



Cisco Video Surveillance Manager API Reference, Release 6.3.2

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

THE SPECIFICATIONS AND INFORMATION REGARDING THE PRODUCTS IN THIS MANUAL ARE SUBJECT TO CHANGE WITHOUT NOTICE. ALL STATEMENTS, INFORMATION, AND RECOMMENDATIONS IN THIS MANUAL ARE BELIEVED TO BE ACCURATE BUT ARE PRESENTED WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED. USERS MUST TAKE FULL RESPONSIBILITY FOR THEIR APPLICATION OF ANY PRODUCTS.

THE SOFTWARE LICENSE AND LIMITED WARRANTY FOR THE ACCOMPANYING PRODUCT ARE SET FORTH IN THE INFORMATION PACKET THAT SHIPPED WITH THE PRODUCT AND ARE INCORPORATED HEREIN BY THIS REFERENCE. IF YOU ARE UNABLE TO LOCATE THE SOFTWARE LICENSE OR LIMITED WARRANTY, CONTACT YOUR CISCO REPRESENTATIVE FOR A COPY.

The Cisco implementation of TCP header compression is an adaptation of a program developed by the University of California, Berkeley (UCB) as part of UCB's public domain version of the UNIX operating system. All rights reserved. Copyright © 1981, Regents of the University of California.

NOTWITHSTANDING ANY OTHER WARRANTY HEREIN, ALL DOCUMENT FILES AND SOFTWARE OF THESE SUPPLIERS ARE PROVIDED "AS IS" WITH ALL FAULTS. CISCO AND THE ABOVE-NAMED SUPPLIERS DISCLAIM ALL WARRANTIES, EXPRESSED OR IMPLIED, INCLUDING, WITHOUT LIMITATION, THOSE OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT OR ARISING FROM A COURSE OF DEALING, USAGE, OR TRADE PRACTICE.

IN NO EVENT SHALL CISCO OR ITS SUPPLIERS BE LIABLE FOR ANY INDIRECT, SPECIAL, CONSEQUENTIAL, OR INCIDENTAL DAMAGES, INCLUDING, WITHOUT LIMITATION, LOST PROFITS OR LOSS OR DAMAGE TO DATA ARISING OUT OF THE USE OR INABILITY TO USE THIS MANUAL, EVEN IF CISCO OR ITS SUPPLIERS HAVE BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

CCDE, CCENT, Cisco Eos, Cisco Lumin, Cisco Nexus, Cisco StadiumVision, Cisco TelePresence, Cisco WebEx, the Cisco logo, DCE, and Welcome to the Human Network are trademarks; Changing the Way We Work, Live, Play, and Learn and Cisco Store are service marks; and Access Registrar, Aironet, AsyncOS, Bringing the Meeting To You, Catalyst, CCDA, CCDP, CCIE, CCIP, CCNA, CCNP, CCSP, CCVP, Cisco, the Cisco Certified Internetwork Expert logo, Cisco IOS, Cisco Press, Cisco Systems, Cisco Systems Capital, the Cisco Systems logo, Cisco Unity, Collaboration Without Limitation, EtherFast, EtherSwitch, Event Center, Fast Step, Follow Me Browsing, FormShare, GigaDrive, HomeLink, Internet Quotient, IOS, iPhone, iQuick Study, IronPort, the IronPort logo, LightStream, Linksys, MediaTone, MeetingPlace, MeetingPlace Chime Sound, MGX, Networkers, Networking Academy, Network Registrar, PCNow, PIX, PowerPanels, ProConnect, ScriptShare, SenderBase, SMARTnet, Spectrum Expert, StackWise, The Fastest Way to Increase Your Internet Quotient, TransPath, WebEx, and the WebEx logo are registered trademarks of Cisco Systems, Inc. and/or its affiliates in the United States and certain other countries.

All other trademarks mentioned in this document or website are the property of their respective owners. The use of the word partner does not imply a partnership relationship between Cisco and any other company. (0809R)

Any Internet Protocol (IP) addresses and phone numbers used in this document are not intended to be actual addresses and phone numbers. Any examples, command display output, network topology diagrams, and other figures included in the document are shown for illustrative purposes only. Any use of actual IP addresses or phone numbers in illustrative content is unintentional and coincidental.

Cisco Video Surveillance Manager API Reference, Release 6.3.2

© 2011-2013 Cisco Systems, Inc. All rights reserved.



CONTENTS

Preface ix

Audience ix

Organization ix

Obtaining Documentation and Submitting a Service Request x

CHAPTER 1

Overview 1-1

URL-Based Media Server API commands 1-1

VSMS Commands 1-1

Proxy Commands 1-2

Archive Commands 1-2

Event Commands 1-2

AxClient API Methods 1-2

Programming Notes for C# 1-3

Programing Notes for JavaScript 1-3

CHAPTER 2

VSMS Commands 2-1

Get VSMS Version 2-2

Get SDP Information for Video Stream 2-3

Camera Control 2-5

RTSP Stream from VSMS 2-13

CHAPTER 3

Proxy Commands 3-1

Start Proxy 3-2

Update Proxy 3-6

Stop Proxy 3-10

View JPEG Frames 3-11

List All Proxies 3-12

Get Proxy Source 3-14

Get Proxy Media Type 3-15

Get Proxy Frame Rate or Bit Rate 3-16

Get MJPEG Proxy Quality 3-17

Get Proxy Video Width 3-18

Get Proxy Video Height 3-19

Get Proxy Model 3-20

Text Part Number:

Get Proxy Status	3-21
Get Proxy Device	3-22
Get Proxy Frame Size	3-23

CHAPTER 4

Archive Commands	4-1
Start Archive	4-2
Update Archive	4-5
Update JPEG Archive Frame Rate	4-6
Update Archive Expiration Time	4-7
Rename Archive	4-8
Stop Archive	4-10
Remove Archive	4-11
List All Archives	4-12
List All Running Archives	4-13
Get Archive MediaType	4-14
Get Archive Details	4-15
Get Archive Recording Details	4-16
Archive Details	4-18
Get Archive Monitoring Detail	4-20
Get Archive Monitoring Summary	4-21
Create Archive Clip	4-22

CHAPTER 5

Event Commands	5-1
Event Setup	5-2
Event Setup Notification	5-7
Enable Event	5-9
Disable Event	5-10
Remove Event	5-11
Trigger VSMS Event	5-13
Event Trigger Notification	5-14
Event Clip Creation Notification	5-16
Event Clip Start/Stop	5-18
Get Event Information	5-19
Motion Event Configuration and Event Handling	5-21
Single Alarm (trigger) Event Configuration and Handling	5-22
Soft Trigger Event Configuration and Handling	5-22

CHAPTER 6

AxClient API Methods	6-1
Methods for Controlling Video Operations	6-2

mtStartStream	6-3
mtStreamStarting	6-5
mtStartStreamWait	6-7
playForward	6-8
playRewind	6-10
stepForward	6-11
stepRewind	6-12
pause	6-13
playResume	6-14
stop	6-15
close	6-16
setPlayrateEx	6-17
repeatUTCsegment	6-18
seekToUTCtime	6-19
seekToPercentage	6-20
showTimestamp	6-21
addToSync	6-22
removeFromSync	6-23
createSyncId	6-24
Methods for Obtaining Information about the AxClient or Video Streams	6-25
getCiscoHD	6-26
getVersion	6-27
getUTCseekTime	6-28
getUTCStartTime	6-29
getUTCStopTime	6-30
getUTCcurrentTime	6-31
getUTCOriginalStartTime	6-32
getState	6-33
getContentType	6-34
getCurrentSource	6-35
getRecordrateEx	6-36
getPlayrateEx	6-37
getErrorText	6-38
getProfiles	6-39
getProfilesSSV	6-40
getStreamCodecSubtype	6-41
getVideoWidth	6-42
getVideoHeight	6-43
getDisplayWidth	6-44
getDisplayHeight	6-45

getX	6-46
getY	6-47
Methods for Creating Clips and Snapshots	6-48
saveInPortableFormat	6-49
createCiscoVideoArchive	6-51
snapshot	6-53
getSnapshotDIB	6-54
getSnapshotWin32DIB	6-55
Methods for Controlling VMR Display	6-57
setAlpha	6-58
getAlpha	6-59
setTransparent	6-60
getTransparent	6-61
setBaseRectColor	6-62
getBaseRectColor	6-63
setZoomRectColor	6-64
getZoomRectColor	6-65
setTimeStampRect	6-66
setVmrDisplayMode	6-67
getVmrDisplayMode	6-68
setZoomFactor	6-69
getZoomFactor	6-70
move	6-71
deltaMove	6-72
resizeVideoWindow	6-73
Methods for Setting up Callbacks	6-75
setOnEndOfStream	6-76
setOnMtStartStreamDone	6-77
setOnPlayrateChanged	6-79
setOnSaveResponse	6-81
setOnSeekTimeChanged	6-83
setOnStartOfStream	6-84
setOnStartTimeChanged	6-85
setOnStateChanged	6-87
setOnStopTimeChanged	6-89

APPENDIX A	Turning on VMR with a C# Application	A-1
-------------------	---	------------

APPENDIX B	Supported Media Devices	B-1
-------------------	--------------------------------	------------

INDEX		
--------------	--	--



Preface

This document provides the information that is required to understand and use the Cisco Video Surveillance Manager (VSM) release 6.3.2 application programming interface (API).

Audience

This document is intended for developers who want to use the Cisco VSM API to control various Cisco VSM features and functions. It assumes that developers have knowledge or experience with Cisco VSM, C# and/or JavaScript programming languages, Hyper Text Transport Protocol (HTTP) or Secure HTTP (HTTPS), and Extensible Markup Language (XML).

Organization

This document is organized as follows:

Chapter 1, “Overview”	Provides an overview of the Cisco VSM server-side and client-side APIs.
Chapter 2, “VSMS Commands”	Provides detailed descriptions of the URL-based media server API commands that are used to retrieve information from a media server, and retrieve a Real Time Streaming Protocol (RTSP) stream for a third party video player.
Chapter 3, “Proxy Commands”	Provides detailed descriptions of the URL-based media server API commands that are used to configure and manage proxies.
Chapter 4, “Archive Commands”	Provides detailed descriptions of the URL-based media server API commands that are used to configure and manage archives.
Chapter 5, “Event Commands”	Provides detailed descriptions of the URL-based media server API commands that are used to configure and manage events.
Chapter 6, “AxClient API Methods”	Provides detailed descriptions of the AxClient API methods that are used to configure and manage a client-side video surveillance interface.

Text Part Number:

Appendix A, “Turning on VMR with a C# Application”	Describes how to edit a class file to enable VMR.
Appendix B, “Supported Media Devices”	Lists the media devices supported by Cisco VSM and describes the model, media type, transport, format, and resolution for each supported device.

Obtaining Documentation and Submitting a Service Request

For information about obtaining documentation, submitting a service request, gathering additional information, and for a list of new and revised Cisco technical documentation, see the monthly *What's New in Cisco Product Documentation* at:

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

Subscribe to the *What's New in Cisco Product Documentation* as a Really Simple Syndication (RSS) feed and set content to be delivered directly to your desktop by using a reader application. The RSS feeds are a free service and Cisco currently supports RSS Version 2.0.



CHAPTER 1

Overview

This chapter provides an overview of the Cisco VSM APIs and includes the following sections:

- [URL-Based Media Server API commands, page 1-1](#)
- [AxClient API Methods, page 1-2](#)

URL-Based Media Server API commands

Media server commands are URL-based API commands that are neither application-platform nor programming language specific. Commands are sent to dynamically loaded modules (for example, info.bwt, command.bwt, and event.bwt) on the media server using arguments in the form of name-value pairs. You can issue server API commands either manually or programmatically after an HTTP connection is established.

This section describes the four following server API command types:

- [VSMS Commands, page 1-1](#)
- [Proxy Commands, page 1-2](#)
- [Archive Commands, page 1-2](#)
- [Event Commands, page 1-2](#)

VSMS Commands

Cisco Video Surveillance Media Server (VSMS) commands perform the following media server-related functions:

- Retrieves the VSMS version
- Retrieves Session Description Protocol (SDP) information from a video stream
- Retrieves a Real Time Streaming Protocol (RTSP) stream for a third party video player
- Configures camera features, such as enabling or disabling backlight compensation or digital zoom
- Controls camera functions, such as PTZ movement, iris control, and focus

For more information on VSMS commands, see [Chapter 2, “VSMS Commands.”](#)

Text Part Number:

Proxy Commands

Proxy commands perform the following proxy-related functions:

- Starts, stops, or updates a proxy
- Views JPEG frames
- Lists all proxies
- Retrieves information about a proxy, such as its media source, media type, framerate or bitrate, video width or height, and status value.

For more information on proxy commands, see [Chapter 3, “Proxy Commands.”](#)

Archive Commands

Archive commands perform the following archive-related functions:

- Starts, stops, updates, renames, or removes an archive
- Lists all archives
- Creates an archive clip
- Retrieves information about an archive, such as its media type value, performance information, or details.

For more information on archive commands, see [Chapter 4, “Archive Commands.”](#)

Event Commands

Event commands perform the following event-related functions:

- Sets up, enables, disables, or removes an event
- Triggers an event
- Starts and stops an event-based clip
- Retrieves event information

For more information on event commands, see [Chapter 5, “Event Commands.”](#)

AxClient API Methods

VSM AxClient is an ActiveX-based API that is installed on a video surveillance client workstation, and it is used to control video operations (such as starting, pausing fast-forwarding, reversing, or stopping video streams), obtain information about AxClient or video streams, create clips and snapshots, control VMR display, and set up callbacks.

Developers can include API methods in their C# or JavaScript code to send commands to AxClient. For more information on AxClient API methods, see [Chapter 6, “AxClient API Methods.”](#)

Programming Notes for 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.

Programing Notes for 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 they 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 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.



CHAPTER 2

VSMS Commands

Table 2-1 provides a summary of the Video Surveillance Media Server (VSMS) commands. Each command is described in detail in the section that is listed.

Table 2-1 VSMS Command Summary

Name and Reference	Description
Get VSMS Version, page 2-2	Gets the VSM server version
Get SDP Information for Video Stream, page 2-3	Gets SDP information for the video stream
Camera Control, page 2-5	VSMS camera control module
RTSP Stream from VSMS, page 2-13	Gets the RTSP stream from VSMS

Get VSMS Version

http://host/info.bwt?type=version

Purpose

Retrieves the VSMS version.

Required Fields

<i>host</i>	IP address or hostname (<i>hostname.domain</i>) where VSMS is running. By default, VSMS runs on port 80 (HTTP), however, you can use an alternate port, such as port 8080. For example, to specify port 8080, use <i>host:8080</i> .
type=version	Version type. The version keyword specifies the version command type. The version keyword is a reserved value.

Return Values

A standard HTTP/1.x header followed by:

Content-Type: text/plain

Return Code: <[Server version] or -1 or output>

[Server version] Successful completion of the URL command

-1 Error in execution of the URL command

Examples

The following example retrieves the VSMS version (for example, 5.1):

http://vsms.cisco.com/info.bwt?type=version

Get SDP Information for Video Stream

`http://host/info.bwt?type=sdp&name=proxyName`

Purpose

Retrieves the Session Description Protocol (SDP) information from a video stream.

Required Fields

<i>host</i>	IP address or hostname (<i>hostname.domain</i>) where VSMS is running. By default, VSMS runs on port 80 (HTTP), however, you can use an alternate port, such as port 8080. For example, to specify port 8080, use <i>host:8080</i> .
type=sdp	SDP type. The sdp keyword specifies the SDP command type. The sdp keyword is a reserved value.
name=proxyName	Proxy name where <i>proxyName</i> specifies the name of the proxy or archive. The valid value for <i>proxyName</i> is an alphanumeric string containing 1 to 256 of the following characters: • Digits (0 to 9) • Upper case letters (A to Z) • Lower case letters (a to z) • Underscore (_) • Hyphen (-) The reserved <i>proxyName</i> value is -1. 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: application/sdp
v=0
o=- 15011761763204733780 15011761763204733810 IN IP4 10.10.40.120
s=Cisco Live Media Streaming Session
e=NONE
c=IN IP4 0.0.0.0
b=AS:15000
t=0 0
a=control:*
a=range:npt=now-
m=video 0 RTP/AVP 97
b=AS:15000
a=rtpmap:97 H264/90000
a=fmtp:97
profile-level-id=4d4028;packetization-mode=0;sprop-parameter-sets=J01AKI2NKA8ARP9gIA==
,KO48gA==;width=1920;height=1088;4CIF=1
a=x-codec:h264.cisco_hd
a=framerate:5.00
a=range:npt=now-
a=control:rtsp://10.194.66.120/live/cisco_hd

HTTP 400 Bad Request
```

Examples**Retrieving SDP Information from a Proxy**

The following command retrieves the SDP information for a proxy named ABC:

```
http://vsms.cisco.com/command.bwt?type=sdp&name=ABC
```

Retrieving SDP information from an Archive

The following command retrieves the SDP information for an archive named BCD:

```
http://vsms.cisco.com/command.bwt?type=sdp&name=BCD
```

Camera Control

**`http://host/camera.bwt?source=id@host&srctype=device&model=cameraModel&protocol=D
&comport=portNum&number=chainNum&priority=priorityNum
&command=cmdLetterOperand&button=macroName&speed=speedNum&ms=msTime`**

Purpose

Configures camera features (such as enabling or disabling backlight compensation or digital zoom) and controls camera functions (such as PTZ movement, iris control, and focus) through the network without having to know the low-level control protocols for specific cameras.

Required Fields

<i>host</i>	IP address or hostname (<i>hostname.domain</i>) where VSMS is running. By default, VSMS runs on port 80 (HTTP), however, you can use an alternate port, such as port 8080. For example, to specify port 8080, use <i>host:8080</i>.
<code>source=id@host</code>	Video source where <i>id@host</i> specifies the channel number and IP address of a video source where the <i>id</i> value can be one of the following: • Video input number of the IP camera or encoder. Valid values are 1 to 64. • Video input number and feed number (separated by an underscore) of the IP camera or encoder. This option applies only to video sources that support dual streaming. Valid input number values are 1 to 64. Valid feed number values are 1 and 2. • Name of the parent proxy. This option applies only to parent-child proxy configurations. The valid value is an alphanumeric string containing 1 to 256 of the following characters: – Digits (0 to 9) – Upper case letters (A to Z) – Lower case letters (a to z) – Underscore (_) – Hyphen (-) The reserved value is -1. For example: – 10.10.10.1 – 1@10.10.10.1 – 1_1@10.10.10.1. The <i>host</i> value is the IP address or hostname (<i>hostname.domain</i>) for the video source. You can optionally specify a port number with the IP address or hostname. For example, to specify port 8080, use <i>host:8080</i>. If no port number is specified, port 80 is used by default. Note For child proxies, the parent proxy name becomes the source.

srctype= <i>device</i>	Source type where <i>device</i> specifies the device to use as the media source for the proxy, such as a parent proxy, an encoder, or an IP camera. If the source is a parent proxy, the <i>device</i> value is proxy . For all other devices (encoders and IP cameras), the valid <i>device</i> values are listed in the Keyword column of Table B-1 in Appendix B, “Supported Media Devices.” The type of encoder or IP camera, which the module uses to select the appropriate device driver.
model= <i>modelNum</i>	The type of encoder or IP camera, which the module uses to select the appropriate device driver.
protocol=D	The particular variant of the camera control protocol for VSMS to employ. Only Pelco-D is supported. D is a reserved value.
comport= <i>portNum</i>	The encoder’s serial port to which the camera is connected. This is required for analog cameras unless a proxy is provided. COM1 and COM2 are reserved values.
number= <i>chainNum</i>	The chain number for the camera. Valid values are 0 to 64.
priority= <i>priorityNum</i>	Assigns a priority for the current command. Each time a command is sent, the priority of the command is compared to the priority of the previous command. If the priority 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. The range of valid values is 1 to 100.

Operational Fields

command=*cmdLetterOperand* Operational command. Defines a command to perform a camera control operation, such moving the camera, adjusting the camera focus, opening or closing the camera iris, and using camera position presets. An operational command is specified as a name-value pair using the following format:

command=*cmdLetterOperand*

where:

- The name is **command**.
- The value is *cmdLetterOperand*, which represents a single letter identifying the command to be executed and the operand for that command.

For example, **command=F9** specifies a command to shift the camera focus farther away by a distance of 9.

Note Only one operational command or operational button should be specified in a camera control API command. That is, you should not use multiple operational commands, multiple operational buttons, or a mixture of operational commands and buttons in the same camera control API command.

The operational command values are described in [Table 2-2](#) through [Table 2-6](#).

button=*macroName*

Operational Button. Specifies the name of a macro defined for a camera that performs a camera control operation, such as moving the camera, adjusting the camera focus, and opening or closing the camera iris. An operational button is specified as a name-value pair using the following format:

button=*macroName*

where:

- The name is **button**.
- The value is *macroName*, which represents the name of the macro that is to perform a camera control operation.

For example, **button=dzoom_on** specifies a macro that enables digital zoom.

Note Only one operational command or operational button should be specified in a camera control API command. That is, you should not use multiple operational commands, multiple operational buttons, or a mixture of operational commands and buttons in the same camera control API command.

The operational button values are described in [Table 2-2](#) through [Table 2-5](#).

speed= <i>speedNum</i>	(Optional) Sets the pan and tilt speed for momentary PTZ movement commands. The range of valid values is 1 to 100, with 1 being the slowest speed and 100 being the fastest speed.
ms= <i>msTime</i>	(Optional) Transforms a continuous PTZ movement operational command to a momentary one. After a continuous PTZ movement operational command (command= <i>_pSpeed,tSpeed,zSpeed</i>) is sent, the server waits the time specified by the <i>msTime</i> value and then automatically issues a stop movement command to the camera. The range of valid values is 20 to 20000000 milliseconds.

Table 2-2 Configuration Values

Value	Type	Description
*	Command	Passes through a low-level, camera-specific command. All camera drivers support the pass through feature, but the interpretation of the operational command depends on the driver. For example, the following command passes through the low-level command that disables backlight compensation for PTZ cameras using the PelcoD protocol: command=*00310001 Using the pass through feature is a way to use camera features that are not currently supported by VSM. To display the syntax for a particular model, find the appropriate entry in the camera PTZ XML file.
backlight_off	Button	Disables backlight compensation.
backlight_on	Button	Enables backlight compensation.
dzoom_off	Button	Disables digital zoom.
dzoom_on	Button	Enables digital zoom.
focus_auto	Button	Enables auto focus.
focus_manual	Button	Enables manual focus.
init	Button	Initializes the default PTZ control settings.
iris_auto	Button	Enables auto iris.
iris_manual	Button	Enables manual iris.
night_auto	Button	Enables auto night mode.
night_off	Button	Disables night mode.
night_on	Button	Enables night mode.
reset	Button	Resets the PTZ control settings.
wb_auto	Button	Enables auto white balance.
wb_indoor	Button	Enables indoor white balance.
wb_outdoor	Button	Enables Outdoor white balance.
wb_manual	Button	Enables Manual white balance.

Table 2-3 Focus Values

Value	Type	Description
Fdist	Command	Shifts the camera focus farther away specified by the <i>dist</i> value. The range of valid <i>dist</i> values is 0 to 9, where 0 is a short distance and 9 is a long distance.
Rdist	Command	Shifts the camera focus nearer specified by the <i>dist</i> value. The range of valid <i>dist</i> values is 0 to 9, where 0 is a short distance and 9 is a long distance.
far	Button	Focus farther away.
near	Button	Focus nearer.

Table 2-4 Iris Values

Value	Type	Description
DcloseNum	Command	Closes (dims) the camera iris specified by the <i>closeNum</i> value. The range of valid <i>closeNum</i> values is 0 to 9, where 0 is a small amount and 9 is a large amount.
EopenNum	Command	Opens (brightens) the camera iris specified by the <i>openNum</i> value. The range of valid <i>openNum</i> values is 0 to 9, where 0 is a small amount and 9 is a large amount.
bright	Button	Opens (brightens) the camera iris.
dim	Button	Closes (dims) the camera iris.

Table 2-5 PTZ Values

Value	Type	Description
Bdist	Command	Moves the camera down and left the relative distance specified by the <i>dist</i> value. The range of valid <i>dist</i> values is 1 to 360, where 1 is a short distance and 360 is a long distance.
Hdist	Command	Pans the camera left the relative distance specified by the <i>dist</i> value. The range of valid <i>dist</i> values is 1 to 360, where 1 is a short distance and 360 is a long distance.
Jdist	Command	Tilts the camera down the relative distance specified by the <i>dist</i> value. The range of valid <i>dist</i> values is 1 to 360, where 1 is a short distance and 360 is a long distance.
Kdist	Command	Tilts the camera up the relative distance specified by the <i>dist</i> value. The range of valid <i>dist</i> values is 1 to 360, where 1 is a short distance and 360 is a long distance.
Ldist	Command	Pans the camera right the relative distance specified by the <i>dist</i> value. The range of valid <i>dist</i> values is 1 to 360, where 1 is a short distance and 360 is a long distance.
Ndist	Command	Moves the camera down and right the relative distance specified by the <i>dist</i> value. The range of valid <i>dist</i> values is 1 to 360, where 1 is a short distance and 360 is a long distance.

Table 2-5 PTZ Values

Value	Type	Description
<i>Udist</i>	Command	Moves the camera up and right the relative distance specified by the <i>dist</i> value. The range of valid <i>dist</i> values is 1 to 360, where 1 is a short distance and 360 is a long distance.
<i>Wdist</i>	Command	Zooms the camera out the relative distance specified by the <i>dist</i> value. The range of valid <i>dist</i> values is 1 to 360, where 1 is a short distance and 360 is a long distance.
<i>Ydist</i>	Command	Moves the camera up and left the relative distance specified by the <i>dist</i> value. The range of valid <i>dist</i> values is 1 to 360, where 1 is a short distance and 360 is a long distance.
<i>Zdist</i>	Command	Zooms the camera in the relative distance specified by the <i>dist</i> value. The range of valid <i>dist</i> values is 1 to 360, where 1 is a short distance and 360 is a long distance.
<i>_pSpeed,tSpeed,zSpeed</i>	Command	Starts continuous PTZ movement. The camera will continue to move at the speeds specified by the <i>pSpeed</i>, <i>tSpeed</i>, and <i>zSpeed</i> values until a subsequent command is issued (unless an ms=msTime parameter is supplied with this operational command). The range of valid speed values is -100 to 100, where: - For the <i>pSpeed</i> value, negative values indicate pan left, and positive values indicate pan right. - For <i>tSpeed</i> value, negative values indicate tilt down, and positive values indicate tilt up. - For <i>zSpeed</i> value, negative values indicate zoom out, and positive values indicate zoom in. - For all three values, -100 and 100 indicate the fastest movement, and 0 indicates a stop (no movement). For example, command=_0,0,0 stops all camera PTZ movement.
down	Button	Tilts the camera down.
downleft	Button	Moves the camera down and left.
downright	Button	Moves the camera down and right.
left	Button	Pans the camera left.
right	Button	Pans the camera right.
stop	Button	Stops all PTZ movement.
tele	Button	Zooms the camera in.
up	Button	Tilts the camera the camera up.
upleft	Button	Moves the camera the camera up and left.
upright	Button	Moves the camera up and right.
wide	Button	Zooms the camera out.

Table 2-6 Presets Values

Value	Type	Description
<i>GpresetNum</i>	Command	Moves the camera to the preset position specified by the <i>presetNum</i> value. Preset numbering starts at 1. Most cameras support at least 10 presets.
<i>SpresetNum,label</i>	Command	Assigns a preset number and text label to the current camera position, as specified by the <i>presetNum</i> and <i>label</i> values. Preset numbering starts at 1. Most cameras support at least 10 presets.

Examples

Starting Continuous PTZ Movement

The following example starts a continuous PTZ movement, with pan moving left at a speed of 75, tilt moving up at a speed of 50, and zoom moving in at a speed of 25:

