip.
11	Format: -key The key used to sign a .bwx clip and is prefixed with -.

Example

Extract a clip, from the Hallway_Monday with start date/time in UTC of 1042133280000 and end date/time in UTC of 1042133400000 with permanent storage called HallWay_0130_Clip archive to be saved to local host. Send status to the notification handler servlet Echo when clip is completed.

```
/usr/BWhttpd/bin/coutar Hallway_Monday 1042133280000 1042133400000 0 "clip"
HallWay_0130_Clip local "http://myhost/servlets/Echo?"
```

List Archives: listar

Prints a comma delimited list of unexpired archives. The fields listed in order are archive name, directory path, archive type, requested size in Kbytes, actual archive size, status of archive, archive begin time, archive end time, and the archives expiration time.

Command

```
listar [-p | -h]
```

Options

List the archives and applicable information in a readable format.

`-p`

Prints the help

`-h`

Example

List archives in the Storage Manager repository using the readable print option.

```
/usr/BWhttpd/bin/listar -p
0
            ID: A_Bldg-12fl
          Path: /data1/A_Bldg-12fl
          Type: loop
Requested Size: 24073404
   Actual Size: 24061732
         State: SHELVED
     Born Time: Tue Dec 10 12:50:28 2002
     Stop Time: Thu Dec 12 10:08:16 2002
   Expire Time: lifetime!!
```

Example

List archives in the Storage Manager repository using an ssv output option.

```
/usr/BWhttpd/bin/listar
id;path;type;requestedSize;actualSize;status;beginTime;endTime;expireTime
A_Bldg-12fl;/data1/A_Bldg-12fl;loop;24073404;24061732;SHELVED;...
    ...Tue Dec 10 12:50:28 2002;Thu Dec 12 10:08:16 2002;lifetime
```

Remove Archives: rmar

Removes stopped archives from the Storage Manager repository and disk allocation. The absolute path must be used in order to remove the archive. Wildcards for archive names are supported.

Command

```
rmar /<full path>/archive_name1 [/<full path>/archive_name2 /<full path>/archive_name3...]
```

Options

Prints the help

```
-h
```

Examples

Remove archive1

```
rmar /data/archive1
```

Remove all archives with names starting with archive.

```
rmar /media0/archive*
```

Remove archives Jan01 and Feb01.

```
rmar /data/Jan01 /data/Feb01
```

Query Repository for Storage Availability: rquery

Returns disk space information.

Command

```
rquery <requested size in kilobytes>
```

Example

Verify that the Storage Manager has 5000000 KB available.

```
/usr/BWhttpd/bin/rquery 5000000
```

List Repository Disk Usage: rusage

This command will display the repository mount point and the amount of free space and used space based on the repository setting in the system.cfg file.

Command

```
rusage [-h]
```

Options

Prints the help

```
-h
```

Example

Returns the disk space usage amount.

```
/usr/BWhttpd/bin/rusage
Repos Mount Point: /media0
             Size: 42601197568
        Used (DF): 937213952
        Free (DF): 39499915264
       Used Space: 1726914699
       Free Space: 38710214517
```

where:

Repos Mount Point is defined in the system.cfg file.

Size is the total disk space for the Mount point.

Used (DF) is the actual disk space up to this point in time (using the Unix df command)

Free (DF) is the disk space not used up to this point in time.

Used Space is the disk space used by the archives registered with the Storage Manager.

Free Space is the disk space available for starting new archives

Note All values are in kilobytes.

CHAPTER 13

SNMP Configuration

This chapter describes the Simple Network Management Protocol (SNMP) and SNMP configuration on Cisco Video Surveillance Media Server (VSMS), and contains the following sections:

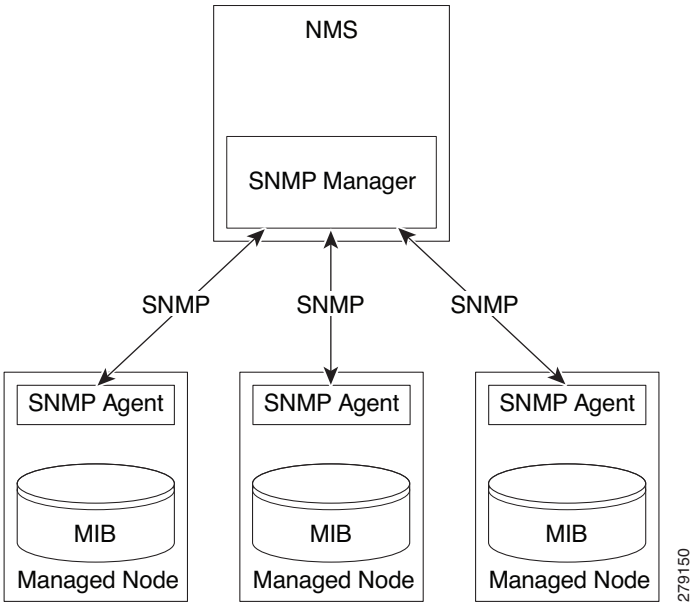
SNMP Overview

This section contains the following topics:

SNMP Framework

SNMP is an application-layer protocol that provides a message format for communication between SNMP managers and SNMP agents. It provides a standardized client-server framework and a common language used for the monitoring and management of devices in a network. Figure 13-1 shows an example of a simple SNMP client-server framework.