```
http://vsms.cisco.com/camera.bwt?source=1@192.168.1.109&srctype=sony_snc_rz30
&model=sony_snc_rz30&protocol=D&comport=COM1&number=0&priority=100&command=_-75,50,25
```

Stopping Continuous PTZ Movement

The following example stops continuous PTZ movement:

```
http://vsms.cisco.com/camera.bwt?source=1@192.168.1.109&srctype=sony_snc_rz30
&model=sony_snc_rz30&protocol=D&comport=COM1&number=0&priority=100&command=_0,0,0
```

Stopping Continuous PTZ Movement After a Specific Amount of Time

The following example stops continuous PTZ movement after 1,250 milliseconds (1.25 seconds):

```
http://vsms.cisco.com/camera.bwt?source=1@192.168.1.109&srctype=sony_snc_rz30
&model=sony_snc_rz30&protocol=D&comport=COM1&number=0&priority=100&command=_-75,50,25
&ms=1250
```

Passing Through a Low-Level Camera-Specific Command

The following example passes through the low-level command that switches to manual iris control for a PTZ camera using the PelcoD protocol:

```
http://vsms.cisco.com/camera.bwt?source=1@192.168.1.109&srctype=sony_snc_rz30
&model=sony_snc_rz30&protocol=D&comport=COM1&number=0&priority=100&command=*002D0002
```

Momentary Pan

The following example uses an operational command to perform a momentary pan left for a relative distance of 100:

```
http://vsms.cisco.com/camera.bwt?source=1@192.168.1.109&srctype=sony_snc_rz30
&model=sony_snc_rz30&protocol=D&comport=COM1&number=0&priority=100&command=H100
```

The following example uses an operational command to perform a momentary pan left for a relative distance of 100 at a speed of 80:

```
http://vsms.cisco.com/camera.bwt?source=1@192.168.1.109&srctype=sony_snc_rz30
&model=sony_snc_rz30&protocol=D&comport=COM1&number=0&priority=100&command=H100&speed=80
```

The following example uses an operational button to perform a momentary pan right:

```
http://vsms.cisco.com/camera.bwt?source=1@192.168.1.109&srctype=sony_snc_rz30
&model=sony_snc_rz30&protocol=D&comport=COM1&number=0&priority=100&button=right
```

Momentary Tilt

The following example uses an operational command to perform a momentary tilt up for a relative distance of 125:

```
http://vsms.cisco.com/camera.bwt?source=1@192.168.1.109&srctype=sony_snc_rz30
&model=sony_snc_rz30&protocol=D&comport=COM1&number=0&priority=100&command=K125
```

The following example uses an operational command to perform a momentary tilt up for a relative distance of 125 at a speed of 75:

```
http://vsms.cisco.com/camera.bwt?source=1@192.168.1.109&srctype=sony_snc_rz30
&model=sony_snc_rz30&protocol=D&comport=COM1&number=0&priority=100&command=K125&speed=75
```

The following example uses an operational button to perform a momentary tilt down:

```
http://vsms.cisco.com/camera.bwt?source=1@192.168.1.109&srctype=sony_snc_rz30
&model=sony_snc_rz30&protocol=D&comport=COM1&number=0&priority=100&button=down
```

Momentary Zoom

The following example uses an operational command to perform a momentary zoom out for a relative distance of 35:

```
http://vsms.cisco.com/camera.bwt?source=1@192.168.1.109&srctype=sony_snc_rz30
&model=sony_snc_rz30&protocol=D&comport=COM1&number=0&priority=100&command=W35
```

The following example uses an operational command to perform a momentary zoom out for a relative distance of 35 at a speed of 50:

```
http://vsms.cisco.com/camera.bwt?source=1@192.168.1.109&srctype=sony_snc_rz30
&model=sony_snc_rz30&protocol=D&comport=COM1&number=0&priority=100&command=W35&speed=50
```

The following example uses an operational button to perform a momentary zoom in:

```
http://vsms.cisco.com/camera.bwt?source=1@192.168.1.109&srctype=sony_snc_rz30
&model=sony_snc_rz30&protocol=D&comport=COM1&number=0&priority=100&button=tele
```

Assigning a Preset

The following example assigns the current PTZ position to preset 1 and labels the preset as Front_Door:

```
http://vsms.cisco.com/camera.bwt?source=1@192.168.1.109&srctype=sony_snc_rz30
&model=sony_snc_rz30&protocol=D&comport=COM1&number=0&priority=100&command=S1,Front_Door
```

Going to a Preset

The following example moves the camera to PTZ position preset 5:

```
http://vsms.cisco.com/camera.bwt?source=1@192.168.1.109&srctype=sony_snc_rz30
&model=sony_snc_rz30&protocol=D&comport=COM1&number=0&priority=100&command=G5
```

RTSP Stream from VSMS

rtsp://host/live/proxyName

rtsp://host/archive/archiveName

Purpose

Retrieves a Real Time Streaming Protocol (RTSP) stream from a proxy or archive for a third party video player.

Required Fields

<i>host</i>	IP address or hostname (<i>hostname.domain</i>) where VSMS is running. By default, VSMS runs on port 80 (HTTP), however, you can use an alternate port, such as port 8080. For example, to specify port 8080, use <i>host:8080</i> .
<i>live/proxyName</i>	The name of the proxy. The valid <i>proxyName</i> value is an alphanumeric string containing 1 to 256 of the following characters: - Digits (0 to 9) - Upper case letters (A to Z) - Lower case letters (a to z) - Underscore (_) - Hyphen (-) The reserved value is -1.
<i>archive/archiveName</i>	The name of the archive. The valid <i>archiveName</i> value is an alphanumeric string containing 1 to 256 of the following characters: - Digits (0 to 9) - Upper case letters (A to Z) - Lower case letters (a to z) - Underscore (_) - Hyphen (-) The reserved value is -1.

Examples

Retrieving an RTSP Stream from a Proxy

The following command retrieves an RTSP stream from proxy ABC for a 3rd party video player (such as VLC Player):

```
rtsp://vsms.cisco.com/live/ABC
```

Retrieving an RTSP Stream from an Archive

The following command retrieves an RTSP stream from archive BCD for a 3rd party video player (such as VLC Player):

```
rtsp://vsms.cisco.com/archive/BCD
```




CHAPTER 3

Proxy Commands

Table 3-1 provides a summary of the proxy commands. Each command is described in detail in the section that is listed.

Table 3-1 **Proxy Command Summary**

Name and Reference	Description
Start Proxy, page 3-2	Starts a proxy process on a media server that establishes a connection to a media source and writes media data to shared memory.
Update Proxy, page 3-6	Updates an existing proxy with different parameter values.
Stop Proxy, page 3-10	Stops a running proxy and all archives accessing the media data written to shared memory by the proxy.
View JPEG Frames, page 3-11	Displays JPEG frames from a proxy.
List All Proxies, page 3-12	Displays a list of all proxies running on a VSMS host.
Get Proxy Source, page 3-14	Retrieves the media source value for a proxy.
Get Proxy Media Type, page 3-15	Retrieves the media type value for a proxy.
Get Proxy Frame Rate or Bit Rate, page 3-16	Retrieves the frame rate value for a JPEG proxy or the bit rate value for an MPEG or audio proxy.
Get MJPEG Proxy Quality, page 3-17	Retrieves the quality value for a JPEG proxy.
Get Proxy Video Width, page 3-18	Retrieves the width in pixels for a video proxy.
Get Proxy Video Height, page 3-19	Retrieves the height in pixels for a video proxy.
Get Proxy Model, page 3-20	Retrieves the model ID value for a proxy media source.
Get Proxy Status, page 3-21	Retrieves the status value for a proxy.

Start Proxy

```
http://host/command.bwt?command=start&type=proxy&name=proxyName
&source=id@host&srctype=device&mediatype=codec
&quality=qualNum&framerate=rateNum&width=widthNum&height=heightNum
&bitrate=rateNum&username=name&password=password
&resolution=resVal&format=format&udp=udpNum&multicast=multicastIPAddress
```

Purpose Starts a proxy process on a media server that establishes a connection to a media source (such as an IP camera, an encoder, or another proxy) and writes media data to shared memory.

Required Fields	<i>host</i>	IP address or hostname (<i>hostname.domain</i>) where VSMS is running. By default, VSMS runs on port 80 (HTTP), however, you can use an alternate port, such as port 8080. For example, to specify port 8080, use <i>host:8080</i> .
	command=start	Start command. The start keyword associates the command with a start action, in this case a start proxy action. The start keyword is a reserved value.
	type=proxy	Proxy type. The proxy keyword specifies the proxy command type. The proxy keyword is a reserved value.
	name=proxyName	Proxy name where <i>proxyName</i> specifies the name of the proxy instance and the name of the source for a child proxy. The valid value for <i>proxyName</i> is an alphanumeric string containing 1 to 256 of the following characters: • Digits (0 to 9) • Upper case letters (A to Z) • Lower case letters (a to z) • Underscore (_) • Hyphen (-) The reserved <i>proxyName</i> value is -1.
	Note Each proxy must have a unique name on a given VSMS host.	

source=id@host	Video source where <i>id@host</i> specifies the channel number and IP address of a video source. The <i>id</i> value can be one of the following: • Video input number of the IP camera or encoder. Valid values are 1 to 64. • Video input number and feed number (separated by an underscore) of the IP camera or encoder. This option applies only to video sources that support dual streaming. Valid input number values are 1 to 64. Valid feed number values are 1 and 2. • Name of the parent proxy. This option applies only to parent-child proxy configurations. The valid value is an alphanumeric string containing 1 to 256 of the following characters: – Digits (0 to 9) – Upper case letters (A to Z) – Lower case letters (a to z) – Underscore (_) – Hyphen (-) The reserved value is -1. For example: – 10.10.10.1 – 1@10.10.10.1 – 1_1@10.10.10.1. The <i>host</i> value is the IP address or hostname (<i>hostname.domain</i>) for the video source. You can optionally specify a port number with the IP address or hostname. For example, to specify port 8080, use <i>host:8080</i>. If no port number is specified, port 80 is used by default. Note For child proxies, the parent proxy name becomes the source.
srctype=device	Source type where <i>device</i> specifies the device to use as the media source for the proxy, such as a parent proxy, an encoder, or an IP camera. If the source is a parent proxy, the <i>device</i> value is proxy. For all other devices (encoders and IP cameras), the valid <i>device</i> values are listed in the Keyword column of Table B-1 in Appendix B, “Supported Media Devices.”
mediatype=codec	Media type where <i>codec</i> specifies the codec used to encode the media. Valid <i>codec</i> values include the following reserved keywords: • mjpeg • mpeg2 • mpeg4 • h264 Note For more information about the supported devices for each media type, see Appendix B, “Supported Media Devices.”

Optional Fields

quality= <i>qualNum</i>	Quality of the feed where <i>qualNum</i> specifies the compression ratio (the quality of the JPEG feed), which depends on the device and media type. 200 is the highest possible quality. The range of valid values for the <i>qualNum</i> value is 1 to 200; the default value is 50.
framerate= <i>rateNum</i>	Frame rate where <i>rateNum</i> specifies the maximum number of MJPEG frames transmitted per second. The default value is 5. Note A child proxy cannot have a higher frame rate value than the parent proxy.
width= <i>widthNum</i>	Width where <i>widthNum</i> specifies the width of the video feed in pixels, as supported by the device.
height= <i>heightNum</i>	Height where <i>heightNum</i> specifies the height of the video feed in pixels, as supported by the device.
bitrate= <i>rateNum</i>	Bit rate where <i>rateNum</i> specifies the Kilobytes transmitted per second for a video feed (applicable to non-MJPEG feeds). The minimum, maximum, and actual bit rates are device-dependent.
username= <i>name</i>	Username where <i>name</i> is the user name for an administrative user account of an encoding device, if required for authentication.
password= <i>password</i>	Password where <i>password</i> is the password for an administrative user account of an encoding device, if required for authentication.
resolution= <i>resVal</i>	Resolution where <i>resVal</i> specifies a predefined video width and height to use for the proxy, from the device.xml file. Valid <i>resVal</i> values include the following reserved keywords: • cif—Common intermediate format (CIF). For example, 352 x 240 for NTSC or 352 x 288 for PAL. • qcif—Quarter CIF. For example, 176 x 120 for NTSC or 176 x 144 for PAL. • 2cif—2 x CIF. For example, 704 x 240 for NTSC or 704 x 288 for PAL. • 4cif—4 x CIF. For example, 704 x 480 for NTSC or 704 x 576 for PAL. • d1—Sony D-1. For example, 720 x 480 for NTSC or 720 x 576 for PAL. • 1M—1 megapixel. For example, 1024 x 768 for NTSC. • 2M—2 megapixel. For example, 1280 x 960 for NTSC. • 3M—3 megapixel. For example, 1280 x 1024 for NTSC. • 4M—4 megapixel. For example, 2272 x 1704 for NTSC. • 5M—5 megapixel. For example, 2560 x 1920 for NTSC.
format= <i>format</i>	Format where <i>format</i> specifies the video standard to use when determining the width and height of the video. Valid <i>format</i> values include the following reserved keywords: • ntsc—Used mostly in the United States, Canada, and portions of South America and specifies 525 lines per frame with a 60 Hz refresh rate. • pal—Used mostly in Europe, China, and Australia and specifies 625 lines per frame with a 50 Hz refresh rate.

udp=udpNum	Transport protocol where <i>udpNum</i> enables the TCP or UDP protocol for streams. Valid <i>udpNum</i> values are 0 (TCP) and 1 (UDP).
multicast=multicastIP Address	Multicast host name where <i>multicastIPAddress</i> is the IP address or hostname (<i>hostname.domain</i>) of the device originating the multicast stream. To start or join a multicast stream, specify the multicast group name.

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
```

Usage Guidelines

The video source can be an IP camera, an encoding device, or another proxy. Parent-child proxies can be nested indefinitely as resources permit.

Some encoding devices may not support all available bit rate, frame rate, resolution, or quality settings. Be sure to use settings supported by your encoding device. For more information about the supported settings for the encoding device, see [Appendix B, “Supported Media Devices.”](#)

Examples

The following example starts a video proxy using the required fields. The proxy named basicProxy runs on the host named vsms.cisco.com, the video source is video input 1 of the device with IP address 192.168.200.20:

```
http://vsms.cisco.com/command.bwt?command=start&type=proxy&name=basicProxy
&srctype=c&type=jpeg&source=1@192.168.200.20
```

Update Proxy

```
http://host/command.bwt?command=update&type=proxy&name=proxyName
&source=id@host&srctype=device&mediatype=codec&framerate=rateNum
&width=widthNum&height=heightNum&bitrate=rateNum&username=name
&password=password&resolution=resVal&format=format
```

Purpose

Updates an existing proxy with different parameter values. For example, updating a proxy to a different source seamlessly changes the feed being viewed by all clients.



Caution

If an archive is accessing the media data written to shared memory by 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.

Required Fields

host	IP address or hostname (<i>hostname.domain</i>) where VSMS is running. By default, VSMS runs on port 80 (HTTP), however, you can use an alternate port, such as port 8080. For example, to specify port 8080, use <i>host:8080</i> .
command=update	Update command. The update keyword associates the command with an update action. The update keyword is a reserved value.
type=proxy	Proxy type. The proxy keyword specifies the proxy command type. The proxy keyword is a reserved value.
name=proxyName	Proxy name where <i>proxyName</i> specifies the name of the proxy instance and the name of the source for a child proxy. The valid value for <i>proxyName</i> is an alphanumeric string containing 1 to 256 of the following characters: • Digits (0 to 9) • Upper case letters (A to Z) • Lower case letters (a to z) • Underscore (_) • Hyphen (-) The reserved <i>proxyName</i> value is -1. Note Each proxy must have a unique name on a given VSMS host.

source=id@host	Video source where <i>id@host</i> specifies the channel number and IP address of a video source. The <i>id</i> value can be one of the following: • Video input number of the IP camera or encoder. Valid values are 1 to 64. • Video input number and feed number (separated by an underscore) of the IP camera or encoder. This option applies only to video sources that support dual streaming. Valid input number values are 1 to 64. Valid feed number values are 1 and 2. • Name of the parent proxy. This option applies only to parent-child proxy configurations. The valid value is an alphanumeric string containing 1 to 256 of the following characters: – Digits (0 to 9) – Upper case letters (A to Z) – Lower case letters (a to z) – Underscore (_) – Hyphen (-) The reserved value is -1. For example: – 10.10.10.1 – 1@10.10.10.1 – 1_1@10.10.10.1. The <i>host</i> value is the IP address or hostname (<i>hostname.domain</i>) for the video source. You can optionally specify a port number with the IP address or hostname. For example, to specify port 8080, use <i>host:8080</i>. If no port number is specified, port 80 is used by default. Note For child proxies, the parent proxy name becomes the source.
srctype=device	Source type where <i>device</i> specifies the device to use as the media source for the proxy, such as a parent proxy, an encoder, or an IP camera. If the source is a parent proxy, the <i>device</i> value is proxy. For all other devices (encoders and IP cameras), the valid <i>device</i> values are listed in the Keyword column of Table B-1 in Appendix B, “Supported Media Devices.”
mediatype=codec	Media type where <i>codec</i> specifies the codec used to encode the media. Valid <i>codec</i> values include the following reserved keywords: • mjpeg • mpeg2 • mpeg4 • h264 Note For more information about the supported devices for each media type, see Appendix B, “Supported Media Devices.”
framerate=rateNum	Frame rate where <i>rateNum</i> specifies the maximum number of MJPEG frames transmitted per second. The default value is 5. Note A child proxy cannot have a higher frame rate value than the parent proxy.

width= <i>widthNum</i>	Width where <i>widthNum</i> specifies the width of the video feed in pixels, as supported by the device.
height= <i>heightNum</i>	Height where <i>heightNum</i> specifies the height of the video feed in pixels, as supported by the device.
bitrate= <i>rateNum</i>	Bit rate where <i>rateNum</i> specifies the Kilobytes transmitted per second for a video feed (applicable to non-MJPEG feeds). The minimum, maximum, and actual bit rates are device-dependent.
username= <i>name</i>	Username where <i>name</i> is the user name for an administrative user account of an encoding device, if required for authentication.
password= <i>password</i>	Password where <i>password</i> is the password for an administrative user account of an encoding device, if required for authentication.
resolution= <i>resVal</i>	Resolution where <i>resVal</i> specifies a predefined video width and height to use for the proxy, from the device.xml file. Valid <i>resVal</i> values include the following reserved keywords: • cif—Common intermediate format (CIF). For example, 352 x 240 for NTSC or 352 x 288 for PAL. • qcif—Quarter CIF. For example, 176 x 120 for NTSC or 176 x 144 for PAL. • 2cif—2 x CIF. For example, 704 x 240 for NTSC or 704 x 288 for PAL. • 4cif—4 x CIF. For example, 704 x 480 for NTSC or 704 x 576 for PAL. • d1—Sony D-1. For example, 720 x 480 for NTSC or 720 x 576 for PAL. • 1M—1 megapixel. For example, 1024 x 768 for NTSC. • 2M—2 megapixel. For example, 1280 x 960 for NTSC. • 3M—3 megapixel. For example, 1280 x 1024 for NTSC. • 4M—4 megapixel. For example, 2272 x 1704 for NTSC. • 5M—5 megapixel. For example, 2560 x 1920 for NTSC.
format= <i>format</i>	Format where <i>format</i> specifies the video standard to use when determining the width and height of the video. Valid <i>format</i> values include the following reserved keywords: • ntsc—Used mostly in the United States, Canada, and portions of South America and specifies 525 lines per frame with a 60 Hz refresh rate. • pal—Used mostly in Europe, China, and Australia and specifies 625 lines per frame with a 50 Hz refresh rate.

Optional Fields

username= <i>name</i>	Username where the <i>name</i> value is the user name for an administrative user account of an encoding device. Note The <i>name</i> value is required to do Camera Controls and Events Setup, and when user authentication is enabled for a device.
password= <i>password</i>	Password where the <i>password</i> value is the password for an administrative user account of an encoding device. Note The <i>password</i> value is required to do Camera Controls and Events Setup, and when user authentication is enabled for a device.
format= <i>format</i>	Format where the <i>format</i> value specifies the video standard to use when determining the resolution of the video. Valid <i>format</i> values include the following reserved keywords: • ntsc—Used mostly in the United States, Canada, and portions of South America and specifies 525 lines per frame with a 60 Hz refresh rate. • pal—Used mostly in Europe, China, and Australia and specifies 625 lines per frame with a 50 Hz refresh rate. Note The actual video size is determined by the resolution , format, and device driver settings for the source type (unless you specify the width and height). The default <i>format</i> value is ntsc .

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
```

Examples

The following example updates the proxy named Front_Office on vsms.cisco.com to use a frame rate of 10 frames per second:

```
http://vsms.cisco.com/command.bwt?command=update&source=1@camera.cisco.com&type=proxy&srct
ype=proxy&name=Front_Office&framerate=10
```

Stop Proxy

```
http://host/command.bwt?command=stop&type=proxy&name=proxyName
```

Purpose

Stops a running proxy and all archives accessing the media data written to shared memory by the proxy.

Required Fields	host	IP address or hostname (<i>hostname.domain</i>) where VSMS is running. By default, VSMS runs on port 80 (HTTP), however, you can use an alternate port, such as port 8080. For example, to specify port 8080, use <i>host:8080</i> .
	command=stop	Stop command. The stop keyword associates the command with a stop action. The stop keyword is a reserved value.
	type=proxy	Proxy type. The proxy keyword specifies the proxy command type. The proxy keyword is a reserved value.
	name=proxyName	Proxy name where <i>proxyName</i> specifies the name of the proxy instance and the name of the source for a child proxy. The valid value for <i>proxyName</i> is an alphanumeric string containing 1 to 256 of the following characters: • Digits (0 to 9) • Upper case letters (A to Z) • Lower case letters (a to z) • Underscore (_) • Hyphen (-) The reserved <i>proxyName</i> value is -1.
	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
```

Examples

The following example stops the proxy named officeCam and any archives running against that proxy on the VSMS host.

```
http://vsms.cisco.com/command.bwt?command=stop&type=proxy&name=officeCam
```

View JPEG Frames

http://host/video.jpg?source=proxyName&framerate=rateNum

Purpose

Displays JPEG frames from a proxy. This command only applies to MJPEG video streams.

Required Fields

<i>host</i>	IP address or hostname (<i>hostname.domain</i>) where VSMS is running. By default, VSMS runs on port 80 (HTTP), however, you can use an alternate port, such as port 8080. For example, to specify port 8080, use <i>host:8080</i> .
source=proxyName	Source proxy where <i>proxyName</i> specifies the proxy name for the MJPEG media source. The valid value is an alphanumeric string containing 1 to 256 of the following characters: – Digits (0 to 9) – Upper case letters (A to Z) – Lower case letters (a to z) – Underscore (_) – Hyphen (-) The reserved value is -1.
framerate=rateNum	Frame rate where <i>rateNum</i> specifies the maximum number of MJPEG frames transmitted per second. The default value is 5.

Examples

The following example views the JPEG frames from the MJPEG proxy named officeCam.

http://vsms.cisco.com/video.jpg?source=officeCam&framerate=0

List All Proxies

http://host/info.bwt?type=proxy&name=proxyName&display=dispFormat

Purpose	Displays a list of all proxies running on a VSMS host.	
Required Fields	<i>host</i>	IP address or hostname (<i>hostname.domain</i>) where VSMS is running. By default, VSMS runs on port 80 (HTTP), however, you can use an alternate port, such as port 8080. For example, to specify port 8080, use <i>host:8080</i> .
	type=proxy	Proxy type. The proxy keyword specifies the proxy command type. The proxy keyword is a reserved value.
Optional Fields	name=proxyName	Proxy name where <i>proxyName</i> specifies the name of the proxy instance and the name of the source for a child proxy. The valid value for <i>proxyName</i> is an alphanumeric string containing 1 to 256 of the following characters: • Digits (0 to 9) • Upper case letters (A to Z) • Lower case letters (a to z) • Underscore (_) • Hyphen (-) The reserved <i>proxyName</i> value is -1. Note Each proxy must have a unique name on a given VSMS host.
	display=dispFormat	Displayformat where <i>dispFormat</i> specifies the format to use when displaying the list of running proxies. Valid <i>dispFormat</i> values include the following keywords: • html—Hypertext markup language format • text—Plain text format • ssv—Space-separated value format The default <i>dispFormat</i> value is html .
Return Values	A standard HTTP/1.x header followed by: Content-Type: text/plain Return Code: <[Proxy List] or -1 or output> [Proxy List] Successful completion of the URL command -1 Error in execution of the URL command	

Examples**Listing All Running Proxies**

The following example retrieves a list in SSV format of all running proxies on the host named vsms.cisco.com:

```
http://vsms.cisco.com/info.bwt?type=proxy&display=ssv
```

Listing A Single Proxy

The following example retrieves the details in TEXT format of a proxy named officeCam running proxies on the host named vsms.cisco.com:

```
http://vsms.cisco.com/info.bwt?type=proxy&name=officeCam&display=text
```

Get Proxy Source

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

Purpose

Retrieves device information for a proxy. The return value is in plain text format can be in one of the following media sources:

- Video input number of an IP camera or encoder
- Video input number and feed number (separated by an underscore) of an IP camera or encoder
- Name of a parent proxy

Required Fields

<i>host</i>	IP address or hostname (<i>hostname.domain</i>) where VSMS is running. By default, VSMS runs on port 80 (HTTP), however, you can use an alternate port, such as port 8080. For example, to specify port 8080, use <i>host:8080</i> .
type=proxy	Proxy type. The proxy keyword specifies the proxy command type. The proxy keyword is a reserved value.
name=proxyName	Proxy name where <i>proxyName</i> specifies the name of the proxy instance and the name of the source for a child proxy. The valid value for <i>proxyName</i> is an alphanumeric string containing 1 to 256 of the following characters: • Digits (0 to 9) • Upper case letters (A to Z) • Lower case letters (a to z) • Underscore (_) • Hyphen (-) The reserved <i>proxyName</i> value is -1. Note Each proxy must have a unique name on a given VSMS host.
property=source	Source property. The source keyword requests the source value for the specified proxy. The source keyword is a reserved value.

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
```

Examples

The following example retrieves device information for proxy 1036:

```
http://vsms.cisco.com/info.bwt?type=proxy&name=1036&property=source
```

Get Proxy Media Type

http://host/info.bwt?type=proxy&name=proxyName&property=mediatype

Purpose

Retrieves the encoding media type value for a proxy.

Required Fields

<i>host</i>	IP address or hostname (<i>hostname.domain</i>) where VSMS is running. By default, VSMS runs on port 80 (HTTP), however, you can use an alternate port, such as port 8080. For example, to specify port 8080, use <i>host:8080</i> .
type=proxy	Proxy type. The proxy keyword specifies the proxy command type. The proxy keyword is a reserved value.
name=proxyName	Proxy name where <i>proxyName</i> specifies the name of the proxy instance and the name of the source for a child proxy. The valid value for <i>proxyName</i> is an alphanumeric string containing 1 to 256 of the following characters: • Digits (0 to 9) • Upper case letters (A to Z) • Lower case letters (a to z) • Underscore (_) • Hyphen (-) The reserved <i>proxyName</i> value is -1. Note Each proxy must have a unique name on a given VSMS host.
property=mediatype	Media type property. The mediatype keyword requests the media type value for the specified proxy. The mediatype keyword is a reserved value.

Return Values

A standard HTTP/1.x header followed by:

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

Examples

The following example retrieves the encoding media type value for the proxy named officeCam:

```
http://vsms.cisco.com/info.bwt?type=proxy&name=officeCam&property=mediatype
```

Get Proxy Frame Rate or Bit Rate

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

Purpose	Retrieves the frame rate value for a JPEG proxy or the bit rate value for an MPEG proxy. The return value is in plain text format.	
Required Fields	host	IP address or hostname (<i>hostname.domain</i>) where VSMS is running. By default, VSMS runs on port 80 (HTTP), however, you can use an alternate port, such as port 8080. For example, to specify port 8080, use <i>host:8080</i> .
	type=proxy	Proxy type. The proxy keyword specifies the proxy command type. The proxy keyword is a reserved value.
	name=proxyName	Proxy name where <i>proxyName</i> specifies the name of the proxy instance and the name of the source for a child proxy. The valid value for <i>proxyName</i> is an alphanumeric string containing 1 to 256 of the following characters: • Digits (0 to 9) • Upper case letters (A to Z) • Lower case letters (a to z) • Underscore (_) • Hyphen (-) The reserved <i>proxyName</i> value is -1. Note Each proxy must have a unique name on a given VSMS host.
	property=rate	Rate property. The rate keyword requests the the frame rate or bit rate value for the specified proxy. The rate keyword is a reserved value.

Return Values	A standard HTTP/1.x header followed by: <pre>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</pre>
---------------	---

Examples	The following example retrieves the bit rate or frame rate value, which depends on the encoding media type (MJPEG or MPEG), for the proxy named officeCam: <pre>http://vsms.cisco.com/info.bwt?type=proxy&name=officeCam&property=rate</pre>
----------	---

Get MJPEG Proxy Quality

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

Purpose

Retrieves the quality value for an MJPEG proxy. The return value is in plain text format.

Required Fields

host	IP address or hostname (<i>hostname.domain</i>) where VSMS is running. By default, VSMS runs on port 80 (HTTP), however, you can use an alternate port, such as port 8080. For example, to specify port 8080, use <i>host:8080</i> .
type=proxy	Proxy type. The proxy keyword specifies the proxy command type. The proxy keyword is a reserved value.
name=proxyName	Proxy name where <i>proxyName</i> specifies the name of the proxy instance and the name of the source for a child proxy. The valid value for <i>proxyName</i> is an alphanumeric string containing 1 to 256 of the following characters: • Digits (0 to 9) • Upper case letters (A to Z) • Lower case letters (a to z) • Underscore (_) • Hyphen (-) The reserved <i>proxyName</i> value is -1. Note Each proxy must have a unique name on a given VSMS host.
property=quality	Quality property. The quality keyword requests the quality value for the MJPEG proxy. The quality keyword is a reserved value. Note This command only works for MJPEG proxies. For other media type proxies, an error is returned (-1).

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>
```

Examples

The following example retrieves the quality value for the proxy named testProxy:

```
http://vsms.cisco.com/info.bwt?type=proxy&name=testProxy&property=quality
```

Get Proxy Video Width

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

Purpose	Retrieves the width in pixels for a video proxy. The return value is in plain text format.	
Required Fields	host	IP address or hostname (<i>hostname.domain</i>) where VSMS is running. By default, VSMS runs on port 80 (HTTP), however, you can use an alternate port, such as port 8080. For example, to specify port 8080, use <i>host:8080</i> .
	type=proxy	Proxy type. The proxy keyword specifies the proxy command type. The proxy keyword is a reserved value.
	name=proxyName	Proxy name where <i>proxyName</i> specifies the name of the proxy instance and the name of the source for a child proxy. The valid value for <i>proxyName</i> is an alphanumeric string containing 1 to 256 of the following characters: • Digits (0 to 9) • Upper case letters (A to Z) • Lower case letters (a to z) • Underscore (_) • Hyphen (-) The reserved <i>proxyName</i> value is -1. Note Each proxy must have a unique name on a given VSMS host.
	property=width	Width property. The width keyword requests the width in pixels of the video stream. The width keyword is a reserved value.

Return Values	A standard HTTP/1.x header followed by: Content-Type: text/plain Return Code: <[Pixel width] or -1 or output> [Pixel width] Successful completion of the URL command -1 Error in execution of the URL command
---------------	---

Examples	The following example retrieves the video width in pixels for the proxy named officeCam: http://vsms.cisco.com/info.bwt?type=proxy&name=officeCam&property=width
----------	---

Get Proxy Video Height

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

Purpose

Retrieves the height in pixels for a video proxy. The return value is in plain text format.

Required Fields

<code>host</code>	IP address or hostname (<i>hostname.domain</i>) where VSMS is running. By default, VSMS runs on port 80 (HTTP), however, you can use an alternate port, such as port 8080. For example, to specify port 8080, use <code>host:8080</code> .
<code>type=proxy</code>	Proxy type. The <code>proxy</code> keyword specifies the proxy command type. The <code>proxy</code> keyword is a reserved value.
<code>name=proxyName</code>	Proxy name where <i>proxyName</i> specifies the name of the proxy instance and the name of the source for a child proxy. The valid value for <i>proxyName</i> is an alphanumeric string containing 1 to 256 of the following characters: • Digits (0 to 9) • Upper case letters (A to Z) • Lower case letters (a to z) • Underscore (_) • Hyphen (-) The reserved <i>proxyName</i> value is -1. Note Each proxy must have a unique name on a given VSMS host.
<code>property=height</code>	Height property. The <code>height</code> keyword requests the height in pixels of the video stream. The <code>height</code> keyword is a reserved value.

Return Values

A standard HTTP/1.x header followed by:

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

Examples

The following example retrieves the video height in pixels for proxy 1036:

```
http://vsms.cisco.com/info.bwt?type=proxy&name=1036&property=height
```

Get Proxy Model

http://host/info.bwt?type=proxy&name=proxyName&property=model

Purpose	Retrieves the model number for a proxy media source. The return value is in plain text format.	
Required Fields	host	IP address or hostname (<i>hostname.domain</i>) where VSMS is running. By default, VSMS runs on port 80 (HTTP), however, you can use an alternate port, such as port 8080. For example, to specify port 8080, use <i>host:8080</i> .
	type=proxy	Proxy type. The proxy keyword specifies the proxy command type. The proxy keyword is a reserved value.
	name=proxyName	Proxy name where <i>proxyName</i> specifies the name of the proxy instance and the name of the source for a child proxy. The valid value for <i>proxyName</i> is an alphanumeric string containing 1 to 256 of the following characters: • Digits (0 to 9) • Upper case letters (A to Z) • Lower case letters (a to z) • Underscore (_) • Hyphen (-) The reserved <i>proxyName</i> value is -1. Note Each proxy must have a unique name on a given VSMS host.
	property=model	Model property. The model keyword requests the model number of the device for the specified proxy. The model number is the unique proxy ID associated with the device. The model keyword is a reserved value.

Return Values	A standard HTTP/1.x header followed by: Content-Type: text/plain Return Code: <[1-200] or -1 or output> [1-200] Successful completion of the URL command -1 Error in execution of the URL command
---------------	---

Examples	The following example retrieves the model number for proxy 1036: http://vsms.cisco.com/info.bwt?type=proxy&name=1036&property=model
----------	--

Get Proxy Status

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

Purpose

Retrieves the status (running, stopped, or suspended) for a proxy. The return value is in plain text format and can be one of the following reserved keywords:

- **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

Required Fields

<i>host</i>	IP address or hostname (<i>hostname.domain</i>) where VSMS is running. By default, VSMS runs on port 80 (HTTP), however, you can use an alternate port, such as port 8080. For example, to specify port 8080, use <i>host:8080</i> .
type=proxy	Proxy type. The proxy keyword specifies the proxy command type. The proxy keyword is a reserved value.
name=proxyName	Proxy name where <i>proxyName</i> specifies the name of the proxy instance and the name of the source for a child proxy. The valid value for <i>proxyName</i> is an alphanumeric string containing 1 to 256 of the following characters: • Digits (0 to 9) • Upper case letters (A to Z) • Lower case letters (a to z) • Underscore (_) • Hyphen (-) The reserved <i>proxyName</i> value is -1. Note Each proxy must have a unique name on a given VSMS host.
property=status	Status property. The status keyword requests the current status of the specified proxy. The status keyword is a reserved value.

Return Values

A standard HTTP/1.x header followed by:

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

Examples

The following example retrieves the status for proxy 1036:

```
http://vsms.cisco.com/info.bwt?type=proxy&name=1036&property=status
```

Get Proxy Device

http://host/info.bwt?type=proxy&name=proxyName&property=device

Purpose	Retrieves the current device information for a proxy. The return value is in plain text format.	
Required Fields	host	IP address or hostname (<i>hostname.domain</i>) where VSMS is running. By default, VSMS runs on port 80 (HTTP), however, you can use an alternate port, such as port 8080. For example, to specify port 8080, use <i>host:8080</i> .
	type=proxy	Proxy type. The proxy keyword specifies the proxy command type. The proxy keyword is a reserved value.
	name=proxyName	Proxy name where <i>proxyName</i> specifies the name of the proxy instance and the name of the source for a child proxy. The valid value for <i>proxyName</i> is an alphanumeric string containing 1 to 256 of the following characters: • Digits (0 to 9) • Upper case letters (A to Z) • Lower case letters (a to z) • Underscore (_) • Hyphen (-) The reserved <i>proxyName</i> value is -1. Note Each proxy must have a unique name on a given VSMS host.
	property=device	Device property. The device keyword requests the device information for the specified proxy. The device keyword is a reserved value.

Return Values	A standard HTTP/1.x header followed by: Content-Type: text/plain Return Code: <[Device name] or -1 or output> [Device name] Successful completion of the URL command -1 Error in execution of the URL command
---------------	---

Examples	The following example retrieves the current device information for proxy 1036: http://vsms.cisco.com/info.bwt?type=proxy&name=1036&property=device
----------	---

Get Proxy Frame Size

http://host/info.bwt?type=framesize&name=proxyName

Purpose

Retrieves the frame size of a proxy. The return value is in plain text format.

Required Fields

<i>host</i>	IP address or hostname (<i>hostname.domain</i>) where VSMS is running. By default, VSMS runs on port 80 (HTTP), however, you can use an alternate port, such as port 8080. For example, to specify port 8080, use <i>host:8080</i> .
type=framesize	Frame size type. The framesize keyword requests the frame size for the proxy. The framesize keyword is a reserved value.
name=proxyName	Proxy name where <i>proxyName</i> specifies the name of the proxy instance and the name of the source for a child proxy. The valid value for <i>proxyName</i> is an alphanumeric string containing 1 to 256 of the following characters: • Digits (0 to 9) • Upper case letters (A to Z) • Lower case letters (a to z) • Underscore (_) • Hyphen (-) The reserved <i>proxyName</i> value is -1. 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: <[Device name] or -1 or output>
             [Device name] Successful completion of the URL command
             -1 Error in execution of the URL command
```

Examples

The following example retrieves the current device information for proxy 1036:

```
http://vsms.cisco.com/info.bwt?type=proxy&name=1036&property=device
```




CHAPTER 4

Archive Commands

Table 4-1 provides a summary of the archive commands. Each command is described in detail in the section that is listed.

Table 4-1 **Archive Command Summary**

Name and Reference	Description
Start Archive, page 4-2	Starts an archive process on a media server that records data from a source proxy.
Update Archive, page 4-5	Updates an existing archive with different parameter values.
Rename Archive, page 4-8	Renames an archive that is not currently running.
Stop Archive, page 4-10	Stops a running archive.
Remove Archive, page 4-11	Removes an archive from the repository.
List All Archives, page 4-12	Displays information about all archives on a VSMS host.
List All Running Archives, page 4-13	Displays information for all running archives.
Get Archive MediaType, page 4-14	Retrieves the media type value for an archive.
Get Archive Details, page 4-15	Gets archive details.
Get Archive Monitoring Detail, page 4-20	Retrieves detailed archive performance information.
Get Archive Monitoring Summary, page 4-21	Retrieves summary archive performance information.
Create Archive Clip, page 4-22	Creates a clip from an archive.

Start Archive

```
http://host/command.bwt?command=start&type=archive&name=archiveName
&source=proxyName&duration=seconds&framerate=rateNum&loop=loopVal
&desc=description&repos=location&daystolive=liveNum&killproxy=killVal
&force=forceVal
```

Purpose

Starts an archive process on a media server that records data from a source proxy.

Required Fields

<i>host</i>	IP address or hostname (<i>hostname.domain</i>) where VSMS is running. By default, VSMS runs on port 80 (HTTP), however, you can use an alternate port, such as port 8080. For example, to specify port 8080, use <i>host:8080</i> .
command=start	Start command. The start keyword associates the command with a start action, in this case a start proxy action. The start keyword is a reserved value.
type=archive	Archive type. The archive keyword specifies the archive command type. The archive keyword is a reserved value.
name=archiveName	Archive name where <i>archiveName</i> specifies the name of the archive instance. The valid value for the <i>archiveName</i> value is an alphanumeric string containing 1 to 256 of the following characters: • Digits (0 to 9) • Upper case letters (A to Z) • Lower case letters (a to z) • Underscore (_) • Hyphen (-) The reserved <i>archiveName</i> value is -1. Note Each archive must have a unique name on a given VSMS host.
source=proxyName	The name of the source proxy being archived on the VSMS host. The valid value for <i>proxyName</i> is an alphanumeric string containing 1 to 256 of the following characters: • Digits (0 to 9) • Upper case letters (A to Z) • Lower case letters (a to z) • Underscore (_) • Hyphen (-) The reserved <i>proxyName</i> value is -1.

Optional Fields	duration=seconds	The total archive time where <i>seconds</i> specifies the total recording time (in seconds) for the archive. Valid <i>seconds</i> values are integers. The default value is 3600. Note For loop archives, the minimum supported duration is 3600 seconds.
	desc=description	Archive description where <i>description</i> specifies a brief description of the archive. The valid value for the <i>description</i> value is an alphanumeric string containing 1 to 256 of the following characters: • Digits (0 to 9) • Upper case letters (A to Z) • Lower case letters (a to z) • Underscore (_) • Hyphen (-) • Space character The reserved <i>description</i> value is -1. Note If you do not specify a description, the source proxy name is used as the description.
	framerate=rateNum	Frame rate where <i>rateNum</i> specifies the maximum number of frames per second requested from the source proxy. Valid values for the <i>rateNum</i> value are 0.001 to 30. Note The archive frame rate cannot be higher than the source proxy frame rate.
	loop=loopVal	Specifies whether to record a loop or a regular archive. The <i>loopVal</i> value is one of the following boolean values: • 0—Regular archive. The archive stops when it reaches the end of its duration. • 1—Loop archive. The archive continuously records, overwriting the beginning of the archive when it reaches the end of its duration. The default value is 0.
	repos=location	DO NOT USE. This parameter is ignored.
	daystolive=liveNum	Days to live where <i>liveNum</i> specifies the number of days (starting from the day the archive stops) that the archive is stored before being removed from the system. Valid <i>liveNum</i> values are integers. The default value is 0, which permanently stores the archive in the repository.

killproxy=killVal	Specifies whether to pause the archive. The <i>killVal</i> value is one of the following boolean values: • 0—Immediately pauses the archive. • 1—Pauses the archive after the archiver process completes. The default value is 0.
force=forceVal	Specifies whether to ignore the storage space check and start the archive. The <i>forceVal</i> value is one of the following boolean values: • 0—The system checks whether there is enough space to store the archive. If there is insufficient storage space, the archiver process does not start. • 1—Ignores the storage space check and starts the archive. If there is insufficient storage space for the archive, VSM grooms the oldest data to make room for the new archive. The default value is 0.

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
```

Examples

Starting a Non-Loop Archive

The following example starts an archive named officeCam that stops recording after 60 minutes:

```
http://vsms.cisco.com/command.bwt?command=start&type=archive&name=officeCamFor60Min
&source=officeCam&duration=3600&force=1
```

Starting a 24-Hour Loop Archive

The following example starts an archive named officeCam that records in a continuous 24 hour loop:

```
http://vsms.cisco.com/command.bwt?command=start&type=archive
&name=officeCamFor24Hr&source=officeCam&duration=86400&loop=1
```

Starting a 2-Hour Archive

The following example starts a 2-hour archive that pauses when complete:

```
http://vsms.cisco.com/command.bwt?command=start&type=archive
&name=audio_arch&source=audiocast&duration=7200&killproxy=1
```


Update Archive

Updates an existing archive with different parameter values.