Figure 13-1 SNMP Client-Server Framework



The SNMP manager is the system that is used to control and monitor the activities of network nodes that use SNMP. The most common managing system is called a Network Management System (NMS). The term NMS can be applied to either a dedicated device used for network management, or the applications that are used on such a device. A variety of network management applications are available for use with SNMP. These features range from simple command-line applications to feature-rich graphical user interfaces.

The SNMP agent is the software component within a managed node that maintains the data for the device and reports the data, as needed, to SNMP managers. An SNMP agent resides on the VSMS platform, enabling the platform to function as a managed node.

VS Event MIB

A management information base (MIB) is a formal description of a set of objects that can be monitored and managed using SNMP. The VS Event MIB allows for specialized monitoring capabilities. It provides an asynchronous notification mechanism that is supported by SNMP and that can be set to monitor any SNMP MIB object on VSMS and perform notification (trap or inform) operations when specific triggers (conditions) occur. Table 13-1 describes the triggers and trap events that are supported by the VS Event MIB. For more information about the event MIB, see the "BROADWARE-EVENT-MIB Definition" section on page 13-6.

Table 13-1 Triggers and Trap Events

Trigger	Trap Event	Description
Archive Started	bwArchiverEvent	Notification that simple, loop, or recurring archives scheduled through VSOM have started.
Archive Stopped	bwArchiverEvent	Notification that simple, loop, or recurring archives scheduled through VSOM have stopped.

Table 13-1 Triggers and Trap Events (continued)

Trigger	Trap Event	Description
Loss of Connection to Device	bwConnectionEvent	Notification that VSMS has lost communication with an encoder or IP camera, which cause a loss of video feed to the VSMS host. Reconnection is attempted.
Reestablish Connection to Device	bwConnectionEvent	Notification that reconnection has been established.
Proxy Added	bwProxyEvent	Notification that a proxy has been added to the system.
Proxy Failed	bwProxyEvent	Notification that a proxy has encountered an internal error.
Proxy Deleted	bwProxyEvent	Notification that a proxy has been deleted from the system.
View Proxy	bwProxyEvent	Notification that a proxy has been connected to the device.

To translate BROADWARE-EVENT-MIB SNMPv2c traps to understandable event names, the BROADWARE-EVENT-MIB.txt definition file must be installed on the NMS. For more information about downloading the BROADWARE-EVENT-MIB.txt definition file, see the "Downloading the VS Event MIB" section on page 13-4.

SNMP Notifications

A key feature of SNMP is the ability to generate notifications from an SNMP agent. These notifications do not require that requests be sent from the SNMP manager. Unsolicited (asynchronous) notifications can be generated as traps or inform requests. Traps are messages that alert the SNMP manager to a condition on the network. Inform requests (informs) are traps that include a request for confirmation of receipt from the SNMP manager. Notifications indicate significant VSMS events. For more information, see Table 13-1 on page 13-2.

Traps are less reliable than informs because the SNMP manager does not send any acknowledgment when it receives a trap. The SNMP agent cannot determine if the trap was received. However, traps often are preferred because informs consume more resources in VSMS and in the network.

SNMP trap destinations (SNMP managers) must be configured in VSMS. All traps are enabled in VSMS by default, but if trap destinations are not configured, SNMP traps are not sent. The list of SNMP trap destinations and other notification configuration information are shared by all events and are read when VSMS initializes. SNMP trap destination configuration is done through the Video Surveillance Management Console (VSMC). For more information, see "Configuring an SNMP Trap Destination" section on page 13-4.

SNMP Versions

VSMS supports both SNMPv1 and SNMPv2c traps. However, we recommend using SNMPv2c because of its enhanced MIB support. VSMS also supports SNMPv2c Inform messages, which are identical to trap messages except that an inform message is acknowledged by the NMS, which adds a layer of reliability.

Note If VSMS is configured to use SNMPv2c traps, an NMS must be capable of receiving, parsing, and presenting SNMPv2c traps.

Viewing SNMP Monitoring Status

To view the SNMP monitoring status, click the **SNMP Trap Destinations** link under the Configuration category in the sidebar of the Management Console. The status is shown in the SNMP Monitoring area of the SNMP Trap Destinations page. The status can be either of the following:

- On—SNMP service is operating. This state is the system default.
- Off—SNMP service has not started, has been stopped, or has failed.

  To troubleshoot this issue, start by issuing the **/etc/init.d/cisco status** Linux command to check the status of the SNPM service VSMS host.

Downloading the VS Event MIB

SNMP trap destinations on client SNMP servers use the VS Event MIB file (BROADWARE-EVENT-MIB.txt) to interpret traps that they receive from a VSMS host.

To download the VS Event MIB file, perform the following steps:

Step 1 Click the **SNMP Trap Destinations** link under the Configuration category in the sidebar of the Management Console.