The following APIs are available for updating existing archives:

- [Update JPEG Archive Frame Rate, page 4-6](#)
- [Update Archive Expiration Time, page 4-7](#)

Update JPEG Archive Frame Rate

`http://host/command.bwt?command=update&type=archive&name=archiveName
&framerate=rateNum`

Purpose

Updates the frame rate of a JPEG archive.

Required Fields

<i>host</i>	IP address or hostname (<i>hostname.domain</i>) where VSMS is running. By default, VSMS runs on port 80 (HTTP), however, you can use an alternate port, such as port 8080. For example, to specify port 8080, use <i>host:8080</i> .
command=update	Update command. The update keyword associates the command with an update action (an update archive action in this case). The update keyword is a reserved value.
type=archive	Archive type. The archive keyword specifies the archive command type. The archive keyword is a reserved value.
name=archiveName	Archive name where <i>archiveName</i> specifies the name of the archive instance. The valid value for the <i>archiveName</i> value is an alphanumeric string containing 1 to 256 of the following characters: • Digits (0 to 9) • Upper case letters (A to Z) • Lower case letters (a to z) • Underscore (_) • Hyphen (-) The reserved <i>archiveName</i> value is -1. Note Each archive must have a unique name on a given VSMS host.
framerate=rateNum	Frame rate where <i>rateNum</i> specifies the maximum number of JPEG frames per second requested from the source proxy. Valid values for the <i>rateNum</i> value are 0.001 to 30. Note The archive frame rate cannot be higher than the source proxy frame rate.

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
```

Examples

The following example updates the frame rate of a JPEG archive named Archive30 to a value of 15:

```
http://vsms.cisco.com/command.bwt?command=update& type=archive&name=Archive30
&framerate=15
```

Update Archive Expiration Time



Caution

Use this API with caution.

**http://host/command.bwt?command=update&type=archive&name=archiveName
&daystolive=liveNum**

Purpose

Updates an existing archive with the number of days (starting from the day the archive stops) that the archive is stored before being removed from the system.

Required Fields

host	IP address or hostname (<i>hostname.domain</i>) where VSMS is running. By default, VSMS runs on port 80 (HTTP), however, you can use an alternate port, such as port 8080. For example, to specify port 8080, use host:8080 .
command=update	Update command. The update keyword associates the command with an update action (an update archive action in this case). The update keyword is a reserved value.
type=archive	Archive type. The archive keyword specifies the archive command type. The archive keyword is a reserved value.
name=archiveName	Archive name where <i>archiveName</i> specifies the name of the archive instance. The valid value for the <i>archiveName</i> value is an alphanumeric string containing 1 to 256 of the following characters: • Digits (0 to 9) • Upper case letters (A to Z) • Lower case letters (a to z) • Underscore (_) • Hyphen (-) The reserved <i>archiveName</i> value is -1. Note Each archive must have a unique name on a given VSMS host.
daystolive=liveNum	Days to live where <i>liveNum</i> specifies the number of days (starting from the day the archive stops) that the archive is stored before being removed from the system. Valid <i>liveNum</i> values are integers. The default value is 0, which permanently stores the archive in the repository.

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

Examples

The following example updates the archive named Archive30 to expire after 10 days:

**http://vsms.cisco.com/command.bwt?command=update&type=archive&name=Archive30
&daystolive=10**

Rename Archive

**http://host/rename_archive.bwt?command=rename&oldname=archiveName
&newname=archiveName**

Purpose

Renames an archive that is not currently running.

Required Fields

<i>host</i>	IP address or hostname (<i>hostname.domain</i>) where VSMS is running. By default, VSMS runs on port 80 (HTTP), however, you can use an alternate port, such as port 8080. For example, to specify port 8080, use <i>host:8080</i> .
command=rename	Rename command. The rename keyword associates the command with a rename action, in this case a rename archive action. The rename keyword is a reserved value.
oldname=archiveName	Old archive name where <i>archiveName</i> specifies the name of the archive instance. The valid value for the <i>archiveName</i> value is an alphanumeric string containing 1 to 256 of the following characters: • Digits (0 to 9) • Upper case letters (A to Z) • Lower case letters (a to z) • Underscore (_) • Hyphen (-) The reserved <i>archiveName</i> value is -1. Note Each archive must have a unique name on a given VSMS host.
newname=archiveName	New archive name where <i>archiveName</i> specifies the name of the archive instance. The valid value for the <i>archiveName</i> value is an alphanumeric string containing 1 to 256 of the following characters: • Digits (0 to 9) • Upper case letters (A to Z) • Lower case letters (a to z) • Underscore (_) • Hyphen (-) The reserved <i>archiveName</i> value is -1. Note Each archive 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: <1 OK or -1 or output>
1 OK Successful completion of the URL command
-1 Error in execution of the URL command

Examples

The following command renames an archive from ABC to BCD.

```
http://vsms.cisco.com/command.bwt?command=rename&oldname=ABC&newname=BCD
```

Stop Archive

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

Purpose Stops a running archive.

Required Fields	<i>host</i>	IP address or hostname (<i>hostname.domain</i>) where VSMS is running. By default, VSMS runs on port 80 (HTTP), however, you can use an alternate port, such as port 8080. For example, to specify port 8080, use <i>host:8080</i> .
	command=stop	Stop command. The stop keyword associates the command with a stop action. The stop keyword is a reserved value.
	type=archive	Archive type. The archive keyword specifies the archive command type. The archive keyword is a reserved value.
	name=archiveName	Archive name where <i>archiveName</i> specifies the name of the archive instance. The valid value for the <i>archiveName</i> value is an alphanumeric string containing 1 to 256 of the following characters: • Digits (0 to 9) • Upper case letters (A to Z) • Lower case letters (a to z) • Underscore (_) • Hyphen (-) The reserved <i>archiveName</i> value is -1. Note Each archive 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: <[archive name] or -1 or output>
[archive name] Successful completion of the URL command
-1 Error in execution of the URL command
```

Examples The following command stops an archive with server name ABC.

```
http://vsms.cisco.com/command.bwt?command=stop&type=archive&name=ABC
```

Remove Archive

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

Purpose

Removes a stopped archive from the repository.

Required Fields

<i>host</i>	IP address or hostname (<i>hostname.domain</i>) where VSMS is running. By default, VSMS runs on port 80 (HTTP), however, you can use an alternate port, such as port 8080. For example, to specify port 8080, use <i>host:8080</i> .
command=remove	Remove command. The remove keyword associates the command with a remove action. The remove keyword is a reserved value.
type=archive	Archive type. The archive keyword specifies the archive command type. The archive keyword is a reserved value.
name=archiveName	Archive name where <i>archiveName</i> specifies the name of the archive instance. The valid value for the <i>archiveName</i> value is an alphanumeric string containing 1 to 256 of the following characters: • Digits (0 to 9) • Upper case letters (A to Z) • Lower case letters (a to z) • Underscore (_) • Hyphen (-) The reserved <i>archiveName</i> value is -1. Note Each archive 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: <0 [archive name] or -1>
0 [archive name] Successful completion of the URL command
-1 Error in execution of the URL command

Examples

The following command removes archive ABC.

`http://vsms.cisco.com/cgi-bin/smanager.bwt?command=remove&name=ABC`

List All Archives

`http://host/info.bwt?type=archive&display=dispFormat`

Purpose

Displays information about all archives on a VSMS host.

Required Fields

<i>host</i>	IP address or hostname (<i>hostname.domain</i>) where VSMS is running. By default, VSMS runs on port 80 (HTTP), however, you can use an alternate port, such as port 8080. For example, to specify port 8080, use <i>host:8080</i> .
type=archive	Archive type. The archive keyword specifies the archive command type. The archive keyword is a reserved value.

Optional Fields

display=dispFormat	Type of display output where <i>dispFormat</i> specifies the format to use when displaying the list of running proxies. Valid <i>dispFormat</i> values include the following reserved keywords: html—Hypertext markup language format text—Plain text format ssv—Space-separated value format The default <i>dispFormat</i> value is html .
---------------------------	---

Return Values

A standard HTTP/1.x header followed by:

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

Examples

The following command lists all the archives in SSV format.

```
http://vsms.cisco.com/info.bwt?type=archive&display=ssv
```


List All Running Archives

`http://host/info.bwt?type=archiver&display=dispFormat`

Purpose Displays information for all running archives.

Required Fields	<i>host</i>	IP address or hostname (<i>hostname.domain</i>) where VSMS is running. By default, VSMS runs on port 80 (HTTP), however, you can use an alternate port, such as port 8080. For example, to specify port 8080, use <i>host:8080</i> .
	type=archiver	Archive type. The archiver keyword specifies the archive command type. The archiver keyword is a reserved value.

Optional Fields	display=dispFormat	Type of display output where <i>dispFormat</i> specifies the format to use when displaying the list of running proxies. Valid <i>dispFormat</i> values include the following reserved keywords: html—Hypertext markup language format text—Plain text format ssv—Space-separated value format The default <i>dispFormat</i> value is html .
------------------------	---------------------------	---

Return Values A standard HTTP/1.x header followed by:

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

Examples The following command lists all the running archives in SSV format.

```
http://vsms.cisco.com/info.bwt?type=archiver&display=ssv
```

Get Archive MediaType

http://host/info.bwt?type=archive&name=archiveName&property=mediatype

Purpose Retrieves the media type value for an archive.

Required Fields	<i>host</i>	IP address or hostname (<i>hostname.domain</i>) where VSMS is running. By default, VSMS runs on port 80 (HTTP), however, you can use an alternate port, such as port 8080. For example, to specify port 8080, use <i>host:8080</i> .
	type=archive	Archive type. The archive keyword specifies the archive command type. The archive keyword is a reserved value.
	name=archiveName	Archive name where <i>archiveName</i> specifies the name of the archive instance. The valid value for the <i>archiveName</i> value is an alphanumeric string containing 1 to 256 of the following characters: • Digits (0 to 9) • Upper case letters (A to Z) • Lower case letters (a to z) • Underscore (_) • Hyphen (-) The reserved <i>archiveName</i> value is -1. Note Each archive must have a unique name on a given VSMS host.
	property=mediatype	Media type. The mediatype keyword requests the media type value for the specified archive, which is a JPEG, MPEG, or audio value. The mediatype keyword is a reserved value.

Return Values A standard HTTP/1.x header followed by:

```
Content-Type: text/plain
Return Code: <[jpeg | mpeg2-v | mpeg4-v | h264-v | audio] or -1 or output>
[jpeg | mpeg2-v | mpeg4-v | h264-v | audio] Successful completion of the URL command
-1 Error in execution of the URL command
```

Examples The following example retrieves the media type value for the archive named testArchive:

```
http://vsms.cisco.com/info.bwt?type=archive&name=testArchive&property=mediatype
```

Get Archive Details

Gets archive details.

The following APIs are available for retrieving detailed archive recording information:

- [Get Archive Recording Details, page 4-16](#)
- [Archive Details, page 4-18](#)

Get Archive Recording Details

`http://host/info.bwt?type=archive&property=mmconf_properties&display=dispFormat
&name=archiveName`

Purpose

Displays recording details for an archive.



Caution

This command is CPU intensive and should not be run frequently.

Required Fields

<i>host</i>	IP address or hostname (<i>hostname.domain</i>) where VSMS is running. By default, VSMS runs on port 80 (HTTP), however, you can use an alternate port, such as port 8080. For example, to specify port 8080, use <i>host:8080</i> .
type=archive	Archive type. The archive keyword specifies the archive command type. The archive keyword is a reserved value.
name=archiveName	Archive name where <i>archiveName</i> specifies the name of the archive instance. The valid value for the <i>archiveName</i> value is an alphanumeric string containing 1 to 256 of the following characters: • Digits (0 to 9) • Upper case letters (A to Z) • Lower case letters (a to z) • Underscore (_) • Hyphen (-) The reserved <i>archiveName</i> value is -1. Note Each archive must have a unique name on a given VSMS host.
property=mmconf_properties	Recording information. The mmconf_properties keyword requests the XML-formatted recording details for the specified archive. The mmconf_properties keyword is a reserved value.

Optional Fields

display=dispFormat	Type of display output where <i>dispFormat</i> specifies the format to use when displaying the list of running proxies. Valid <i>dispFormat</i> values include the following reserved keywords: • html—Hypertext markup language format • text—Plain text format • ssv—Space-separated value format The default <i>dispFormat</i> value is html .
---------------------------	---

Return Values

Detailed recording information about the archive.

Examples

The following command displays the detailed recording information for the archive named ABC.

```
http://vms.cisco.com/info.bwt?type=archive&name=ABC&property=mmconf_properties
```

Archive Details

**`http://host/info.bwt?type=archive&property=archive_details&display=dispFormat
&name=archiveName`**

Purpose

Displays details for an archive, excluding information about the first and last frame time.

Required Fields

<code>host</code>	IP address or hostname (<i>hostname.domain</i>) where VSMS is running. By default, VSMS runs on port 80 (HTTP), however, you can use an alternate port, such as port 8080. For example, to specify port 8080, use <code>host:8080</code> .
<code>type=archive</code>	Archive type. The <code>archive</code> keyword specifies the archive command type. The <code>archive</code> keyword is a reserved value.
<code>name=archiveName</code>	Archive name where <i>archiveName</i> specifies the name of the archive instance. The valid value for the <i>archiveName</i> value is an alphanumeric string containing 1 to 256 of the following characters: • Digits (0 to 9) • Upper case letters (A to Z) • Lower case letters (a to z) • Underscore (_) • Hyphen (-) The reserved <i>archiveName</i> value is -1. Note Each archive must have a unique name on a given VSMS host.
<code>property=archive_details</code>	Recording information. The <code>archive_details</code> keyword requests the XML-formatted recording details for the specified archive, excluding information about the first and last frame time. The <code>archive_details</code> keyword is a reserved value.

Optional Fields

<code>display=dispFormat</code>	Type of display output where <i>dispFormat</i> specifies the format to use when displaying the list of running proxies. Valid <i>dispFormat</i> values include the following reserved keywords: • <code>html</code>—Hypertext markup language format • <code>text</code>—Plain text format • <code>ssv</code>—Space-separated value format The default <i>dispFormat</i> value is <code>html</code> .
--	---

Return Values

Detailed recording information about the archive, excluding information about the first and last frame time.

Examples

The following command displays the detailed recording information for the archive named ABC.

```
http://vms.cisco.com/info.bwt?type=archive&name=ABC&property=archive_details
```

Get Archive Monitoring Detail

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

Purpose Retrieves detailed archive performance information.



Caution This command is CPU intensive and should not be run frequently.

Required Fields	<i>host</i>	IP address or hostname (<i>hostname.domain</i>) where VSMS is running. By default, VSMS runs on port 80 (HTTP), however, you can use an alternate port, such as port 8080. For example, to specify port 8080, use <i>host:8080</i> .
	type=archive	Archive type. The archive keyword specifies the archive command type. The archive keyword is a reserved value.
	name=archiveName	Archive name where <i>archiveName</i> specifies the name of the archive instance. The valid value for the <i>archiveName</i> value is an alphanumeric string containing 1 to 256 of the following characters: • Digits (0 to 9) • Upper case letters (A to Z) • Lower case letters (a to z) • Underscore (_) • Hyphen (-) The reserved <i>archiveName</i> value is -1. Note Each archive must have a unique name on a given VSMS host.
	property=armon_detail	Archive monitoring summary property. The armon_detail keyword requests XML information that shows detailed performance information for an archive. The armon_detail keyword is a reserved value.

Return Values XML information that shows detailed performance information about the archive.

Examples The following example retrieves the XML information that shows detailed recording information for the archive ABC.

`http://vsms.cisco.com/info.bwt?type=archive&name=ABC&property=armon_detail`

Get Archive Monitoring Summary

`http://host/info.bwt?type=archive&name=archiveName&property=armon_summary`

Purpose

Retrieves summary archive performance information.



Caution

This command is CPU intensive and should not be run frequently.

Required Fields

<i>host</i>	IP address or hostname (<i>hostname.domain</i>) where VSMS is running. By default, VSMS runs on port 80 (HTTP), however, you can use an alternate port, such as port 8080. For example, to specify port 8080, use <i>host:8080</i> .
type=archive	Archive type. The archive keyword specifies the archive command type. The archive keyword is a reserved value.
name=archiveName	Archive name where <i>archiveName</i> specifies the name of the archive instance. The valid value for the <i>archiveName</i> value is an alphanumeric string containing 1 to 256 of the following characters: • Digits (0 to 9) • Upper case letters (A to Z) • Lower case letters (a to z) • Underscore (_) • Hyphen (-) The reserved <i>archiveName</i> value is -1. Note Each archive must have a unique name on a given VSMS host.
property=armon_summary	Archive monitoring summary property. The armon_summary keyword requests XML information that shows the recording rate of the archive in MB per second. The armon_summary keyword is a reserved value.

Return Values

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

Examples

The following example retrieves the XML information that shows detailed recording information for the archive ABC.

`http://vsms.cisco.com/info.bwt?type=archive&name=ABC&property=armon_summary`

Create Archive Clip

```
http://host/cgi-bin/smanager.bwt?command=save&source=archiveName&startutc=utcDate
&stoputc=utcDate&name=target_id&savemode=local&desc=description
&saveformat=clipType&daystolive=liveNum&notifyurl=notifyUrl
```

Purpose

Creates a clip from an archive.

Required Fields

<i>host</i>	IP address or hostname (<i>hostname.domain</i>) where VSMS is running. By default, VSMS runs on port 80 (HTTP), however, you can use an alternate port, such as port 8080. For example, to specify port 8080, use <i>host:8080</i> .
command=save	Save command. The save keyword associates the command with a save action, in this case saving a clip from an archive. The save keyword is a reserved value.
source=archiveName	Source archive where <i>archiveName</i> specifies the name of the archive instance. This is the parent archive from which to create the clip. The valid value for the <i>archiveName</i> value is an alphanumeric string containing 1 to 256 of the following characters: • Digits (0 to 9) • Upper case letters (A to Z) • Lower case letters (a to z) • Underscore (_) • Hyphen (-) The reserved <i>archiveName</i> value is -1. Note Each archive must have a unique name on a given VSMS host.
savemode=local	Save location for the archive clip. The local keyword specifies that only local clips are supported. The local keyword is a reserved value.
startutc=utcDate	Start date of the child clip, specified in UTC milliseconds. Verify the parent archive contains data for this date.
stoputc=utcDate	Stop date of the child clip, specified in UTC milliseconds. Make sure the parent archive contains data for this date.

Optional Fields

name= <i>target_id</i>	Target archive name where <i>target_id</i> specifies the name of the archive instance. The valid value for the <i>target_id</i> value is an alphanumeric string containing 1 to 256 of the following characters: • Digits (0 to 9) • Upper case letters (A to Z) • Lower case letters (a to z) • Underscore (_) • Hyphen (-) The reserved <i>archiveName</i> value is -1. Note Each archive must have a unique name on a given VSMS host.
desc= <i>description</i>	Archive description where <i>description</i> specifies a brief description of the archive. The valid value for the <i>description</i> value is an alphanumeric string containing 1 to 20 of the following characters: • Digits (0 to 9) • Upper case letters (A to Z) • Lower case letters (a to z) • Underscore (_) • Hyphen (-) • Space character The reserved <i>description</i> value is -1. If this parameter is not specified, VSMS will store the name of the proxy source in the description field.
repos= <i>location</i>	DO NOT USE. This parameter is ignored.
saveformat= <i>clipType</i>	Save format where <i>clipType</i> specifies the type of archive clip to generate. Valid <i>clipType</i> values include the following keywords: • regular—Regular streamable archive clip • bwm—BWM archive clip • bwx—BWX secure video clip The default <i>clipType</i> value is regular.
daystolive= <i>liveNum</i>	Days to live where <i>liveNum</i> 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 <i>liveNum</i> 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= <i>notifyUrl</i>	Format: [http://<i>host</i>/<i>handler_path</i>] 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. The save clip operation will run 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's notification URL to receive notification if a clip has succeeded or failed.