Step 2 Click the **VS Event MIB** link in the SNMP Trap Destinations area of the SNMP Trap Destinations page.

Configuring an SNMP Trap Destination

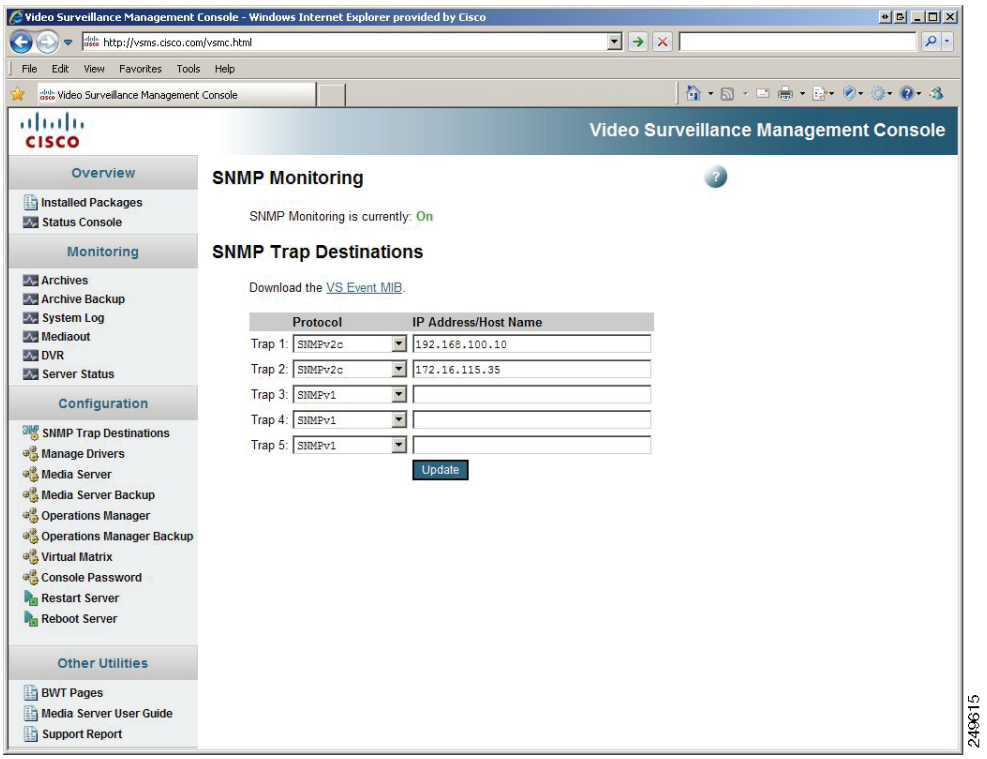
Note

- The VSMS host automatically generates SNMP traps on the local disk. It performs this process even if you do not configure any SNMP trap destinations.
- VSMS supports both SNMPv1 and SNMPv2c traps. However, we recommend using SNMPv2c.
- Running a third-party trap receiver on a VSMS host is not supported. If you are using a third-party trap receiver, configure it on another system in your VSM network.
- You can configure up to five SNMP trap destinations.

To configure an SNMP trap destination, perform the following steps:

Step 1 Click the **SNMP Trap Destinations** link under the Configuration category in the sidebar of the Management Console.

The SNMP Trap Destinations page appears.



Step 2 From the Protocol drop-down list, choose **SNMPv2c**.

Step 3 In the IP Address/Host Name drop-down field, enter the IP address or hostname of the server to receive SNMP traps.

Note Leading protocol strings (for example, http://) and port numbers (for example, 8080) are not allowed.

Step 4 (Optional) Repeat Step 1 through Step 3 for each additional trap destination that you want to configure.

Step 5 Click the **Update** button.

Step 6 (Optional) To verify that the SNMP trap destinations are successfully placed in the /usr/BWhttpd/etc/snmpd.conf directory, issue the **more /usr/BWhttpd/etc/snmpd.conf | grep trap2sink** Linux command.

BROADWARE-EVENT-MIB Definition

The BROADWARE-EVENT-MIB Definition is as follows:

```
BROADWARE-EVENT-MIB DEFINITIONS ::= BEGIN


IMPORTS
    MODULE-IDENTITY, OBJECT-TYPE, Integer32, enterprises,
    NOTIFICATION-TYPE                       FROM SNMPv2-SMI
    SnmpAdminString                         FROM SNMP-FRAMEWORK-MIB
    netSnmp                                 FROM NET-SNMP-MIB
    RowStatus, StorageType                  FROM SNMPv2-TC
    InetAddressType, InetAddress            FROM INET-ADDRESS-MIB
;

broadware MODULE-IDENTITY
    LAST-UPDATED "200701300000Z"
    ORGANIZATION "www.broadware.com"
    CONTACT-INFO
         "postal:   BroadWare Support
                    3333 Octavius Dr.
                    Santa Clara CA 95054

          email:    support@broadware.com"
    DESCRIPTION
        "Top-level infrastructure of the Broadware enterprise MIB tree"
    REVISION     "200701300000Z"
    DESCRIPTION
        "First draft"
    ::= { enterprises 28196}

events    OBJECT IDENTIFIER ::= { broadware 1 }

--
--  Broadware Notifications
--
broadwareEventNotificationPrefix  OBJECT IDENTIFIER ::= { events 1 }
broadwareEventNotifications OBJECT IDENTIFIER ::= { broadwareEventNotificationPrefix 0 }
broadwareEventNotificationObjects OBJECT IDENTIFIER ::= { broadwareEventNotificationPrefix
 1 }

--
--  Broadware Notificationi Desc
--
bwProxyEvent NOTIFICATION-TYPE
        OBJECTS   { bwEventDesc }
        STATUS    current
        DESCRIPTION
                "Notification that the proxy hosted in Broadware Media
         Server (BMS) has changed its state. Proxy is a process which maintains
         the view of a particular video cam."
::= { broadwareEventNotifications 1 }

bwArchiverEvent NOTIFICATION-TYPE
        OBJECTS   { bwEventDesc }
        STATUS    current
        DESCRIPTION
                "Notification that the archiver hosted in Broadware Media
         Server (BMS) has changed its state. Archiver stores the captured
         video information into a secondary storage device."
::= { broadwareEventNotifications 2 }
```

```
bwConnectionEvent NOTIFICATION-TYPE
        OBJECTS   { bwEventDesc }
        STATUS    current
        DESCRIPTION
                "Notification that the network connection has been lost with the
        encoder/ camera".
::= { broadwareEventNotifications 3 }

--
-- Broadware Notification Objects
--
bwEventDesc OBJECT-TYPE
        SYNTAX   SnmpAdminString
        MAX-ACCESS accessible-for-notify
        STATUS  current
        DESCRIPTION
                "This object describes the event corresponding to the notifying entity."
::= { broadwareEventNotificationObjects 1 }

END
```

GLOSSARY

A

Alarm — The action or event that triggers an alarm for which an event profile is logged. Events can be caused by an encoder with serial contact closures, a motion detected above defined thresholds, or another application using the soft-trigger command API.

Alarm Trigger — The action or event that triggers an alarm for which an event profile is logged. Events can be caused by an encoder with serial contact closures, a motion detected above defined thresholds, another application using the soft-trigger command API, or a window or door opening/closing.

Alert — The action or event that triggers an alarm for which an event profile is logged. Events can be caused by an encoder with serial contact closures, a motion detected above defined thresholds, or another application using the soft-trigger command API.

API — Application Programming Interface

Archive — A place in which records or historical documents are stored and/or preserved. An archive is a collection of video data from any given proxy source. This enables a feed from a camera-encoder to be stored in multiple locations and formats to be viewed at a later time. There are three types of archives: Regular - where the archive recording terminates after a pre-set time duration lapses and is stored for the duration of its Days-to-Live. Loop - where the archive continuously records until the archive is stopped. Loop archives reuse the space (first-in-first-out) allocated after every completion of the specified loop time. Clip - the source of the archive is extracted from one of the previous two types and is stored for the duration of its Days-to-Live.

Archive Clip — The source of the archive that is extracted from one of the other two types and stored for the duration of its Days-to-Live.

Archive Command — A URL-based API that is neither application-platform nor programming language specific. Commands are sent to dynamically loaded modules (e.g. info.bwt, command.bwt, event.bwt, &c.) using arguments in the form of name-value pairs.

Archive Server — Programs which receive incoming video streams or loops, interprets them, and takes the applicable action

Archiver — An application that manages off-line storage of video and audio onto sources such as back-up tapes, floppy disks, and optical disks.

AVI — Audio Video Interleave

AXClient — The ActiveX client name that displays the video. Each video panel can switch between multiple video streams.

B

Buffer Archive An archive used to extract event triggered clips. When an event profile includes triggered clips, an archive is started for each of the proxy sources associated with the event profile. The duration of these buffer archives includes the combined values of the pre-buffer and post-buffer times from the event profile.

C

Camera Control Permits users to change the camera lens direction and field view depth. Panning a camera moves its field of view back and forth along a horizontal <device>. Tilting commands move it up and down the vertical <device>. Zooming a camera moves objects closer to or further from the field of view. Many of these cameras also include focus and iris control. A camera may have a subset of these features such as zoom, pan, or tilt only.

Camera Drivers Responsible for converting standardized URL commands supported by the module into binary control protocols read by a specific camera model.

Child Proxy An agent, process, or function that acts as a substitute or stand-in for another. A proxy is a process that is started on a host acting as a source for a camera and encoder. This enables a single camera-encoder source to be viewed and recorded by hundreds of clients. There are three types of proxies: A direct proxy is the initial or direct connection between the edge camera-encoder source. By definition at least one direct proxy exists for a given video source. A parent proxy is the source of a nested or child proxy. Parent proxies may be from remote or local hosts. Proxies are nested in a hierarchy with inheritance rights. A child proxy is the result of a nested or parent proxy. Child proxies run on the local host. Proxies are nested in a hierarchy with inheritance rights. A child proxy has the same resolution, quality, and media type of its parent, but can have a lower frame rate for motion JPEG.