Examples

The following command saves a clip from archive ABC, 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.cisco.com/cgi-bin/smanager.bwt?command=save&startutc=1020530754089&stoputc=1020530786232&source=ABC&savemode=local&saveformat=regular
```



CHAPTER 5

Event Commands

Table 5-1 provides a summary of the event commands. Each command is described in detail in the section that is listed.

Table 5-1 **Event Command Summary**

Name and Reference	Description
Event Setup, page 5-2	Sets up an event in VSMS.
Enable Event, page 5-9	Enables an event previously set up or disabled in VSMS.
Disable Event, page 5-10	Disables an event previously set up or enabled in VSMS.
Remove Event, page 5-11	Removes an event previously set up in VSMS.
Trigger VSMS Event, page 5-13	Triggers an event configured in VSMS.
Event Clip Start/Stop, page 5-18	Starts or stops event-based clips.
Get Event Information, page 5-19	Retrieves event information.

Event Setup

`http://host/event.bwt?command=setup&data=xmlData`

Purpose Sets up an event in VSMS.

Required Fields	<i>host</i>	IP address or hostname (<i>hostname.domain</i>) where VSMS is running. By default, VSMS runs on port 80 (HTTP), however, you can use an alternate port, such as port 8080. For example, to specify port 8080, use <i>host:8080</i> .
	command=setup	Setup command. The setup keyword associates the command with a setup action. The setup keyword is a reserved value.
	data=xmlData	XML data comprised of XML elements and values. The structure of the XML data is as follows: <pre> <xml> <event> <name></name> <ipdevice></ipdevice> <srctype></srctype> <notifyURL></notifyURL> <action> <clip/> <accelerate/> </action> <cliphost> <localhost/> </cliphost> <trigger> <input></input> <state></state> <type></type> <notificationtype></notificationtype> <maxevents></maxevents> <daystolive></daystolive> <framerate></framerate> <acclframerate></acclframerate> <duration></duration> <prebuffer></prebuffer> <postbuffer></postbuffer> <proxysource></proxysource> </trigger> </event> </xml> </pre>
For more information about the XML elements, see Table 5-2 .		

Table 5-2 Event Setup XML Elements

XML Element	Description
xml	Start XML parsing element; contains the event element.
event	Start event data element; contains the name, ipdevice, srctype, notifyurl, action, cliphost, and trigger elements.

Table 5-2 Event Setup XML Elements

XML Element	Description
name	The name for this event. The name element may contain 1 to 32 of the following characters: • Digits (0 to 9) • Upper case letters (A to Z) • Lower case letters (a to z) • Underscore (_) • Hyphen (-) The reserved value is -1.
ipdevice	Can be one of the following: • Format: [IP address] IP address of device for a running proxy where the trigger is set up. This will enable the triggers from the event driver of the device. • Format: [unique ID] for soft triggers. Each event must be setup with a unique IP address or ID. Contains no other elements.
srctype	Specifies the type of video server to set up the trigger; contains no other elements. 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.
notifyurl	Format: http://host/handlerPath. The notification URL to use when an event trigger is received by VSMS, and if archive clips are requested, after archives are saved; contains no other elements. Use the notifyurl element in conjunction with the notificationtype element. Valid values for the notificationtype element can be one of the following: • 0—An event notification is sent. • 2—An event notification and after event clip saved notification are sent. For more information, see the “Event Setup Notification” section on page 5-7.
action	Start action data element; contains the clip-ondemand, clip, and accelerate elements. Note If the action element is not specified but notificationtype is set to 2 (record event triggered archives), event clips are still recorded.
clip	Tag Format: <clip/>. Creates a clip when an event occurs. Note The notificationtype must be set to 2 (record event triggered archives).
accelerate	Tag Format: <accelerate/>. Accelerates the event archive recording framerate at the event for the postbuffer time. Note The notificationtype must be set to 2 (record event triggered archives).
cliphost	Start cliphost data element; contains localhost element. Note If no cliphost element is specified, then the event clip is saved to local host.

Table 5-2 Event Setup XML Elements

XML Element	Description
name	The name for this event. The name element may contain 1 to 32 of the following characters: • Digits (0 to 9) • Upper case letters (A to Z) • Lower case letters (a to z) • Underscore (_) • Hyphen (-) The reserved value is -1.
ipdevice	Can be one of the following: • Format: [IP address] IP address of device for a running proxy where the trigger is set up. This will enable the triggers from the event driver of the device. • Format: [unique ID] for soft triggers. Each event must be setup with a unique IP address or ID. Contains no other elements.
srctype	Specifies the type of video server to set up the trigger; contains no other elements. 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.
notifyurl	Format: http://host/handlerPath. The notification URL to use when an event trigger is received by VSMS, and if archive clips are requested, after archives are saved; contains no other elements. Use the notifyurl element in conjunction with the notificationtype element. Valid values for the notificationtype element can be one of the following: • 0—An event notification is sent. • 2—An event notification and after event clip saved notification are sent. For more information, see the “Event Setup Notification” section on page 5-7.
action	Start action data element; contains the clip-ondemand, clip, and accelerate elements. Note If the action element is not specified but notificationtype is set to 2 (record event triggered archives), event clips are still recorded.
clip	Tag Format: <clip/>. Creates a clip when an event occurs. Note The notificationtype must be set to 2 (record event triggered archives).
accelerate	Tag Format: <accelerate/>. Accelerates the event archive recording framerate at the event for the postbuffer time. Note The notificationtype must be set to 2 (record event triggered archives).
cliphost	Start cliphost data element; contains localhost element. Note If no cliphost element is specified, then the event clip is saved to local host.

Table 5-2 **Event Setup XML Elements**

XML Element	Description
localhost	Tag Format: <localhost/>. Event clip is saved to the local host. Saves the event clip directly to the repository mount location. Only one mount will be recognized. If no repository is specified, 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 clip repository option must be chosen from the Clipping drop-down menu on the VSMC Console page.
trigger	Start trigger data element; contains the input, state, type, notificationtype, maxevents, daystolive, framerate, accelframerate, duration, prebuffer, postbuffer, and proxysource elements.
input	Range: [0] Reserved for generic trigger input number. Make sure to pair this with a unique ID for the ipdevice value. Range: [1-6] Trigger input number on the 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 (should have action type as clip) 3: Unsupported
maxevents	Format: [integer] Maximum number of events recorded per month.
daystolive	Format: [integer] (default=0) Number of days from the date archive stops the archive will be stored before system removal. For permanent storage set daystolive to Ø.
framerate	Range: [0.001-30](proxy frame rate) Maximum number of frames per second transmitted to record proxy.
accelframerate	Range: [0.001-30] Accelerated archive frame rate that the event is recorded at. Note The accelerated archive frame rate must be less than or equal to the proxy frame rate.
duration	Format: [integer] Duration of the event archive loop in seconds. Note The only supported duration is 300 seconds.
prebuffer	Format: [integer] (default=10 seconds) The number of seconds before the event that will be included in the event archive clip.

Table 5-2 Event Setup XML Elements

XML Element	Description
postbuffer	Format: [integer] (default=30 seconds) The number of seconds after the event that will be included in the event archive clip.
proxysource	Proxy name for event to archive. Each triggered event can record up to 10 different proxies. The proxysource element may contain 1 to 64 of the following characters: • Digits (0 to 9) • Upper case letters (A to Z) • Lower case letters (a to z) • Underscore (_) • Hyphen (-) The reserved value is -1. 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

Usage Guidelines

When an event is set up using an IP address (device trigger), the device inputs are used to trigger the event through the event driver.

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.

Examples

The following is an example of the XML data that is specified as an HTTP command to the VSM:

```
<xml>
  <event>
    <name>test_event</name>
    <ipdevice>1234567</ipdevice>
    <srctype>optelecom_c50</srctype>
    <trigger>
      <input>0</input>
      <state>rising</state>
      <type>alarm</type>
      <notificationtype>0</notificationtype>
    </trigger>
  </event>
</xml>
```

The above XML data is specified as an HTTP command to VSMS as follows:

```
http://vsms.cisco.com/event.bwt?command=setup&data=<xml><event><name>test_event</name><ipdevice>1234567</ipdevice><srctype>optelecom_c50</srctype><trigger><input>0</input><state>rising</state><type>alarm</type><notificationtype>0</notificationtype></trigger></event></xml>
```

Event Setup Notification

Purpose

When an event is set up with a notificationtype element value of 2, event setup notification data is sent to the specified notification URL.



Note

No event setup notification data is sent when an event is set up with a notificationtype element value of 0.

notifyUrl?data=xmlData

Required Fields

notifyUrl The notification URL specified by an Event Setup command. For more information, see the notifyurl XML element description in [Table 5-2 on page 5-2](#).

data=xmlData XML data comprised of XML elements and values. The structure of the XML data is as follows:

```
<xml>
  <ConfigNotification>
    <Host></Host>
    <VideoServer></VideoServer>
    <TriggerInput></TriggerInput>
    <ProxyName></ProxyName>
    <ArchiveName></ArchiveName>
    <StartUTC></StartUTC>
    <Duration></Duration>
    <StopMode></StopMode>
    <Type></Type>
  </ConfigNotification>
</xml>
```

For more information about the XML elements, see [Table 5-3](#).

Table 5-3 Event Setup Notification XML Elements

XML Element	Description
xml	Start XML parsing element; contains the TriggerNotification element.
ConfigNotification	Event configuration (setup) notification element. Contains the Host, VideoServer, TriggerInput, ProxyName, ArchiveName, StartUTC, Duration, StopMode, and Type elements.
Host	Format: [hostname.domain IP address] The web address of the host where VSMS is running. VSMS runs on port 80 by default.

Table 5-3 Event Setup Notification XML Elements

XML Element	Description
VideoServer	Name of the event. The event name may contain 1 to 256 of the following characters: • Digits (0 to 9) • Upper case letters (A to Z) • Lower case letters (a to z) • Underscore (_) • Hyphen (-)
TriggerInput	Range: [0] Generic Trigger input number. Range: [1-6] Trigger input number on device. Range: [1-10] Window number for motion detection.
ProxyName	The proxy name for the event. The ProxyName element may contain 1 to 256 of the following characters: • Digits (0 to 9) • Upper case letters (A to Z) • Lower case letters (a to z) • Underscore (_) • Hyphen (-) The reserved value is -1. Note The proxy must exist before adding an event trigger.
ArchiveName	The archive name for this event. The ArchiveName element may contain 1 to 256 of the following characters: • Digits (0 to 9) • Upper case letters (A to Z) • Lower case letters (a to z) • Underscore (_) • Hyphen (-) The reserved value is -1.
StartUTC	Format: [UTC milliseconds] Date of the event in UTC milliseconds. This date is when VSMS received notification of the event from the encoder.
Duration	Format: [integer] Duration of the event archive in seconds. Note The only supported duration is 300 seconds.
StopMode	Reserved values: [auto manual] Specifies whether the clip is auto-stopped or manually stopped.
Type	Reserved values: [motion alarm]. Specifies whether the type of event is motion detection or trigger.

Enable Event

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

Purpose

Enables an event previously set up or disabled in VSMS.

For more information about setting up or disabling an event, see [Event Setup, page 5-2](#) or [Disable Event, page 5-10](#).

Required Fields

<code>host</code>	IP address or hostname (<i>hostname.domain</i>) where VSMS is running. By default, VSMS runs on port 80 (HTTP), however, you can use an alternate port, such as port 8080. For example, to specify port 8080, use <code>host:8080</code> .
<code>command=enable</code>	Enable event command. The <code>enable</code> keyword associates the command with an enable event action. The <code>enable</code> keyword is a reserved value.
<code>name=eventName</code>	The name for the event to be enabled. The <i>eventName</i> value may contain 1 to 32 of the following characters: • Digits (0 to 9) • Upper case letters (A to Z) • Lower case letters (a to z) • Underscore (_) • Hyphen (-) The reserved value is -1.

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

Usage Guidelines

Enabling an event is accomplished by sending a HTTP request to VSMS with the event name to be enabled. This command enables the specified event set up in VSMS.

Examples

The following example enables the event named test:

`http://vsms.cisco.com/event.bwt?command=enable&name=test`

Disable Event

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

Purpose

Disables an event previously set up or enabled in VSMS.

For more information about setting up or enabling an event, see [Event Setup, page 5-2](#) or [Enable Event, page 5-9](#).

Required Fields	host	IP address or hostname (<i>hostname.domain</i>) where VSMS is running. By default, VSMS runs on port 80 (HTTP), however, you can use an alternate port, such as port 8080. For example, to specify port 8080, use <i>host:8080</i> .
	command=disable	Disable event command. The disable keyword associates the command with an disable event action. The disable keyword is a reserved value.
	name=eventName	The name for the event to be disabled. The <i>eventName</i> value may contain 1 to 32 of the following characters: • Digits (0 to 9) • Upper case letters (A to Z) • Lower case letters (a to z) • Underscore (_) • Hyphen (-) The reserved value is -1.

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
```

Usage Guidelines

Disabling an event is accomplished by sending a HTTP request to VSMS with the event name to be disabled. This command disables the specified event set up in VSMS.

Examples

The following example disables the event named test:

```
http://vsms.cisco.com/event.bwt?command=disable&name=test
```

Remove Event

http://host/event.bwt?name=eventName&command=remove&killarchive=boolValue

Purpose

Removes an event previously set up in VSMS.

For more information about setting up an event, see [Event Setup, page 5-2](#).

Required Fields

<i>host</i>	IP address or hostname (<i>hostname.domain</i>) where VSMS is running. By default, VSMS runs on port 80 (HTTP), however, you can use an alternate port, such as port 8080. For example, to specify port 8080, use <i>host:8080</i> .
command=remove	Remove event command. The remove keyword associates the command with a remove event action. The remove keyword is a reserved value.
name=eventName	The name for the event to be removed. The <i>eventName</i> value may contain 1 to 32 of the following characters: • Digits (0 to 9) • Upper case letters (A to Z) • Lower case letters (a to z) • Underscore (_) • Hyphen (-) The reserved value is -1.

Optional Fields

killarchive=boolVal	Specifies whether the buffered event archive is removed when the event is removed. The <i>killVal</i> value is one of the following boolean values: • true—The buffered event archive is stopped and removed from storage. • false—The buffered event archive is shelved after the event is removed. The default value is true.
----------------------------	--

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
```

Usage Guidelines

Removing an event set-up is accomplished by sending an HTTP request to VSMS with the event name to be removed. Because events require a unique name even for the same device, remove requests do not need the trigger parameter. The command removes the event set up for the trigger only in VSMS.

Examples

The following example removes the event named test:

```
http://vsms.cisco.com/event.bwt?command=remove&name=test
```


Trigger VSMS Event

`http://host/event.bwt?command=event&name=triggerName&nolog=1`

Purpose

Triggers an event configured in VSMS.

Required Fields

host	IP address or hostname (<i>hostname.domain</i>) where VSMS is running. By default, VSMS runs on port 80 (HTTP), however, you can use an alternate port, such as port 8080. For example, to specify port 8080, use <i>host:8080</i> .
command=event	Send event command. The event keyword associates the command with an send event action. The event keyword is a reserved value.
name=triggerName	The name for the event. The <i>triggerName</i> element may contain 1 to 32 of the following characters: • Digits (0 to 9) • Upper case letters (A to Z) • Lower case letters (a to z) • Underscore (_) • Hyphen (-) The reserved value is -1.
nolog=1	specifies that clips will have no entry in media server database (repos.db).

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
```

Usage Guidelines

When an event occurs, the device sends an HTTP request to notify VSMS that 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.

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 must have a handler running that parses the XML data and can react to the notification.

When event archives are requested for this event setup, the notification is sent only after each archive clip has been created. That is, a separate notification is sent for each event archive clip after it has been saved.

Examples

The following example notifies VSMS that an event named test has occurred on a trigger device.

```
http://host/event.bwt?command=event&name=test&nolog=1
```

Event Trigger Notification

Purpose

When an event is set up with a notificationtype element value of 0 or 2, and an event is triggered, event trigger notification information is sent to the specified notification URL.

*N*otifyUrl?*info*=*xmlData*

Required Fields	<i>notifyUrl</i> The notification URL specified by an Event Setup command. For more information, see the notifyurl XML element description in Table 5-2 on page 5-2.
	<i>data=xmlData</i> XML data comprised of XML elements and values. The structure of the XML data is as follows: <pre> <xml> <TriggerNotification> <Host></Host> <EventUTC></EventUTC> <Name></Name> <IpDevice></IpDevice> <SrcType><SrcType> <TriggerInput><TriggerInput> <ProxyList> <ProxyName></ProxyName> </ProxyList> <ArchiveList> <ArchiveName></ArchiveName> </ArchiveList> </TriggerNotification> </xml> </pre> For more information about the XML elements, see Table 5-5.

Table 5-4 Event Trigger Notification XML Elements

XML Element	Description
xml	Start XML parsing element; Contains the TriggerNotification element.
TriggerNotification	Event trigger notification element. Contains the Host, EventUTC, Name, IpDevice, SrcType, TriggerInput, ProxyList, and ArchiveList elements.
Host	Format: [hostname.domain IP address] The web address of the host where VSMS is running. VSMS runs on port 80 by default.
EventUTC	Format: [UTC milliseconds] Date of the event in UTC milliseconds. This date is when VSMS received notification of the event from the encoder.

Table 5-4 Event Trigger Notification XML Elements

XML Element	Description
Name	The name of this event as defined in Event Setup as <name>. The name may contain 1 to 32 of the following characters: • Digits (0 to 9) • Upper case letters (A to Z) • Lower case letters (a to z) • Underscore (_) • Hyphen (-) The reserved value is -1.
IpDevice	Format: [hostname.domain IP address] The web address where the trigger is set up; contains no other elements.
SrcType	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.
ProxyList	Proxy list element. Contains the ProxyName element.
ProxyName	The proxy name for the event. The ProxyName element may contain 1 to 256 of the following characters: • Digits (0 to 9) • Upper case letters (A to Z) • Lower case letters (a to z) • Underscore (_) • Hyphen (-) The reserved value is -1. Note The proxy must exist before adding an event trigger.
ArchiveList	Archive list element. Contains the ArchiveName element.
ArchiveName	The archive name for this event. The ArchiveName element may contain 1 to 256 of the following characters: • Digits (0 to 9) • Upper case letters (A to Z) • Lower case letters (a to z) • Underscore (_) • Hyphen (-) The reserved value is -1.

Event Clip Creation Notification

notifyUrl?data=xmlData

Purpose

When an event is set up with a notificationtype element value of 2, and an event is triggered, event clip creation notification data is sent to the specified notification URL.

Required Fields

<i>notifyUrl</i>	The notification URL specified by an Event Setup command. For more information, see the notifyurl XML element description in Table 5-2 on page 5-2 .
<i>info=xmlData</i>	XML data comprised of XML elements and values. The structure of the XML data is as follows: <pre><xml> <TriggerNotification> <Host></Host> <VideoServer></VideoServer> <TriggerInput></TriggerInput> <ProxyName></ProxyName> <ArchiveName></ArchiveName> <StartUTC></StartUTC> <Duration></Duration> <StopMode></StopMode> <Type></Type> </TriggerNotification> </xml></pre> For more information about the XML elements, see Table 5-4.

Table 5-5 Event Clip Creation Notification XML Elements

XML Element	Description
xml	Start XML parsing element; contains the TriggerNotification element.
TriggerNotification	Trigger notification data element. Contains the Host, VideoServer, TriggerInput, ProxyName, ArchiveName, StartUTC, Duration, StopMode, and Type elements.
Host	Format: [hostname.domain IP address] The web address of the host where VSMS is running. VSMS runs on port 80 by default.
VideoServer	Name of the event. The event name may contain 1 to 256 of the following characters: • Digits (0 to 9) • Upper case letters (A to Z) • Lower case letters (a to z) • Underscore (_) • Hyphen (-)
TriggerInput	Range: [0] Generic Trigger input number. Range: [1-6] Trigger input number on device. Range: [1-10] Window number for motion detection.

Table 5-5 *Event Clip Creation Notification XML Elements (continued)*

XML Element	Description
ProxyName	The proxy name for the event. The ProxyName element may contain 1 to 256 of the following characters: • Digits (0 to 9) • Upper case letters (A to Z) • Lower case letters (a to z) • Underscore (_) • Hyphen (-) The reserved value is -1. Note The proxy must exist before adding an event trigger.
ArchiveName	The archive name for this event. The ArchiveName element may contain 1 to 256 of the following characters: • Digits (0 to 9) • Upper case letters (A to Z) • Lower case letters (a to z) • Underscore (_) • Hyphen (-) The reserved value is -1.
StartUTC	Format: [UTC milliseconds] Date of the event in UTC milliseconds. This date is when VSMS received notification of the event from the encoder.
Duration	Format: [integer] Duration of the event archive in seconds. Note The only supported duration is 300 seconds.
StopMode	Reserved values: [auto manual] Specifies whether the clip is auto-stopped or manually stopped.
Type	Reserved values: [motion alarm]. Specifies whether the type of event is motion detection or trigger.

Event Clip Start/Stop

`http://host/event.bwt?command=event&name=triggerName&type=startStop`

Purpose

Starts or stops event-based clips.

Required Fields	<i>host</i>	IP address or hostname (<i>hostname.domain</i>) where VSMS is running. By default, VSMS runs on port 80 (HTTP), however, you can use an alternate port, such as port 8080. For example, to specify port 8080, use <i>host:8080</i> .
	command=event	Event command. The event keyword associates the command with an event action. The event keyword is a reserved value.
	name=triggerName	The name for the event. The <i>triggerName</i> element may contain 1 to 32 of the following characters: • Digits (0 to 9) • Upper case letters (A to Z) • Lower case letters (a to z) • Underscore (_) • Hyphen (-) The reserved value is -1.
	type=startStop	Start/stop type where <i>startStop</i> specifies whether to start or stop a clip. Valid <i>startStop</i> values can be one of the following reserved keywords: • start—Starts the clip. • stop—Stops the clip. The default value is start . No other values are supported.

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
```

Usage Guidelines

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

Examples

The following example stops the clip that was started when the event named lobbyMotion was triggered:

```
http://vsms.cisco.com/event.bwt?command=event&name=lobbyMotion&type=stop
```

Get Event Information

**`http://host/info.bwt?type=event&property=propValue&name=eventName&startutc=utcMs
&stoputc=utcMs&display=dispFormat`**

Purpose

Retrieves event information.

Required Fields

<i>host</i>	IP address or hostname (<i>hostname.domain</i>) where VSMS is running. By default, VSMS runs on port 80 (HTTP), however, you can use an alternate port, such as port 8080. For example, to specify port 8080, use <i>host:8080</i> .
type=event	Event type. The event keyword associates the type with an event action. The event keyword is a reserved value.
property=propValue	Property type where <i>propValue</i> specifies which event information to retrieve. Valid <i>propValue</i> values can be one of the following reserved keywords: setup—List all events set up on the VSMS host being queried. proxies—List all events, and the proxies they use to record event archives. Events with multiple proxies are listed per proxy. archives—List each event trigger received, including any event archives that were requested. An event set up with trigger tracking is returned without archive clip names.

Optional Fields

name=eventName	The name of the event being queried. If no name is given, VSMS returns all events. The <i>eventName</i> value may contain 1 to 32 of the following characters: - Digits (0 to 9) - Upper case letters (A to Z) - Lower case letters (a to z) - Underscore (_) - Hyphen (-) The reserved value is -1.
startutc=utcMs	Start date filter for archives. The <i>utcMs</i> value is the start date in UTC milliseconds. Note This field is used only when property=archives is specified.

stoputc=utcMs	Stop date filter for archives. The <i>utcMs</i> value is the stop date in UTC milliseconds.
Note This field is used only when property=archives is specified.	
display=dispFormat	Display format where <i>dispFormat</i> specifies the format to use when displaying the list of running proxies. Valid <i>dispFormat</i> values include the following keywords: html—Hypertext markup language format text—Plain text format ssv—Space-separated value format The default <i>dispFormat</i> value is 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
```

Usage Guidelines

Information can be retrieved based on property type, event name, start and stop dates.