Clip A place in which records or historical documents are stored and/or preserved. An archive is a collection of video data from any given proxy source. This enables a feed from a camera-encoder to be stored in multiple locations and formats to be viewed at a later time. There are three types of archives: Regular - where the archive recording terminates after a pre-set time duration lapses and is stored for the duration of its Days-to-Live. Loop - where the archive continuously records until the archive is stopped. Loop archives reuse the space (first-in-first-out) allocated after every completion of the specified loop time. Clip - the source of the archive is extracted from one of the previous two types and is stored for the duration of its Days-to-Live.

COM Communications Port

Command API A URL-based API that is neither application-platform nor programming language specific. Commands are sent to dynamically loaded modules (e.g. info.bwt, command.bwt, event.bwt, &c.) using arguments in the form of name-value pairs.

D

Date/Time An international and universal time system. Representation of time used by computers and many programming languages are most often accurate down to the millisecond. UTC values are used to track and archive date/time values and records when events are triggered.

Days-to-Live The amount of time an archive is shelved or stopped and will remain in allocated storage. An archive that has passed its Days-to-Live is removed by a routine maintenance process.

Direct Proxy An agent, process, or function that acts as a substitute or stand-in for another. A proxy is a process that is started on a host acting as a source for a camera and encoder. This enables a single camera-encoder source to be viewed and recorded by hundreds of clients. There are three types of proxies: A direct proxy is the initial or direct connection between the edge camera-encoder source. By definition at least one direct proxy exists for a given video source. A parent proxy is the source of a nested or child proxy. Parent proxies may be from remote or local hosts. Proxies are nested in a hierarchy with inheritance rights. A child proxy is the result of a nested or parent proxy. Child proxies run on the local host. Proxies are nested in a hierarchy with inheritance rights. A child proxy has the same resolution, quality, and media type of its parent, but can have a lower frame rate for motion JPEG.

DVR Digital Video Recorder/Recording - broadcasts on a hard disk drive which can then be played back at a later time.

E

Encoder Driver Sends the output of a camera driver to the encoder to which the camera is attached (via the network protocol supported by a particular type of encoder).

Event When an incident or event occurs, it is captured by a device or application and is tagged. An event is a collection of information about an incident, including name, associated video sources, and a timestamp. If the event setup includes triggered clips, an event will have trigger tracking or video data associated directly with it. Users will need to use the event log to refer to times within a referenced archive, typically a master loop. By using the API to seek to a specific UTC timestamp, events can be used to look up occurrences in an archive that were not necessarily associated with the original event.

Event Buffer Loop An archive used to extract event triggered clips. When an event profile includes triggered clips, an archive is started for each of the proxy sources associated with the event profile. The duration of these buffer archives includes the combined values of the pre-buffer and post-buffer times from the event profile.

Event Command A URL-based API that is neither application-platform nor programming language specific. Commands are sent to dynamically loaded modules such as info.bwt, command.bwt, andevent.bwt using arguments in the form of name-value pairs.

Event Profile A collection of processes and configurations designed to track and notify when alarms or alerts are triggered. Types of event profiles includes event trigger tracking only, event triggers with archive clips, and motion detection. When an event profile includes a trigger from an encoder, part of the profile includes scripts copied to the encoder which release an event notification. When an event profile includes event triggered clips, a pre-post buffer archive is started from the proxies associated with the event profile. Once a trigger occurs, a clip is extracted from the pre-post buffer.

Event Setup A collection of processes and configurations designed to track and notify when alarms or alerts are triggered. Types of event profiles includes event trigger tracking only, event triggers with archive clips, and motion detection. When an event profile includes a trigger from an encoder, part of the profile includes scripts copied to the encoder which release an event notification. When an event profile includes event triggered clips, a pre-post buffer archive is started from the proxies associated with the event profile. Once a trigger occurs, a clip is extracted from the pre-post buffer.

Event Source Archive An archive used to extract event triggered clips. When an event profile includes triggered clips, an archive is started for each of the proxy sources associated with the event profile. The duration of these buffer archives includes the combined values of the pre-buffer and post-buffer times from the event profile.

Event Trigger The action or event that triggers an alarm for which an event profile is logged. Events can be caused by an encoder with serial contact closures, a motion detected above defined thresholds, or another application using the soft-trigger command API.

Event with Timestamp When an incident or event occurs, it is captured by a device or application and is tagged. An event is a collection of information about an incident, including name, associated video sources, and a timestamp. If the event setup includes triggered clips, an event will have trigger tracking or video data associated directly with it. Users will need to use the event log to refer to times within a referenced archive, typically a master loop. By using the API to seek to a specific UTC timestamp, events can be used to look up occurrences in an archive that were not necessarily associated with the original event.

Expiration The amount of time an archive is shelved or stopped and will remain in allocated storage. An archive that has passed its Days-to-Live is removed by a routine maintenance process

F

Feed The transmission of a video signal from point to point.

FPS Frames Per Second

Frame rate The rate at which the source is being recorded. For motion JPEG sources, the play rate is the number of frames-per-second or fps. For MPEG sources, the play rate is the number of megabits-per-second or Mbps and kilobits per second or Kbps.

Frame rate FPS An agent, process, or function that acts as a substitute or stand-in for another. A proxy is a process that is started on a host acting as a source for a camera and encoder. This enables a single camera-encoder source to be viewed and recorded by hundreds of clients. There are three types of proxies: A direct proxy is the initial or direct connection between the edge camera-encoder source. By definition at least one direct proxy exists for a given video source. A parent proxy is the source of a nested or child proxy. Parent proxies may be from remote or local hosts. Proxies are nested in a hierarchy with inheritance rights. A child proxy is the result of a nested or parent proxy. Child proxies run on the local host. Proxies are nested in a hierarchy with inheritance rights. A child proxy has the same resolution, quality, and media type of its parent, but can have a lower frame rate for motion JPEG.

G

Get Request Used to retrieve a piece of management information.

H

Hard-Trigger The action or event that triggers an alarm for which an event profile is logged. Events can be caused by an encoder with serial contact closures, a motion detected above defined thresholds, or another application using the soft-trigger command API.

HTTP Hypertext Transfer Protocol

I

IMC Interactive Media Client - Use this API for archive controls such as play, pause, and seek, and live controls such as pan, tilt, zoom, and presets.

Incident When an incident or event occurs, it is captured by a device or application and is tagged. An event is a collection of information about an incident, including name, associated video sources, and a timestamp. If the event setup includes triggered clips, an event will have trigger tracking or video data associated directly with it. Users will need to use the event log to refer to times within a referenced archive, typically a master loop. By using the API to seek to a specific timestamp, events can be used to look up occurrences in an archive that were not necessarily associated with the original event.

IP Internet Protocol

J

J2EE Java 2 Enterprise Edition

JPEG JPEG stands for Joint Photographic Experts Group, the original name of the committee that wrote the standard. JPEG is designed for compressing full color or gray-scale images of natural, real-world scenes. JPEG is lossy, meaning that the decompressed image is not exactly the same as the original. A useful property of JPEG is that the degree of lossiness can be varied by adjusting compression parameters. This means that the image maker can trade off file size against output image quality. The play rate is the number of frames per second or fps.

K

Kbps The rate at which the source is being recorded. For motion JPEG sources, the play rate is the number of frames-per-second or fps. For MPEG sources, the play rate is the number of megabits-per-second or Mbps and kilobits per second or Kbps

L

Layout The geometric description of one or more video panes.

LDAP Lightweight Directory Access Protocol

LOC Lines of Code

Logged Event When an incident or event occurs, it is captured by a device or application and is tagged. An event is a collection of information about an incident, including name, associated video sources, and a timestamp. If the event setup includes triggered clips, an event will have trigger tracking or video data associated directly with it. Users will need to use the event log to refer to times within a referenced archive, typically a master loop. By using the API to seek to a specific timestamp, events can be used to look up occurrences in an archive that were not necessarily associated with the original event.

Loop A loop is a hardware or software device which feeds the incoming signal or data back to the sender. It is used to aid in debugging physical connection problems.

M

Mask To mask or make inactive, a portion of the camera view.

Mbps The rate at which the source is being recorded. For motion JPEG sources, the play rate is the number of frames-per-second or fps. For MPEG sources, the play rate is the number of megabits-per-second or Mbps and kilobits per second or Kbps.

Media Server A device that processes multimedia applications.

MIB Management Information Base - Contains information to be monitored via Get / Set. A collection of SNMP Object Identifiers (OID) that are usually related.

MPEG MPEG stands for Moving Picture Experts Group and is the name of family of standards used for the compression of digital video and audio sequences. MPEG files are smaller for and use very sophisticated compression techniques. The play rate is the number of megabits-per-second or Mbps and kilobits per second or Kbps.

MRTG Multi-Router Resource Grapher - Used to display MIB information in graphic form.

N

NMS Network Management System. An application or suite of applications designed to monitor networks using SNMP. CiscoView is an example of NMS.

NTLM Net Technology LAN Manager

NTSC National Television System Committee

O

OID A period delimited sequence of numbers of the form a.b.c...x.y.z. A unique identifier for an item of information that is part of a MIB.

P

PAL Phase Alternating Line

Pan-Tilt-Zoom Controls Permits users to change the camera lens direction and field view depth. Panning a camera moves its field of view back and forth along a horizontal <device>. Tilting commands move it up and down the vertical <device>. Zooming a camera moves objects closer to or further from the field of view. Many of these cameras also include focus and iris control. A camera may have a subset of these features such as zoom, pan, or tilt only.