Examples

Retrieving All Event Archive Information

The following example retrieves information about all event archives:

```
http://vsms.cisco.com/info.bwt?type=event&property=archives
```

Retrieving Archive Information for a Specific Event

The following example retrieves information about the event archives for the event named abc:

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

Retrieving Archive Information for a Specific Event Within a Specific Time Frame

The following example retrieves information about the event archives for the event named abc that start on or after 1018642188000 UTC and end on or before 1018642228000 UTC:

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

Retrieving Archive Information in SSV Format for a Specific Event Within a Specific Time Frame

The following example retrieves information in SSV format about all event archives for events named abc that start on or after 101864218800 UTC and end on or before 1018642228000 UTC:

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


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
```

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.

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.
-



CHAPTER 6

AxClient API Methods

This chapter lists the methods in the AxClient API, provides detailed information about each method, provides examples for using the methods. It includes the following topics:

- [Methods for Controlling Video Operations, page 6-2](#)
- [Methods for Obtaining Information about the AxClient or Video Streams, page 6-25](#)
- [Methods for Creating Clips and Snapshots, page 6-48](#)
- [Methods for Controlling VMR Display, page 6-57](#)
- [Methods for Setting up Callbacks, page 6-75](#)

Methods for Controlling Video Operations

The following sections describe the methods that provide functionality for controlling a video feed.

Table 6-1 *API Method Summary*

Method Name	Description
mtStartStream , page 6-3	Loads the designated live or archived video stream into the AxCliet and starts playing the stream.
mtStreamStarting , page 6-5	Returns the status of the video stream started by the mtStartStream method.
mtStartStreamWait , page 6-7	Pauses the AxCliet for a specified number of milliseconds after invoking the mtStartStream method, before invoking another method, or until the mtStartStream command is finished, whichever occurs first.
playForward , page 6-8	Plays the started archive video stream in the forward direction.
playRewind , page 6-10	Plays the started archive video stream in the reverse direction.
stepForward , page 6-11	Moves the loaded archive video stream one frame in the direction of the current playback.
stepRewind , page 6-12	Moves the loaded archive video stream one frame in the opposite direction of the current playback.
pause , page 6-13	Pauses the playback of an archive or pauses a live source on the current frame.
playResume , page 6-14	Starts playing a previously paused stream and continues in the previous direction of play.
stop , page 6-15	Stops streaming the live or archived video.
close , page 6-16	Stops streaming the feed or archive and disconnects the AXCliet from the VSMS host. Also flushes and resets source properties.
setPlayrateEx , page 6-17	Specifies the playback rate for a loaded archive.
repeatUTCsegment , page 6-18	Plays a designated segment of an archive repeatedly (loops the segment).
seekToUTCtime , page 6-19	Seeks to the specified time in an archive. Time is stored as seconds since 1970.
showTimestamp , page 6-21	Shows or hides the timestamp overlay when playing a video source.
addToSync , page 6-22	Allows a client playback window to perform the identical operations that other windows are performing.
removeFromSync , page 6-23	Removes a client playback window from sync control.
createSyncId , page 6-24	Creates a string that can be used for client synchronization.

mtStartStream

```
HRESULT mtStartStream (
    String source,
    int version,
    bool isProxy);
```

Purpose Loads the designated live or archived video stream into the AxClient and starts playing the stream.

Arguments	<i>source</i>	The URI of the video feed.
	<i>version</i>	The server version (for all 6.X releases the value is 6).
	<i>isProxy</i>	Determines whether source is from a proxy or archive.
		The <i>isProxy</i> argument can be one of the following keywords: • VARIANT_TRUE if the source is from a proxy. • VARIANT_FALSE if the source is from an archive.

Return Values HRESULT S_OK/E_FAIL

Exceptions None.

Events Fired onStartOfStream, onMtStartStreamDone.

Notes This method is equivalent to calling both **switchTo()** and **playForward()** in previous versions of the API. Therefore, **playForward()** does not need to be called **after mtStartStream()** is invoked.

This call does not immediately return success or failure: instead, the **mtStartStreamDone** and **onStartofStream** events must be caught. Only once both events are caught can the application be certain a stream is playing. If only the **mtStartStreamDone** event is caught (and no **onStartofStream** fires) then the client finished with the *mtStartStream* call, but the server did not return a stream to play.

Examples

C# Example

```
// Wrap the AxClient with AXImp and include the wrapped ActiveX dll in your project
try {
    this.axc.mtStartStream(
        bwims://1.1.1.1/p_s1_CiscoHDCamera_1, 6, true);
}
catch(Exception ex) {
    this.logger.subLog("Exception in mtStartStream - " + ex.Message);
}
```

JavaScript example

```
/* Embed the AxClient as an object, and then use this syntax where the embedded object ID is IMC1. */
```

```
var axc = document.applets["IMC1"];  
axc.mtStartStream(bwims://1.1.1.1/p_sl_CiscoHDCamera_1, 6, true);
```

Related methods

[close](#), page 6-16

[mtStreamStarting](#), page 6-5

[mtStartStreamWait](#), page 6-7

[setOnEndOfStream](#), page 6-76

[setOnMtStartStreamDone](#), page 6-77

[setOnStartOfStream](#), page 6-84

mtStreamStarting

HRESULT mtStreamStarting (VARIANT_BOOL *pVal);

Purpose	Returns the status of the video stream started by the mtStartStream method.	
Arguments	<i>pVal</i>	Indicates whether the stream is starting. The <i>pVal</i> argument can be one of the following keywords: • VARIANT_TRUE if the stream is starting. • VARIANT_FALSE if the stream is not starting, such as when the stream was already started or was not started.
Return Values	HRESULT S_OK/E_FAIL	
Exceptions	None.	
Events Fired	onStartOfStream.	
Notes	This API method is useful in situations where certain functionality must be performed only after a stream has started. This approach requires the client to perform work to return the state information, however, and does not work well when thread blocking occurs. For example, in the case of a C# application using VMR, the threading model employed may not allow this call to function at all. The more optimal approach is to catch the mtStartStreamDone event in the application, which is fired by the client at the same time this Boolean changes state internally (for example, the same information can be caught instead of polled).	
Examples	C# Example <pre>if (!this.axc.mtStreamStarting()) { // Use other AxClient APIs to modify the stream. }</pre> JavaScript Example <pre>if(!axc.mtStreamStarting()) { //call other Axclient APIs} else { //error out}</pre>	
Related Methods	mtStartStream, page 6-3 mtStartStreamWait, page 6-7 setOnMtStartStreamDone, page 6-77	

[setOnMtStartStreamDone, page 6-77](#)

[setOnStartOfStream, page 6-84](#)

mtStartStreamWait

HRESULT mtStartStreamWait (ULONG msec);

Purpose Pauses the AxClient for a specified number of milliseconds after invoking the **mtStartStream** method, before invoking another method, or until the **mtStartStream** command is finished, whichever occurs first.

Arguments

<i>msec</i>	The number of milliseconds to wait before invoking another method.
-------------	--

Return Values HRESULT S_OK/E_FAIL

Exceptions None.

Events Fired None.

Notes You can use the **mtStartStreamWait** and the **mtStreamStarting** methods to determine whether to wait before invoking another method. If the **mtStreamStarting** indicates that a stream is starting, you can use the **mtStartStreamWait** to cause the system to wait a few seconds before invoking another method.

Examples

C# Example

```
try {
    this.axc.mtStartStream(
        bwims://1.1.1.1/p_sl_CiscoHDCamera_1, 6, true);
} catch (Exception ex) {
    this.logger.subLog("Exception in mtStartStream - " + ex.Message);
}

this.axc.mtStartStreamWait(2000);

// Use other AxClient APIs to modify the stream.
```

JavaScript Example

```
if (axc.mtStreamStarting()) {
    axc.mtStartStreamWait(2000);
}

// Use other AxClient APIs to modify the stream.
```

Related Methods

- [mtStartStream, page 6-3](#)
- [mtStreamStarting, page 6-5](#)
- [setOnStartOfStream, page 6-84](#)

playForward

HRESULT playForward (void);

Purpose Plays the started archive video stream in the forward direction.

Arguments This method has no arguments.

Return Values HRESULT S_OK

Exceptions None.

Events Fired OnStateChanged

Notes This method applies only to archive video streams.

Examples

C# Example

```
try
{
    if (axc.getState() == "0")
    {
        axc.playForward();
    }
    else
    {
        //already streaming..
    }
}
catch (Exception error)
{
    this.logger.callbackLog("exception" + er
}
```

JavaScript Example

```
axc.playForward();
```

Related Methods

[playRewind, page 6-10](#)

[stepForward, page 6-11](#)

[stepRewind, page 6-12](#)

[pause, page 6-13](#)

[playResume, page 6-14](#)

[stop](#), page 6-15

[setOnEndOfStream](#), page 6-76

playRewind

HRESULT playRewind (void);

Purpose

Plays the started archive video stream in the reverse direction.

**Note**

This API method does not work with MPEG2 archives or archives associated with Bosch cameras.

Arguments

This method has no arguments.

Return Values

HRESULT S_OK

Exceptions

None.

Events Fired

onStateChanged

Notes

This method applies only to archive video streams.

Examples**C# Example**

```
this.axc.playRewind();
```

JavaScript Example

```
axc.playRewind();
```

Related Methods

[playForward](#), page 6-8
[stepForward](#), page 6-11
[stepRewind](#), page 6-12
[pause](#), page 6-13
[playResume](#), page 6-14
[stop](#), page 6-15

stepForward

HRESULT stepForward (void);

Purpose

Moves the loaded archive video stream one frame in the direction of the current playback.

**Note**

This API method does not work with MPEG2 archives or archives associated with Bosch cameras.

Arguments

This method has no arguments.

Return Values

HRESULT S_OK/E_FAIL

Exceptions

None.

Events Fired

onStateChanged

Notes

This method applies only to archive video streams.

The step forward movement is in the same direction as the current play direction. For example, calling the [playRewind](#) method followed by the [stepForward](#) method moves the stream one frame backward.

Examples**C# Example**

```
this.axc.stepForward();
```

JavaScript Example

```
axc.stepForward();
```

Related Methods

[playForward](#), page 6-8

[playRewind](#), page 6-10

[stepRewind](#), page 6-12

[pause](#), page 6-13

[playResume](#), page 6-14

[stop](#), page 6-15

stepRewind

HRESULT stepRewind (void);

Purpose

Moves the loaded archive video stream one frame in the opposite direction of the current playback.



Note

This API method does not work with MPEG2 archives or archives associated with Bosch cameras.

Arguments

This method has no arguments.

Return Values

HRESULT S_OK/E_FAIL

Exceptions

None.

Events Fired

onStateChanged

Notes

This method applies only to archive video streams.

Operates in the reverse direction of the current play direction. For example, issuing **playRewind()** then **stepRewind()** moves the stream one step forward.

Examples

C# Example

```
this.axc.stepRewind();
```

JavaScript Example

```
axc.stepRewind();
```

Related Methods

[playForward](#), page 6-8
[playRewind](#), page 6-10
[stepForward](#), page 6-11
[pause](#), page 6-13
[playResume](#), page 6-14
[stop](#), page 6-15

pause

HRESULT pause (void);

Purpose	Pauses the playback of an archive or pauses a live source on the current frame.
Arguments	This method has no arguments.
Return Values	HRESULT S_OK/E_FAIL
Exceptions	None.
Events Fired	onStateChanged
Notes	The application must reload the stream (for example, call <code>mtStartStream</code> again) if the stream has been paused for more than 15 minutes. After 15 minutes of pause, the stream will no longer be loaded in memory and no subsequent API calls will have any effect on it.
Examples	C# Example <pre>this.axc.pause();</pre> JavaScript Example <pre>axc.pause();</pre>
Related Methods	playForward, page 6-8 playRewind, page 6-10 stepForward, page 6-11 stepRewind, page 6-12 playResume, page 6-14 stop, page 6-15 close, page 6-16

playResume

HRESULT playResume (void);

Purpose	Starts playing a previously paused stream and continues in the previous direction of play.
Arguments	This method has no arguments.
Return Values	HRESULT S_OK/E_FAIL
Exceptions	None.
Events Fired	onStateChanged
Notes	The application must reload the stream (for example, call mtStartStream again) if the stream has been paused for more than 15 minutes. After 15 minutes of pause, the stream will no longer be loaded in memory and no subsequent API calls will have any effect on it.
Examples	C# Example <pre>this.axc.playResume();</pre> JavaScript Example <pre>axc.playResume();</pre>
Related Methods	playForward, page 6-8 playRewind, page 6-10 stepForward, page 6-11 stepRewind, page 6-12 pause, page 6-13 stop, page 6-15

stop

HRESULT stop (void);

Purpose Stops streaming the live or archived video.

Arguments This method has no arguments.

Return Values HRESULT S_OK/E_FAIL

Exceptions None.

Events Fired onStateChanged

Notes We do not recommend that you use this method. Use the close() method instead.

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. Because of the work associated with the buffer, stop is not recommended if the application will resume playback. If playback resumes within 15 minutes, pause is recommended instead.

Examples

C# Example

```
this.axc.stop();
```

JavaScript Example

```
axc.stop();
```

Related Methods

- [playForward, page 6-8](#)
- [playRewind, page 6-10](#)
- [stepForward, page 6-11](#)
- [stepRewind, page 6-12](#)
- [pause, page 6-13](#)
- [playResume, page 6-14](#)

close

HRESULT close (void)

Purpose Stops streaming the feed or archive and disconnects the AXclient from the VSMS host. Also flushes and resets source properties.

Arguments This method has no arguments.

Return Values HRESULT S_OK/E_FAIL

Exceptions None.

Events Fired onStateChanged

Notes To reload the stream after calling the close method, you must use the [mtStartStream](#) method. Because of the work associated with flushing the video buffer, we do not recommend using this method if the same video stream is to be viewed again in a relatively short time frame; instead, we recommend that you use the [pause](#) method. This method does not have to be called if the client intends to call the [mtStartStream](#) method for another source in the next moment (for example, changing from one source to another). When the next [mtStartStream](#) method is initiated, the same buffer flush takes place, so placing a close at the end of a viewing session, right before the next session starts (in the same client), duplicates work.

Examples

C# Example

```
this.axc.close();
```

JavaScript Example

```
axc.close();
```

Related Methods [mtStartStream](#), page 6-3
[pause](#), page 6-13

setPlayrateEx

HRESULT setPlayrateEx (Long *playRate*);

Purpose Specifies the playback rate for a loaded archive.

Arguments

<i>playRate</i>	The rate at which a loaded archive is to play. Valid values are 0.05, 0.10, 0.25, 0.50, 0.75, 0.80, 1, 2, 4, 8, 16, 32, and 64: • A value of 1 is the normal play rate. • A value less than 1 is a slower play rate. • A value greater than 1 is a faster play rate.
-----------------	--

Return Values HRESULT S_OK/E_FAIL

Exceptions None.

Events Fired onPlayrateChanged

Notes This method replaces the **setPlayrate** method, which should no longer be used.

Examples

C# Example

```
this.axc.setPlayRateEx(8);
```

JavaScript Example

```
axc.setPlayRateEx(8);
```

Related Methods [getPlayrateEx, page 6-37](#)
[setOnPlayrateChanged, page 6-79](#)

repeatUTCSegment

```
HRESULT repeatUTCSegment (  
    DOUBLE seekTime,  
    LONG startOffset,  
    LONG endOffset);
```

Purpose Plays a designated segment of an archive repeatedly (loops the segment).

Arguments	<i>seekTime</i>	Start time of the segment in UTC format
	<i>startOffset</i>	Specifies how many seconds before the time that seekTime designates the loop to start playing.
	<i>endOffset</i>	Specifies how many seconds after the time that seekTime designates the loop to stop playing.

Return Values HRESULT S_OK/E_FAIL

Exceptions None.

Events Fired onStateChanged

Notes This method seeks the specified seektime in an archive and repeatedly plays the archive segment bounded by *seekTime* – *startOffset* and *seekTime* – *endOffset*. Units are in seconds. The *endOffset* argument is a positive number.

Examples

C# Example

```
this.axc.repeatUTCSegment(12345,1,120);
```

JavaScript Example

```
axc.repeatUTCSegment(12345,1,120);
```

Related Methods

- [seekToUTCTime](#), page 6-19
- [seekToPercentage](#), page 6-20
- [setOnSeekTimeChanged](#), page 6-83
- [setOnStartTimeChanged](#), page 6-85
- [setOnStopTimeChanged](#), page 6-89

seekToUTCTime

HRESULT seekToUTCTime (DOUBLE *time*);

Purpose	Seeks to the specified time in an archive. Time is stored as seconds since 1970.		
Arguments	<table> <tr> <td><i>time</i></td><td>Start time of the segment in UTC format, representing the number of seconds since January 1, 1970.</td></tr> </table>	<i>time</i>	Start time of the segment in UTC format, representing the number of seconds since January 1, 1970.
<i>time</i>	Start time of the segment in UTC format, representing the number of seconds since January 1, 1970.		
Return Values	HRESULT S_OK/E_FAIL		
Exceptions	None.		
Events Fired	onStateChanged		
Notes	Seeking should not be performed before the onStartOfStream event has been received. Wait for the onStartofStream event to fire before calling this API. This method replaces the seek() method, which should no longer be used and may not function as expected.		
Examples	C# Example <pre>double startTime = 0; startTime = axc.getUTCStartTime(); axc.seekToUTCTime(startTime);</pre> JavaScript Example <pre>axc.seekToUTCTime(12345678);</pre>		
Related Methods	repeatUTCsegment, page 6-18 seekToPercentage, page 6-20 getUTCStartTime, page 6-29 setOnSeekTimeChanged, page 6-83 setOnStartTimeChanged, page 6-85 setOnStopTimeChanged, page 6-89		

seekToPercentage

HRESULT seekToPercentage (FLOAT *percent*);

Purpose	Seeks to the specified percentage in an archive.	
Arguments	<i>percent</i>	Percentage of the total time of the archive. Valid values are in decimal format between 0 and 1.
Return Values	HRESULT S_OK/E_FAIL	
Exceptions	None.	
Events Fired	onSeekTimeChanged	
Notes	Seeking should not be performed before the onStartOfStream event has been received. Wait for the onStartofStream event to fire before calling this API. This method replaces the seek() method, which should no longer be used and may not function as expected.	
Examples	C# Example <pre>axc.seekToPercentage(0.56);</pre> JavaScript Example <pre>axc.seekToPercentage(0.56);</pre>	
Related Methods	repeatUTCsegment , page 6-18 seekToUTCTime , page 6-19 getUTCStartTime , page 6-29 setOnSeekTimeChanged , page 6-83 setOnStartTimeChanged , page 6-85 setOnStopTimeChanged , page 6-89	

showTimestamp

HRESULT showTimestamp (VARIANT_BOOL show);

Purpose

Shows or hides the timestamp overlay when playing a video source.

The timestamp overlay displays the time associated with each frame of the stream. This call can only be made after the stream is playing. Calling showTimestamp before the stream is actually playing will result in an error. This API only applies to MPEG2 sources.

Arguments

show

Controls whether the timestamp displays:

- **VARIANT_TRUE**—Timestamp displays.
 - **VARIANT_FALSE**—Timestamp does not display.
-

Return Values

HRESULT S_OK/E_FAIL

Exceptions

None.

Events Fired

None.

Notes

By default, timestamps are not displayed.

Examples**C# Example**

```
// This example uses a UI checkbox to determine if timestamp should be on or off,  
// and then executes that choice once the onStartofStream event has fired  
// (timestamp cannot be set prior to this point).
```

```
protected void axc_OnStartOfStream(  
    object sender,  
    _IMediaPlayerCtrlEvents_OnStartOfStreamEvent e)  
{  
    if (this.timestampOption.Checked) {  
        axc.showTimestamp(true);  
    } else if (!this.timestampOption.Checked) {  
        axc.showTimestamp(false);  
    }  
}
```

JavaScript Example

```
axc.showTimestamp(true);
```

Related Methods

None.

addToSync

```
HRESULT addToSync (
    BSTR aSyncId,
    SHORT aCount);
```

Purpose

Allows a client playback window to perform the identical operations that other windows are performing. This is helpful when viewing a number of archives for a given time period and you desire all the archives to shuttle through the archive set in the same manner (for example, every window starts playing at the same seek point, every window pauses at the same time, etc.).

Arguments

<i>aSyncId</i>	A value against which clients register their UI behavior. Any distinct string is allowed.
<i>aCount</i>	A value greater than 0. Providing a value of 0 will tell the client not to sync against the aSyncId, which is the incorrect way to stop sync from occurring. If you want to stop the sync of a given client, call removeFromSync instead.

Return Values

HRESULT S_OK/E_FAIL

Exceptions

None.

Events Fired

None.

Notes

This API requires user intervention during the execution of the request. For a headless API to perform the clip save, use the **save** method instead.

For VSM 6.3.2 or higher, this method is supported in both JavaScript and C#; for VSM 6.3, this method is supported only in JavaScript.

Examples

C# Example

```
axc.addToSync ("12345678", 1);
```

JavaScript Example

```
axc.addToSync ("12345678", 1);
```

Related Methods

[removeFromSync](#), page 6-23

[createSyncId](#), page 6-24

removeFromSync

HRESULT removeFromSync ();

Purpose

Removes a client playback window from sync control.

Sync control performs the identical operations that other windows are performing, which is helpful when viewing a number of archives for a given time period and you desire all the archives to shuttle through the archive set in the same manner (for example, every window starts playing at the same seek point, every window pauses at the same time, etc.).

Arguments

This method has no arguments.

Return Values

HRESULT S_OK/E_FAIL

Exceptions

None.

Events Fired

None.

Notes

This API requires user intervention during the execution of the request. For a headless API to perform the clip save, use the **save** method instead.

For VSM 6.3.2 or higher, this method is supported in both JavaScript and C#; for VSM 6.3, this method is supported only in JavaScript.

Examples**C# Example**