Parent Proxy An agent, process, or function that acts as a substitute or stand-in for another. A proxy is a process that is started on a host acting as a source for a camera and encoder. This enables a single camera-encoder source to be viewed and recorded by hundreds of clients. There are three types of proxies: A direct proxy is the initial or direct connection between the edge camera-encoder source. By definition at least one direct proxy exists for a given video source. A parent proxy is the source of a nested or child proxy. Parent proxies may be from remote or local hosts. Proxies are nested in a hierarchy with inheritance rights. A child proxy is the result of a nested or parent proxy. Child proxies run on the local host. Proxies are nested in a hierarchy with inheritance rights. A child proxy has the same resolution, quality, and media type of its parent, but can have a lower frame rate for motion JPEG.

PHP Hypertext Preprocessor

Play rate The rate at which the source is being recorded. For motion JPEG sources, the play rate is the number of frames-per-second or fps. For MPEG sources, the play rate is the number of megabits-per-second or Mbps and kilobits per second or Kbps.

Polling System ability to periodically monitor MIBs for changes in state.

Pre-post buffer An archive used to extract event triggered clips. When an event profile includes triggered clips, an archive is started for each of the proxy sources associated with the event profile. The duration of these buffer archives includes the combined values of the pre-buffer and post-buffer times from the event profile.

Proxy An agent, process, or function that acts as a substitute or stand-in for another. A proxy is a process that is started on a host acting as a source for a camera and encoder. This enables a single camera-encoder source to be viewed and recorded by hundreds of clients. There are three types of proxies: A direct proxy is the initial or direct connection between the edge camera-encoder source. By definition at least one direct proxy exists for a given video source. A parent proxy is the source of a nested or child proxy. Parent proxies may be from remote or local hosts. Proxies are nested in a hierarchy with inheritance rights. A child proxy is the result of a nested or parent proxy. Child proxies run on the local host. Proxies are nested in a hierarchy with inheritance rights. A child proxy has the same resolution, quality, and media type of its parent, but can have a lower frame rate for motion JPEG.

Proxy Command A URL-based API that is neither application-platform nor programming language specific. Commands are sent to dynamically loaded modules such as info.bwt, command.bwt, and event.bwt using arguments in the form of name-value pairs.

Proxy Server An agent, process, or function that acts as a substitute or stand-in for another. A proxy is a process that is started on a host acting as a source for a camera and encoder. This enables a single camera-encoder source to be viewed and recorded by hundreds of clients. There are three types of proxies: A direct proxy is the initial or direct connection between the edge camera-encoder source. By definition at least one direct proxy exists for a given video source. A parent proxy is the source of a nested or child proxy. Parent proxies may be from remote or local hosts. Proxies are nested in a hierarchy with inheritance rights. A child proxy is the result of a nested or parent proxy. Child proxies run on the local host. Proxies are nested in a hierarchy with inheritance rights. A child proxy has the same resolution, quality, and media type of its parent, but can have a lower frame rate for motion JPEG.

Proxy Source An agent, process, or function that acts as a substitute or stand-in for another. A proxy is a process that is started on a host acting as a source for a camera and encoder. This enables a single camera-encoder source to be viewed and recorded by hundreds of clients. There are three types of proxies: A direct proxy is the initial or direct connection between the edge camera-encoder source. By definition at least one direct proxy exists for a given video source. A parent proxy is the source of a nested or child proxy. Parent proxies may be from remote or local hosts. Proxies are nested in a hierarchy with inheritance rights. A child proxy is the result of a nested or parent proxy. Child proxies run on the local host. Proxies are nested in a hierarchy with inheritance rights. A child proxy has the same resolution, quality, and media type of its parent, but can have a lower frame rate for motion JPEG.

PTZ Pan Tilt Zoom - Permits users to change the camera lens direction and field view depth. Panning a camera moves its field of view back and forth along a horizontal <device>. Tilting commands move it up and down the vertical <device>. Zooming a camera moves objects closer to or further from the field of view. Many of these cameras also include focus and iris control. A camera may have a subset of these features such as zoom, pan, or tilt only.

R

Rate The rate at which the source is being recorded. For motion JPEG sources, the play rate is the number of frames-per-second or fps. For MPEG sources, the play rate is the number of megabits-per-second or Mbps and kilobits per second or Kbps.

Record Rate The rate at which the source is being recorded. For motion JPEG sources, the play rate is the number of frames-per-second or fps. For MPEG sources, the play rate is the number of megabits-per-second or Mbps and kilobits per second or Kbps.

Recording A place in which records or historical documents are stored and/or preserved. An archive is a collection of video data from any given proxy source. This enables a feed from a camera-encoder to be stored in multiple locations and formats to be viewed at a later time. There are three types of archives: Regular - where the archive recording terminates after a pre-set time duration lapses and is stored for the duration of its Days-to-Live. Loop - where the archive continuously records until the archive is stopped. Loop archives reuse the space (first-in-first-out) allocated after every completion of the specified loop time. Clip - the source of the archive is extracted from one of the previous two types and is stored for the duration of its Days-to-Live.

Recording Archive An archive whose state is running/recording. A running regular archive gathers additional data and increases in size. A running loop archive gathers more data and reuses its allocated space. Regular archives that have not reached their duration and loops that are still recording are running. Running archives have a Days-to-Live value of v-1 which does not update until they have stopped.