```
axc.removeFromSync();
```

JavaScript Example

```
axc.removeFromSync();
```

Related Methods

[addToSync](#), page 6-22

[createSyncId](#), page 6-24

createSyncId

HRESULT addToSync (BSTR *aSyncId);

Purpose	Creates a string that can be used for client synchronization.	
Arguments	<i>aSyncId</i>	A value against which clients register their UI behavior. Any distinct string is allowed.
Return Values	HRESULT S_OK/E_FAIL	
Exceptions	None.	
Events Fired	None.	
Notes	Not recommended for creating a unique string. A GUID generator produces a more unique string and is recommended instead.	
Examples	C# Example <pre>string aSyncId = axc.createSyncId();</pre> JavaScript Example <pre>string aSyncId = axc.createSyncId();</pre>	
Related Methods	addToSync, page 6-22 removeFromSync, page 6-23	

Methods for Obtaining Information about the AxCliEnt or Video Streams

The following sections describe the methods that provide functionality for obtaining information about the AxCliEnt and about video streams.

Table 6-2 **API Method Summary**

Method Name	Description
getCiscoHD , page 6-26	Indicates whether the stream comes from a Cisco high definition IP camera.
getVersion , page 6-27	Retrieves the version number of the AxCliEnt.
getUTCSeekTime , page 6-28	Retrieves the seek time in UTC format.
getUTCStartTime , page 6-29	Retrieves the start time of an archive in UTC format.
getUTCStopTime , page 6-30	Retrieves the last frame time of the archive, in UTC format.
getUTCCurrentTime , page 6-31	Retrieves the current time in an archive, in UTC format.
getUTCOriginalStartTime , page 6-32	Retrieves the actual start time for a loop archive, in UTC format.
getState , page 6-33	Retrieves the state of the player.
getContentType , page 6-34	Retrieves the media type of a loaded source.
getCurrentSource , page 6-35	Retrieves the URL of the currently loaded source.
getRecordrateEx , page 6-36	Retrieves the rate at which the archive was recorded. Depending on the media type of the source, this value is the framerate or the bitrate.
getPlayrateEx , page 6-37	Retrieves the rate at which the loaded archive is playing.
getErrorText , page 6-38	Retrieves the text description of the specified error code.
getProfiles , page 6-39	Retrieves a list of the WMV profiles that the loaded video stream supports.
getProfilesSSV , page 6-40	Retrieves an array of strings that represent the WMV profiles that the loaded video stream supports.
getStreamCodecSubtype , page 6-41	Retrieves the subtype of the codec of the loaded stream.
getVideoWidth , page 6-42	Gets the width of the source stream, in pixels.
getVideoHeight , page 6-43	Gets the height of the source stream, in pixels.
getDisplayWidth , page 6-44	Gets the width in pixels at which the source should be displayed.
getDisplayHeight , page 6-45	Gets the height in pixels at which the source should be displayed.
getX , page 6-46	Retrieves the x coordinate of the center of the view rectangle.
getY , page 6-47	Retrieves the y coordinate of the center of the view rectangle.

getCiscoHD

HRESULT getCiscoHD ();

Purpose Indicates whether the stream comes from a Cisco high definition IP camera.

Arguments None.

Return Values HRESULT S_OK/E_FAIL

<i>pVal</i>	Camera type:
	0—Not a Cisco high definition IP camera 1—Cisco high definition IP camera

Exceptions None.

Events Fired None.

Notes None.

Examples

C# Example

```
int HD = axc.getCiacoHD()
```

JavaScript Example

```
var HD = axc.getCiacoHD()
```

Related Methods None.

getVersion

HRESULT getVersion (BSTR **version*);

Purpose Retrieves the version number of the AxClient.

Arguments	<i>version</i>	Version number in the format <i>n.n.n.n</i> , where <i>n</i> is an integer. For example, 6.2.16.1.
------------------	----------------	--

Return Values HRESULT S_OK/E_FAIL

Exceptions None.

Events Fired None.

Notes The version of imscclient.dll is returned.

Examples

C# Example

```
String myVersion = axc.getVersion();
```

JavaScript Example

```
var myVersion = axc.getVersion();
```

Related Methods None.

getUTCSeekTime

HRESULT getUTCSeekTime (DOUBLE *pTime*);

Purpose Retrieves the seek time in UTC format.

Arguments

<i>pTime</i>	Seek time in UTC format.
--------------	--------------------------

Return Values HRESULT S_OK/E_FAIL

Exceptions None.

Events Fired None.

Notes This method is not supported for live feeds.

Examples

C# Example

```
double mySeekTime axc.getUTCSeekTime();
```

JavaScript Example

```
var mySeekTime = axc.getUTCSeekTime();
```

Related Methods

[getUTCStartTime, page 6-29](#)

[getUTCStopTime, page 6-30](#)

[getUTCCurrentTime, page 6-31](#)

[getUTCOriginalStartTime, page 6-32](#)

getUTCStartTime

HRESULT getUTCStartTime (DOUBLE *ptime);

Purpose Retrieves the start time of an archive in UTC format.

Arguments

<i>ptime</i>	Start time in UTC format.
--------------	---------------------------

Return Values HRESULT S_OK/E_FAIL

Exceptions None.

Events Fired None.

Notes If the stream is paused, the AxClient does not receive an update for the start or stop times of a loop archive until it resumes playing. For looping archives, the start time of which a paused client is aware could be incorrect (until the client begins playing once again and receives the next start time update).

Examples **C# Example**

```
double startTime = 0;
startTime = axc.getUTCStartTime();
axc.seekToUTCTime(startTime);
```

JavaScript Example

```
var startTime = axc.getUTCStartTime();
axc.seekToUTCTime(startTime);
```

Related Methods

- [seekToUTCTime, page 6-19](#)
- [seekToPercentage, page 6-20](#)
- [getUTCSeekTime, page 6-28](#)
- [getUTCStopTime, page 6-30](#)
- [getUTCCurrentTime, page 6-31](#)
- [getUTCOriginalStartTime, page 6-32](#)

getUTCStopTime

HRESULT getUTCStopTime (DOUBLE *ptime);

Purpose	Retrieves the last frame time of the archive, in UTC format.		
Arguments	<table> <tr> <td><i>ptime</i></td><td>The last frame time of the archive, in UTC format.</td></tr> </table>	<i>ptime</i>	The last frame time of the archive, in UTC format.
<i>ptime</i>	The last frame time of the archive, in UTC format.		
Return Values	HRESULT S_OK/E_FAIL		
Exceptions	None.		
Events Fired	None.		
Notes	If the stream is paused, the AxClient does not receive an update for the start or stop times of a loop archive until it resumes playing. For looping archives, the start time of which a paused client is aware could be incorrect (until the client begins playing once again and receives the next start time update).		
Examples	C# Example <pre>double stopTime = 0; stopTime = axc.getUTCStopTime(); axc.seekToUTCtime(stopTime);</pre> JavaScript Example <pre>var stopTime = axc.getUTCStopTime(); axc.seekToUTCtime(stopTime);</pre>		
Related Methods	getUTCSeekTime, page 6-28 getUTCStartTime, page 6-29 getUTCCurrentTime, page 6-31 getUTCOriginalStartTime, page 6-32		

getUTCCurrentTime

HRESULT getUTCCurrentTime (Double *ptime);

Purpose Retrieves the current time in an archive, in UTC format.

Arguments

<i>ptime</i>	The current time of an archive, in UTC format.
--------------	--

Return Values HRESULT S_OK/E_FAIL

Exceptions None.

Events Fired None.

Notes If the stream is paused, the AxClient does not receive an update for the start or stop times of a loop archive until it resumes playing. For looping archives, the start time of which a paused client is aware could be incorrect (until the client begins playing once again and receives the next start time update).

Examples

C# Example

```
double currentTime = 0;
currentTime = axc.getUTCCurrentTime();
axc.seekToUTCtime(currentTime);
```

JavaScript Example

```
var currentTime = axc.getUTCCurrentTime();
```

Related Methods

- [getUTCSeekTime, page 6-28](#)
- [getUTCStartTime, page 6-29](#)
- [getUTCStopTime, page 6-30](#)
- [getUTCOriginalStartTime, page 6-32](#)

getUTCOriginalStartTime

HRESULT getUTCOriginalStartTime (DOUBLE *ptime);

Purpose Retrieves the actual start time for a loop archive, in UTC format.

Arguments

<i>ptime</i>	Original start time of an archive, in UTC format.
--------------	---

Return Values HRESULT S_OK/E_FAIL

Exceptions None.

Events Fired None.

Notes If the stream is paused, the AxClient does not receive an update for the start or stop times of a loop archive until it resumes playing. For looping archives, the start time of which a paused client is aware could be incorrect (until the client begins playing once again and receives the next start time update).

Examples

C# Example

```
double startTime = axc.getUTCOriginalStartTime();
```

JavaScript Example

```
var startTime = axc.getUTCOriginalStartTime();
```

Related Methods

- [getUTCSeekTime, page 6-28](#)
- [getUTCStartTime, page 6-29](#)
- [getUTCStopTime, page 6-30](#)
- [getUTCCurrentTime, page 6-31](#)

getState

HRESULT getState (**BSTR** *state);

Purpose

Retrieves the state of the player.

Arguments

<i>state</i>	The current state of a the player: • 0—Paused • 1—Playing forward • 2—Playing reverse • 3—Seeking • 4—Loading • 5—Stopped • 6—First frame received • -1—(Negative 1) Unknown
--------------	--

Return Values

HRESULT S_OK/E_FAIL

Exceptions

None.

Events Fired

None.

Notes

Based on the called control operation, the state of the AXclient changes, When the state changes, the AX client also fires the **onStateChange** event.

Examples**C# Example**

```
string playerState = axc.getState();
```

JavaScript Example

```
var playerState = axc.getState();
```

Related Methods

[Methods for Controlling Video Operations, page 6-2](#)

getContentType

HRESULT getContent (BSTR *state);

Purpose Retrieves the media type of a loaded source.

Arguments	state	Media type of the source:
		• MPEG1 • MPEG2 • MPEG4 • JPEG • H264 • audio

Return Values HRESULT S_OK/E_FAIL

Exceptions None.

Events Fired None.

Notes None.

Examples

C# Example

```
string playerContent = axc.GetContentType();
```

JavaScript Example

```
var playerContent = axc.GetContentType();
```

Related Methods

[getCurrentSource](#), page 6-35

[getStreamCodecSubtype](#), page 6-41

getCurrentSource

HRESULT getCurrentSource (BSTR *pSource);

Purpose Retrieves the URL of the currently loaded source.

Arguments

<i>pSource</i>	Video source in the format <i>//host/source</i> , where: <i>host</i>—Hostname or IP address of the VMSM host on which the audio source resides <i>source</i>—name of the video source
----------------	---

Return Values HRESULT S_OK/E_FAIL

Exceptions None.

Events Fired None.

Notes None.

Examples

C# Example

```
string playerSource = axc.getCurrentSource();
```

JavaScript Example

```
var playerSource = axc.getCurrentSource();
```

Related Methods [getContentType](#), page 6-34

getRecordrateEx

HRESULT getRecordrateEx (DOUBLE *recordRate);

Purpose Retrieves the rate at which the archive was recorded. Depending on the media type of the source, this value is the framerate or the bitrate.

Arguments

<i>recordRate</i>	The rate at which the archive was recorded.
-------------------	---

Return Values HRESULT S_OK/E_FAIL

Exceptions None.

Events Fired None.

Notes The information that this method provided depends on the media type of the source. For JPEG sources, this value is the framerate. For MPEG sources, this value is the bitrate.

This method does not apply to live feeds. Live feeds return -1.



Caution

The frame rate returned by this API (when reporting on a JPEG source) may only represent the frame rate at the client, not the true frame rate at the server.

Examples

C# Example

```
double myFrameRate = axc.getRecordrateEx();
```

JavaScript Example

```
var myFrameRate = axc.getRecordrateEx();
```

Related Methods [getPlayrateEx](#), page 6-37

getPlayrateEx

HRESULT getPlayrateEx (DOUBLE *playRate);

Purpose Retrieves the rate at which the loaded archive is playing.

Arguments

<i>playRate</i>	The rate at which the loaded archive is playing. Valid values are 0.05, 0.10, 0.25, 0.50, 0.75, 0.80, 1, 2, 4, 8, 16, 32, and 64: • A value of 1 is the normal play rate. • A value less than 1 is a slower play rate. • A value greater than 1 is a faster play rate.
-----------------	--

Return Values HRESULT S_OK/E_FAIL

Exceptions None.

Events Fired None.

Notes None.

Examples **C# Example**

```
double myFrameRate = axc.getRecordrateEx();
```

JavaScript Example

```
var myFrameRate = axc.getRecordrateEx();
```

Related Methods

- [setPlayrateEx, page 6-17](#)
- [getRecordrateEx, page 6-36](#)
- [setOnPlayrateChanged, page 6-79](#)

getErrorText

```
HRESULT getErrorText (
    VARIANT errorCode,
    BSTR* errorText);
```

Purpose Retrieves the text description of the specified error code.

Arguments	<i>errorCode</i>	The numerical value of the error code.
	<i>errorText</i>	The text description of the error.

Return Values HRESULT S_OK/E_FAIL

Exceptions None.

Events Fired None.

Notes None.

Examples

C# Example

```
string strError = axc.getErrorText();
```

JavaScript Example

```
var error = axc.getErrorText();
```

Related Methods None.

getProfiles

```
HRESULT getProfiles (  
    BSTR source,  
    VARIANT *profiles);
```

Purpose	Retrieves a list of the WMV profiles that the loaded video stream supports.
----------------	---

Arguments	<i>source</i>	Video source from which to obtain the list of WMV profiles.
	<i>profiles</i>	List of WMV profiles.

Return Values	HRESULT S_OK/E_FAIL
----------------------	---------------------

Exceptions	None.
-------------------	-------

Events Fired	None.
---------------------	-------

Notes	CBR means constant bitrate and VBR means variable bitrate.
--------------	--

Examples	C# Example <pre>string[] profiles = axc.getProfiles("bwims://10.10.100.200/NorthEntrance");</pre>
	JavaScript Example <pre>var profiles = axc.getProfiles(); var profileArray = profile.toArray("bwims://10.10.100.200/NorthEntrance");</pre>

Related Methods	getProfilesSSV , page 6-40
------------------------	--

getProfilesSSV

```
HRESULT getProfilesSSV (
    BSTR source,
    BSTR* profiles);
```

Purpose	Retrieves an array of strings that represent the WMV profiles that the loaded video stream supports.	
Arguments	<i>source</i>	Video source from which to obtain the array of strings.
	<i>profiles</i>	Array of strings that represent the WMV profiles.
Return Values	HRESULT S_OK/E_FAIL	
Exceptions	None.	
Events Fired	None.	
Notes	CBR means constant bitrate and VBR means variable bitrate.	
Examples	C# Example <pre>string profiles = axc.getProfilesSSV("bwims://10.10.100.200/NorthEntrance");</pre>	
	JavaScript Example <pre>var profiles = axc.getProfilesSSV(); var profileArray = profile.toArray("bwims://10.10.100.200/NorthEntrance");</pre>	
Related Methods	getProfiles , page 6-39	

getStreamCodecSubtype

HRESULT getStreamCodecSubtype (INT **subtype*);

Purpose Retrieves the subtype of the codec of the loaded stream.

Arguments	<i>subtype</i>	Subtype of the codec: • 0—JPEG • 6—MPEG4 • 11—H.264
------------------	----------------	---

Return Values HRESULT S_OK/E_FAIL

Exceptions None.

Events Fired None.

Notes None.

Examples

C# Example

```
int streamTypeInt = axc.getStreamCodecSubtype();
```

JavaScript Example

```
var streamType = axc.getStreamCodecSubtype();
```

Related Methods [getContentType](#), page 6-34

getVideoWidth

HRESULT getVideoWidth (SHORT *width);

Purpose	Gets the width of the source stream, in pixels.		
Arguments	<table> <tr> <td><i>width</i></td><td>Width of the source stream, in pixels.</td></tr> </table>	<i>width</i>	Width of the source stream, in pixels.
<i>width</i>	Width of the source stream, in pixels.		
Return Values	HRESULT S_OK/E_FAIL		
Exceptions	None.		
Events Fired	None.		
Notes	The actual source width may differ from the width that the source should be displayed.		
Examples	C# Example <pre>short videoW= axc.getVideoWidth();</pre> JavaScript Example <pre>var videoW = axc.getVideoWidth();</pre>		
Related Methods	getVideoHeight, page 6-43 getDisplayWidth, page 6-44 getDisplayHeight, page 6-45		

getVideoHeight

HRESULT getVideoHeight (SHORT *height);

Purpose	Gets the height of the source stream, in pixels.
----------------	--

Arguments	<table><tr><td><i>height</i></td><td>Height of the source stream, in pixels.</td></tr></table>	<i>height</i>	Height of the source stream, in pixels.
<i>height</i>	Height of the source stream, in pixels.		

Return Values	HRESULT S_OK/E_FAIL
----------------------	---------------------

Exceptions	None.
-------------------	-------

Events Fired	None.
---------------------	-------

Notes	The actual source height may differ from the height that the source should be displayed.
--------------	--

Examples	C# Example <pre>short videoH = axc.getVideoHeight();</pre>
-----------------	--

	JavaScript Example <pre>var videoH = axc.getVideoHeight();</pre>
--	--

Related Methods	getVideoWidth, page 6-42 getDisplayWidth, page 6-44 getDisplayHeight, page 6-45
------------------------	--

getDisplayWidth

HRESULT getDisplayWidth (SHORT *width);

Purpose Gets the width in pixels at which the source should be displayed.

Arguments

<i>width</i>	Width of the source stream, in pixels.
--------------	--

Return Values HRESULT S_OK/E_FAIL

Exceptions None.

Events Fired None.

Notes The actual source width may differ from the width that the source should be displayed.

Examples

C# Example

```
short displayW = axc.getDisplayWidth();
```

JavaScript Example

```
var displayW = axc.getDisplayWidth();
```

Related Methods

- [getVideoWidth](#), page 6-42
- [getVideoHeight](#), page 6-43
- [getDisplayHeight](#), page 6-45

getDisplayHeight

HRESULT getDisplayHeight (SHORT *height);

Purpose	Gets the height in pixels at which the source should be displayed.		
Arguments	<table><tr><td><i>height</i></td><td>Height of the source stream, in pixels.</td></tr></table>	<i>height</i>	Height of the source stream, in pixels.
<i>height</i>	Height of the source stream, in pixels.		
Return Values	HRESULT S_OK/E_FAIL		
Exceptions	None.		
Events Fired	None.		
Notes	The actual source height may differ from the height that the source should be displayed.		
Examples	C# Example <pre>short displayH = axc.getDisplayHeight();</pre> JavaScript Example <pre>var displayH = axc.getDisplayHeight();</pre>		
Related Methods	getVideoWidth , page 6-42 getVideoHeight , page 6-43 getDisplayWidth , page 6-44		

getX

HRESULT getX (DOUBLE *x);

Purpose Retrieves the x coordinate of the center of the view rectangle.

Arguments

<i>x</i>	The x coordinate of the center of the view rectangle.
----------	---

Return Values HRESULT S_OK/E_FAIL

Exceptions None.

Events Fired None.

Notes None.

Examples

C# Example

```
double xCoord = axc.getX();
```

JavaScript Example

```
var xCoord = axc.getX();
```

Related Methods [getY](#), page 6-47

getY

HRESULT getY (DOUBLE *y);

Purpose	Retrieves the y coordinate of the center of the view rectangle.
----------------	---

Arguments	<table><tr><td>y</td><td>The y coordinate of the center of the view rectangle.</td></tr></table>	y	The y coordinate of the center of the view rectangle.
y	The y coordinate of the center of the view rectangle.		

Return Values	HRESULT S_OK/E_FAIL
----------------------	---------------------

Exceptions	None.
-------------------	-------

Events Fired	None.
---------------------	-------

Notes	None.
--------------	-------

Examples	C# Example <pre>double yCoord = axc.getY();</pre> JavaScript Example <pre>var yCoord = axc.getY();</pre>
-----------------	--

Related Methods	getX , page 6-46
------------------------	--

Methods for Creating Clips and Snapshots

The following sections describe the methods that provide functionality for creating clips and snapshots from archived video.

Table 6-3 *API Method Summary*

Method Name	Description
saveInPortableFormat , page 6-49	Saves a clip in WMV format. A profile window pops up once this API has been called, and the user must select the desired WMV format for the saved file.
createCiscoVideoArchive , page 6-51	Creates a clip of the video archive file with the .cva filename extension.
snapshot , page 6-53	Saves the current frame of video to the viewing client computer. Opens a Windows dialog box in which users can choose to save in a number of image formats, including BMP, GIF, JPEG, PNG, and TIFF.
getSnapshotDIB , page 6-54	Creates a snapshot of the current frame in standard Windows device-independent bitmap (DIB) format that can be saved to a .bmp file.
getSnapshotWin32DIB , page 6-55	Creates a snapshot of the current frame in standard Windows device-independent bitmap (DIB) format that can be saved to a .bmp file.

saveInPortableFormat

```
HRESULT saveInPortableFormat (
    BSTR source,
    BSTR startTime,
    BSTR stopTime,
    BSTR destination,
    BSTR profile);
```

Purpose

Saves a clip in WMV format. A profile window pops up once this API has been called, and the user must select the desired WMV format for the saved file.

Arguments

<i>source</i>	VSMS source URL in the form of <i>//host/source</i> .
<i>startTime</i>	Time to start saving, in UTC milliseconds.
<i>stopTime</i>	Time to end saving, in UTC milliseconds.
<i>destination</i>	Location where the clip is saved on the local client. An empty string causes the system to prompt for a destination.
<i>profile</i>	Must be blank.

Return Values

HRESULT S_OK/E_FAIL

Exceptions

None.

Events Fired

None.

Notes

This API requires user intervention during the execution of the request. For a headless API to perform the clip save, use the **save** method instead.

Examples

C# Example

```
long startTime = (long)Math.Round(axc.getUTCStartTime());
long endingTime = (long)Math.Round(axc.getUTCStopTime());

//convert the getUTCdate seconds to the milliseconds required by this call
long startTimeMS = startTime * 1000;
long clipEndTimeMS = endingTime * 1000;

try {
    axc.saveInPortableFormat(
        "bwims://myServer/myArchive",
        startTimeMS.ToString(),
        clipEndTimeMS.ToString(),
        "c:\\temp.wmv",
```

```

        ""
    );
} catch (Exception ex) {
    threadSafeSet2Text(ex.Message);
}

```

JavaScript Example