Repository A central place where data is stored and maintained. A repository can be a place where multiple databases or files are located for distribution over a network, or a repository can be a location that is directly accessible to the user without having to travel across a network.

Role A group of users created and setup for a specific purpose with the system such as administrators and operators.

Running Archive An archive whose state is running/recording. A running regular archive gathers additional data and increases in size. A running loop archive gathers more data and reuses its allocated space. Regular archives that have not reached their duration and loops that are still recording are running. Running archives have a Days-to-Live value of v-1 which does not update until they have stopped.

S

Scheduled Entities Events, user accounts, and roles are items controlled by a schedule.

Server-side Switch Changes the registered information for a proxy source so that the proxy process will serve multiple videos as required. Once a proxy has been updated, all requests for that proxy will be served via the new feed. All clients requesting the feeds will be switched. Proxies are not trans-coded meaning some attributes may not be changed once registered.

Server InformationCommand A URL-based API that is neither application-platform nor programming language specific. Commands are sent to dynamically loaded modules (e.g. info.bwt, command.bwt, event.bwt, &c.) using arguments in the form of name-value pairs.

Set Request Used to initialize and make a change to a value of a network element.

Shelved Archive An archive whose state is stopped. A shelved archive does not gather additional data or increase in size. Regular archives, clips, recordings, and loops that have reached their duration are considered shelved. Shelved archives are stored for the duration of their Days-to-Live.

SNMP Simple Network Management Protocol - Used to manage/monitor devices on a network using MIBS containing the information to monitor.

Soft Trigger The action or event that triggers an alarm for which an event profile is logged. Events can be caused by an encoder with serial contact closures, a motion detected above defined thresholds, or another application using the soft-trigger command API.

Stopped Archive An archive whose state has been halted. A shelved archive does not gather additional data or increase in size. Regular archives, clips, recordings, and loops that have reached their duration are considered shelved. Shelved archives are stored for the duration of their Days-to-Live.

Storage Duration The amount of time an archive is shelved or stopped and will remain in allocated storage. An archive that has passed its Days-to-Live is removed by a routine maintenance process.

Stored Archive An archive whose state is stopped. A shelved archive does not gather additional data or increase in size. Regular archives, clips, recordings, and loops that have reached their duration are considered shelved. Shelved archives are stored for the duration of their Days-to-Live.

Stream Any data transmission that occurs in a continuous flow.

SUSE A server operating system for professional deployment in IT environments of all sizes and sectors.

T

Tagged Event When an incident or event occurs, it is captured by a device or application and is tagged. An event is a collection of information about an incident, including name, associated video sources, and a timestamp. If the event setup includes triggered clips, an event will have trigger tracking or video data associated directly with it. Users will need to use the event log to refer to times within a referenced archive, typically a master loop. By using the API to seek to a specific timestamp, events can be used to look up occurrences in an archive that were not necessarily associated with the original event.

Time Stamp An international and universal time system. Representation of time used by computers and many programming languages are most often accurate down to the millisecond. UTC values are used to track archive date/time values and records when events are triggered.

Trap Used to report alerts or other asynchronous event s pertaining to a managed subsystem.

Trigger The action or event that triggers an alarm for which an event profile is logged. Events can be caused by an encoder with serial contact closures, a motion detected above defined thresholds, or another application using the soft-trigger command API.

Trigger Profile A collection of processes and configurations designed to track and notify when alarms or alerts are triggered. Types of event profiles includes event trigger tracking only, event triggers with archive clips, and motion detection. When an event profile includes a trigger from an encoder, part of the profile includes scripts copied to the encoder which release an event notification. When an event profile includes event triggered clips, a pre-post buffer archive is started from the proxies associated with the event profile. Once a trigger occurs, a clip is extracted from the pre-post buffer.

U

UI User Interface

UML Unified Modeling Language

Universal Time Code Coordinated Universal Time (UTC) - An international and universal time system. Representation of time used by computers and many programming languages are most often accurate down to the millisecond. UTC values are used to track archive date/time values and records when events are triggered.

Update Proxy Changes the registered information for a proxy source so that the proxy process will serve multiple videos as required. Once a proxy has been updated, all requests for that proxy will be served via the new feed. All clients requesting the feeds will be switched. Proxies are not trans-coded meaning some attributes may not be changed once registered.

UTC Coordinated Universal Time (UTC) - An international and universal time system. Representation of time used by computers and many programming languages are most often accurate down to the millisecond. UTC values are used to track archive date/time values and records when events are triggered.

V

Video Feed The transmission of a video signal from point to point.

View A layout, dwell time, and/or media source display.

VMR Video Mixing Renderer

VSES Video Surveillance Encoding Server

VSMS Video Surveillance Media Manager

VSOM Video Surveillance Operations Manager

VSVM Video Surveillance Virtual Matrix

W

Window All or a portion of the camera view. The display can contain multiple windows either by stacking (only the top one is entirely visible) or tiling (all are visible) or a combination of both.

WMV Windows Media Video

INDEX

Q

R

S

T

U

V

W