```

/*
 * Save Clip (wmv)
 * @param sourceURL  bwims:// url
 * @param startUTC   start time in UTC milliseconds
 * @param stopUTC    end time in UTC milliseconds
 * @param filePath   local client path to save the generated clip. null value will to
prompt user
 */
clipPortable : function(sourceURL, startUTC, stopUTC, filePath) {
    try {
        if (!this.axc.mtStreamStarting()) {
            sourceURL = new String(sourceURL);
            startUTC   = new String(startUTC);
            stopUTC    = new String(stopUTC);
            filePath   = (null == filePath) ? '' : new String(filePath);
            this.axc.saveInPortableFormat(sourceURL, startUTC, stopUTC, filePath, '');
        } else {
            alerts(TEXT['js-stream-not-loaded']);
        }
    } catch(ex) { this.setError(ex) }
}

```

Related Methods

[createCiscoVideoArchive](#), page 6-51

[setOnSaveResponse](#), page 6-81

createCiscoVideoArchive

HRESULT createCiscoVideoArchive (BSTR *xmlStreamInfo*);

Purpose Creates a clip of the video archive file with the .cva filename extension.

Arguments	<i>xmlStreamInfo</i> The parameters are passed as a string using the following XML schema: <pre> <?xml version="1.0" encoding="UTF-8"?> <createCiscoVideoArchive> <layoutName>YourLayoutName</layoutName> <clipTime> <begin>Start UTC Miliseconds</begin> <end>End UTC Miliseconds</end> </clipTime> <grid> <row> <cell> <cameraName>ARCHIVE_NAME</cameraName> <uri>bwims://SERVER_NAME/ARCHIVE_NAME</uri> </cell> <cell> <cameraName>ARCHIVE_NAME</cameraName> <uri>bwims://SERVER_NAME/ARCHIVE_NAME</uri> </cell> </row> [Note: Additional rows and cells are allowed] </grid> </createCiscoVideoArchive> </pre>
------------------	---

Return Values HRESULT S_OK/E_FAIL

Exceptions None.

Events Fired None.

Notes The CVA format supports creating clips with multiple video sources, such as a view in VSOM. The CVA clips can be played using Cisco Review Player.

Examples

C# Example

```

axc.createCiscoVideoArchive("
<?xml version="1.0" encoding="UTF-8"?>
<createCiscoVideoArchive>
  <layoutName>1X1</layoutName>

```

```

    <clipTime>
      <begin>1271021709</begin>
      <end>1271022009</end>
    </clipTime>
  <grid>
    <row>
      <cell>
        <cameraName>My2600Archive</cameraName>
        <uri>bwims://MyServer/My2600Archive</uri>

      </cell>
      <cell>
        <cameraName>My4500Archive</cameraName>
        <uri>bwims://MyServer/My4500Archive</uri>
      </cell>
    </row>
  </grid>
</createCiscoVideoArchive>");

```

JavaScript Example

```

axc.createCiscoVideoArchive("
<?xml version='1.0' encoding='UTF-8'?>
<createCiscoVideoArchive>
  <layoutName>1X1</layoutName>
  <clipTime>
    <begin>1271021709</begin>
    <end>1271022009</end>
  </clipTime>
  <grid>
    <row>
      <cell>
        <cameraName>My2600Archive</cameraName>
        <uri>bwims://MyServer/My2600Archive</uri>

      </cell>
      <cell>
        <cameraName>My4500Archive</cameraName>
        <uri>bwims://MyServer/My4500Archive</uri>
      </cell>
    </row>
  </grid>
</createCiscoVideoArchive>");

```

Related Methods

[saveInPortableFormat](#), page 6-49

[setOnSaveResponse](#), page 6-81

snapshot

HRESULT snapshot (void);

Purpose	Saves the current frame of video to the viewing client computer. Opens a Windows dialog box in which users can choose to save in a number of image formats, including BMP, GIF, JPEG, PNG, and TIFF.
Arguments	This method has no arguments.
Return Values	HRESULT S_OK/E_FAIL
Exceptions	None.
Events Fired	None.
Notes	The file types that are supported by this call are BMP, GIF, JPEG, PNG, and TIFF.
Examples	C# Example You cannot programmatically save the file: the user must enter data in the pop-up window. JavaScript Example You cannot programmatically save the file: the user must enter data in the pop-up window.
Related Methods	getSnapshotDIB, page 6-54 getSnapshotWin32DIB, page 6-55

getSnapshotDIB

HRESULT getSnapshotDIB ();

Purpose Creates a snapshot of the current frame in standard Windows device-independent bitmap (DIB) format that can be saved to a .bmp file.

Arguments This method has no arguments.

Return Values HRESULT S_OK/E_FAIL

Exceptions None.

Events Fired None.

Notes Follow the example provided here to generate a proper BMP file.

The difference between **getSnapshotDIB** and **getSnapshotWin32DIB** is that **getSnapshotDIB** returns a properly formed bitmap, while **getSnapshotWin32DIB** returns a bit array including the bitmap info header, the raw bitmap data, and extra characters that must be parsed out before the payload from the API can be consumed.

Examples

C# Example

```
object bmpFromArchive = axc.getSnapshotDIB();

//convert the object to a byte[]
BinaryFormatter bf = new BinaryFormatter();
MemoryStream ms = new MemoryStream();
bf.Serialize(ms, bmpFromArchive);
byte[] data = ms.ToArray();
string clientSnapShotLocation =
    "c:\\Client2Snapshot" +
    DateTime.UtcNow.ToString("yyMMddHmss") +
    ".bmp";
FileStream fileStream = new FileStream(clientSnapShotLocation, FileMode.Create);
fileStream.Write(data, 0, data.Length);
fileStream.Close();
```

JavaScript Example

None.

Related Methods

[snapshot, page 6-53](#)

[getSnapshotWin32DIB, page 6-55](#)

getSnapshotWin32DIB

HRESULT getSnapshotWin32DIB (VARIANT *pDIB);

Purpose

Creates a snapshot of the current frame in standard Windows device-independent bitmap (DIB) format that can be saved to a .bmp file.



Caution

The first 13 bytes must be trimmed from the object that is returned in order to get a proper bmp.

Arguments

<i>pDIB</i>	Raw video stream information.
-------------	-------------------------------

Return Values

HRESULT S_OK/E_FAIL

Exceptions

None.

Events Fired

None.

Notes

Follow the example provided here to generate a proper BMP file.

The difference between **getSnapshotDIB** and **getSnapshotWin32DIB** is that **getSnapshotDIB** returns a properly formed bitmap, while **getSnapshotWin32DIB** returns a bit array including the bitmap info header, the raw bitmap data, and extra characters that must be parsed out before the payload from the API can be consumed.

Examples

C# Example

```
//get snapshot, return an object
object bmpFromArchive = axc.getSnapshotWin32DIB();

//convert the object to a byte[]
BinaryFormatter bf = new BinaryFormatter();
MemoryStream ms = new MemoryStream();
bf.Serialize(ms, bmpFromArchive);
byte[] data = ms.ToArray();

// The first 13 bytes need to be trimmed off when writing to file (or using in any other
manner).
// The next 14 bytes need to be formatted as a valid BITMAPFILEHEADER.
int j = 13; //this value may need to change based on VMR control output
int size = (data.Length - j);

//fill the 14 byte header
data[j] = 0x42;
```

```
data[j + 1] = 0x4d;
BitConverter.GetBytes(size).CopyTo(data, j + 2);
data[j + 6] = 0;
data[j + 7] = 0;
data[j + 8] = 0;
data[j + 9] = 0;
data[j + 10] = 0x36;
data[j + 11] = 0;
data[j + 12] = 0;
data[j + 13] = 0;

// Now the byte array is correct, from byte 14 forward
string clientSnapshotLocation = "c:\\Client1Snapshot" +
    DateTime.UtcNow.ToString("yyMMddHmss") +
    ".bmp";
FileStream fileStream = new FileStream(clientSnapshotLocation, FileMode.Create);
fileStream.Write(data, j, data.Length - j);
fileStream.Close();
```

JavaScript Example

None.

Related Methods

[snapshot](#), page 6-53

[getSnapshotDIB](#), page 6-54

Methods for Controlling VMR Display

The following sections describe the methods that provide functionality for controlling a Video Mixing Renderer (VMR) display.

Table 6-4 *API Method Summary*

Method Name	Description
setAlpha , page 6-58	Sets the transparency level of the VMR filter.
getAlpha , page 6-59	Retrieves the transparency level of the VMR.
setTransparent , page 6-60	Sets the transparent color and updates the alpha bitmap.
getTransparent , page 6-61	Retrieves the transparent color in the stream.
setBaseRectColor , page 6-62	Sets the base rectangle color and updates the alpha bitmap of the loaded video.
getBaseRectColor , page 6-63	Retrieves the base rectangle color of the loaded stream.
setZoomRectColor , page 6-64	Sets the zoom rectangle color and updates the alpha bitmap of the loaded stream.
getZoomRectColor , page 6-65	Retrieves the zoom rectangle color.
setTimeStampRect , page 6-66	Sets the time stamp rectangle and updates the alpha bitmap.
setVmrDisplayMode , page 6-67	Sets the VMR display mode.
getVmrDisplayMode , page 6-68	Gets the VMR display mode.
setZoomFactor , page 6-69	Sets the zoom factor and updates the display.
getZoomFactor , page 6-70	Retrieves the zoom factor.
move , page 6-71	Changes the size of the viewing rectangle and updates the display.
deltaMove , page 6-72	Changes the view (zoom) rectangle by an increment and updates the display.
resizeVideoWindow , page 6-73	Changes both the rectangle size (zoom) and the video window X and Y co-ordinates.

setAlpha

HRESULT setAlpha (DOUBLE *alpha*);

Purpose

Sets the transparency level of the VMR filter.

Arguments

<i>alpha</i>	Alpha value:
	0—Not transparent 1—Transparent

Return Values

HRESULT S_OK/E_FAIL

Exceptions

None.

Events Fired

None.

Notes

If the VMR is not turned on via the AxClient *bag* property, this call will have no effect.

Examples

C# Example

```
axc.setAlpha(1)
```

JavaScript Example

```
axc.setAlpha(1)
```

Related Methods

[getAlpha](#), page 6-59

getAlpha

HRESULT getAlpha (DOUBLE *alpha);

Purpose Retrieves the transparency level of the VMR.

Arguments None.

Return Values HRESULT S_OK/E_FAIL

<i>alpha</i>	Alpha value:
	• 0—Not transparent • 1—Transparent

Exceptions None.

Events Fired None.

Notes If the VMR is not turned on via the AxClient *bag* property, this call will have no effect.

Examples

C# Example

```
double alpha = axc.getAlpha()
```

JavaScript Example

```
var alpha = axc.getAlpha()
```

Related Methods [setAlpha, page 6-58](#)

setTransparent

HRESULT setTransparent (LONG *rgb*);

Purpose	Sets the transparent color and updates the alpha bitmap.
Arguments	<i>rgb</i> Microsoft Access color code number that represents the transparent color.
Return Values	HRESULT S_OK/E_FAIL
Exceptions	None.
Events Fired	None.
Notes	If the VMR is not turned on via the AxClient <i>bag</i> property, this call will have no effect.
Examples	C# Example <pre>axc.setTransparent(8034025)</pre> JavaScript Example <pre>axc.setTransparent(8034025)</pre>
Related Methods	getTransparent, page 6-61

getTransparent

HRESULT getTransparent (LONG *rgb);

Purpose	Retrieves the transparent color in the stream.		
Arguments	<table><tr><td><i>rgb</i></td><td>Microsoft Access color code number that represents the transparent color.</td></tr></table>	<i>rgb</i>	Microsoft Access color code number that represents the transparent color.
<i>rgb</i>	Microsoft Access color code number that represents the transparent color.		
Return Values	HRESULT S_OK/E_FAIL		
Exceptions	None.		
Events Fired	None.		
Notes	If the VMR is not turned on via the AxClient <i>bag</i> property, this call will have no effect.		
Examples	C# Example <pre>long rgb = axc.getTransparent()</pre> JavaScript Example <pre>var rgb = axc.getTransparent()</pre>		
Related Methods	setTransparent , page 6-60		

setBaseRectColor

HRESULT setBaseRectColor (LONG *rgb*);

Purpose	Sets the base rectangle color and updates the alpha bitmap of the loaded video.
Arguments	<i>rgb</i> Microsoft Access color code number that represents the base rectangle color.
Return Values	HRESULT S_OK/E_FAIL
Exceptions	None.
Events Fired	None.
Notes	If the VMR is not turned on via the AxClient <i>bag</i> property, this call will have no effect.
Examples	C# Example <pre>axc.setBaseRectColor(13474304)</pre> JavaScript Example <pre>axc.setBaseRectColor(13474304)</pre>
Related Methods	getBaseRectColor , page 6-63

getBaseRectColor

HRESULT getBaseRectColor (LONG *rgb);

Purpose	Retrieves the base rectangle color of the loaded stream.		
Arguments	<table><tr><td><i>rgb</i></td><td>Microsoft Access color code number that represents the base rectangle color.</td></tr></table>	<i>rgb</i>	Microsoft Access color code number that represents the base rectangle color.
<i>rgb</i>	Microsoft Access color code number that represents the base rectangle color.		
Return Values	HRESULT S_OK/E_FAIL		
Exceptions	None.		
Events Fired	None.		
Notes	If the VMR is not turned on via the AxClient <i>bag</i> property, this call will have no effect.		
Examples	C# Example <pre>long rgb = axc.getBaseRectColor()</pre> JavaScript Example <pre>var rgb = axc.getBaseRectColor()</pre>		
Related Methods	setBaseRectColor , page 6-62		

setZoomRectColor

HRESULT setZoomRectColor (LONG *rgb*);

Purpose	Sets the zoom rectangle color and updates the alpha bitmap of the loaded stream.	
Arguments	<i>rgb</i>	Microsoft Access color code number that represents the zoom rectangle color.
Return Values	HRESULT S_OK/E_FAIL	
Exceptions	None.	
Events Fired	None.	
Notes	If the VMR is not turned on via the AxClient <i>bag</i> property, this call will have no effect.	
Examples	C# Example <pre>axc.setZoomRectColor(15643136)</pre> JavaScript Example <pre>axc.setZoomRectColor(15643136)</pre>	
Related Methods	getZoomRectColor , page 6-65	

getZoomRectColor

HRESULT getZoomRectColor (LONG *rgb*);

Purpose Retrieves the zoom rectangle color.

Arguments

<i>rgb</i>	Microsoft Access color code number that represents the zoom rectangle color.
------------	--

Return Values HRESULT S_OK/E_FAIL

Exceptions None.

Events Fired None.

Notes If the VMR is not turned on via the AxClient *bag* property, this call will have no effect.

Examples

C# Example

```
long rgb = axc.getZoomRectColor()
```

JavaScript Example

```
var rgb = axc.getZoomRectColor()
```

Related Methods [setZoomRectColor](#), page 6-64

setTimeStampRect

```
HRESULT setTimeStampRect (
    LONG left,
    LONG top,
    LONG right,
    LONG bottom);
```

Purpose Sets the time stamp rectangle and updates the alpha bitmap.

Arguments	<i>left</i>	Left coordinate of the rectangle.
	<i>top</i>	Top coordinate of the rectangle.
	<i>right</i>	Right coordinate of the rectangle.
	<i>bottom</i>	Bottom coordinate of the rectangle.

Return Values HRESULT S_OK/E_FAIL

Exceptions None.

Events Fired None.

Notes If the VMR is not turned on via the AxClient *bag* property, this call will have no effect.

Examples

C# Example

```
axc.setTimeStampRect(1, 3, 5, 1)
```

JavaScript Example

```
axc.setTimeStampRect(1, 3, 5, 1)
```

Related Methods None.

setVmrDisplayMode

HRESULT setVmrDisplayMode (SHORT *displayMode*);

Purpose

Sets the VMR display mode.

This places the zoom reticule on screen or hides the zoom reticule (default). This does not enable VMR (the *bag* property associated with *imsclient.dll* includes this setting and enables VMR).

Arguments

displayMode

VMR display mode:

- 0—Do not show the zoom reticule.
- 1—Show the zoom reticule.

The default value is 0.

Return Values

HRESULT S_OK/E_FAIL

Exceptions

None.

Events Fired

None.

Notes

If the VMR is not turned on via the AxClient *bag* property, this call will have no effect.

Examples**C# Example**

```
axc.setVmrDisplayMode(1);
```

JavaScript Example

```
axc.setVmrDisplayMode(1);
```

Related Methods

[getVmrDisplayMode](#), page 6-68

getVmrDisplayMode

HRESULT getVmrDisplayMode (SHORT *displayMode);

Purpose

Gets the VMR display mode.

This indicates that either the zoom reticule is on screen or the zoom reticule is hidden (default). This does not enable VMR (the *bag* property associated with imscient.dll includes this setting and enables VMR).

Arguments

displayMode

VMR display mode:

- 0—Does not show the zoom reticule.
- 1—Shows the zoom reticule.

The default value is 0.

Return Values

HRESULT S_OK/E_FAIL

Exceptions

None.

Events Fired

None.

Notes

If the VMR is not turned on via the AxClient *bag* property, this call will have no effect.

Examples

C# Example

```
short displayMode = axc.getVmrDisplayMode();
```

JavaScript Example

```
var displayMode = axc.getVmrDisplayMode();
```

Related Methods

[setVmrDisplayMode](#), page 6-67

setZoomFactor

HRESULT setZoomFactor (DOUBLE *val*);

Purpose	Sets the zoom factor and updates the display.
----------------	---

Arguments	<table><tr><td><i>val</i></td><td>The zoom factor.</td></tr></table>	<i>val</i>	The zoom factor.
<i>val</i>	The zoom factor.		

Return Values	HRESULT S_OK/E_FAIL
----------------------	---------------------

Exceptions	None.
-------------------	-------

Events Fired	None.
---------------------	-------

Notes	If the VMR is not turned on via the AxClient <i>bag</i> property, this call will have no effect.
--------------	--

Examples	C# Example <pre>axc.setZoomFactor(2)</pre>
-----------------	--

	JavaScript Example <pre>axc.setZoomFactor(2)</pre>
--	--

Related Methods	getZoomFactor , page 6-70
------------------------	---

getZoomFactor

HRESULT getZoomFactor (DOUBLE *val);

Purpose Retrieves the zoom factor.

Arguments

<i>val</i>	The zoom factor.
------------	------------------

Return Values HRESULT S_OK/E_FAIL

Exceptions None.

Events Fired None.

Notes If the VMR is not turned on via the AxClient *bag* property, this call will have no effect.

Examples

C# Example

```
double rgb = axc.getZoomFactor()
```

JavaScript Example

```
var rgb = axc.getZoomFactor()
```

Related Methods [setZoomFactor](#), page 6-69

move

```
HRESULT move (  
    DOUBLE x,  
    DOUBLE y);
```

Purpose	Changes the size of the viewing rectangle and updates the display.
----------------	--

Arguments	<i>x</i>	X coordinate of the viewing rectangle
	<i>y</i>	Y coordinate of the viewing rectangle

Return Values	HRESULT S_OK/E_FAIL
----------------------	---------------------

Exceptions	None.
-------------------	-------

Events Fired	None.
---------------------	-------

Notes	This is the underlying function to which resizeVideoWindow forwards.
--------------	--

Examples	C# Example
	<pre>//To view video in a 640x480 rectangle axc.Move(640, 480);</pre>

Examples	JavaScript Example
	<pre>axc.Move(640,480);</pre>

Related Methods	deltaMove , page 6-72 resizeVideoWindow , page 6-73
------------------------	--

deltaMove

HRESULT deltaMove (
DOUBLE x,
DOUBLE y);

Purpose Changes the view (zoom) rectangle by an increment and updates the display.

Arguments	<i>x</i>	X coordinate of the viewing rectangle
	<i>y</i>	Y coordinate of the viewing rectangle

Return Values HRESULT S_OK/E_FAIL

Exceptions None.

Events Fired None.

Notes None.

Examples

C# Example

```
//To view video in a 640x480 rectangle
axc.deltaMove(640, 480);
```

JavaScript Example

```
axc.deltaMove(640,480);
```

Related Methods [move, page 6-71](#)
[resizeVideoWindow, page 6-73](#)

resizeVideoWindow

```
void resizeVideoWindow (  
    int owner  
    int x,  
    int y,  
    int w,  
    int h);
```

Purpose Changes both the rectangle size (zoom) and the video window X and Y co-ordinates.

Arguments	<i>owner</i>	The handle of the target display window
	<i>x</i>	New x coordinate of the video window.
	<i>y</i>	New y coordinate of the video window.
	<i>w</i>	New width of the video window.
	<i>h</i>	New height of the video window.

Return Values HRESULT S_OK/E_FAIL

Exceptions None.

Events Fired None.

Notes This method is similar to the [move](#) method, except that the [move](#) method does not actually move the window. It relocates the window to the coordinates provided by the X and Y input parameters. The correct handle of the target display window must be provided to the Owner input parameter or this call will not have a visible impact on the display.

Examples **C# Example**

```
//This will resize the window to a 640x480 rectangle  
//in the upper left-most corner of the screen  
this.axc.ResizeVideoWindow(  
    (int)this.axc.Handle,  
    1,  
    1,  
    640,  
    480  
);
```

JavaScript Example

```
axc.ResizeVideoWindow(  
    axc,  
    1,  
    1,  
    640,  
    480  
);
```

Related Methods[move, page 6-71](#)[deltaMove, page 6-72](#)

Methods for Setting up Callbacks

The following sections describe the methods that provide functionality for setting up callbacks.

Table 6-5 *API Method Summary*

Method Name	Description
setOnEndOfStream , page 6-76	Invokes the specified user-defined callback function when an archive stops playing.
setOnMtStartStreamDone , page 6-77	Invokes the specified user-defined callback function when an archive playback is initiated.
setOnPlayrateChanged , page 6-79	Invokes the specified user-defined callback function when an archive play rate changes.
setOnSaveResponse , page 6-81	Invokes the specified user-defined callback function when a previously initiated save clip has finished.
setOnSeekTimeChanged , page 6-83	Invokes the specified user-defined callback function when an archive seek time changes.
setOnStartOfStream , page 6-84	Invokes the specified user-defined callback function when an archive playback is initiated.
setOnStartTimeChanged , page 6-85	Invokes the specified user-defined callback function when an archive start time changes.
setOnStateChanged , page 6-87	Invokes the specified user-defined callback function when an archive playback state changes.
setOnStopTimeChanged , page 6-89	Invokes the specified user-defined callback function when an archive stop time changes.

setOnEndOfStream

```
void setOnEndOfStream (String callbackFunction);
```

Purpose Invokes the specified user-defined callback function when an archive stops playing.

Arguments

<i>callbackFunction</i>	Name of a user-defined callback function that is invoked when an archive stops playing. user-defined callback function signature: void callbackFunction (string <i>name</i>); Arguments: <i>name</i>—AxCliet object ID.
-------------------------	--

Return Values None.

Exceptions None.

Events Fired None.

Notes None.

Examples

C# Example

```
axc.setOnEndOfStream('callback_SetOnEndOfStream');
function callback_SetOnEndOfStream(name){}
```

JavaScript Example

```
axc.setOnEndOfStream('callback_SetOnEndOfStream');
function callback_SetOnEndOfStream(name){}
```

Related Methods

[mtStartStream](#), page 6-3

[playForward](#), page 6-8

setOnMtStartStreamDone

void setOnMtStartStreamDone (String *callbackFunction*);

Purpose Invokes the specified user-defined callback function when an archive playback is initiated.

Arguments

<i>callbackFunction</i>	Name of a user-defined callback function that is invoked when an archive playback is initiated. User-defined callback function signature: void callbackFunction (string <i>name</i>); Arguments: <i>name</i>—AxClient object ID.
-------------------------	--

Return Values None.

Exceptions None.

Events Fired None.

Notes None.

Examples **C# Example**

```
axc.setOnMtStartStreamDone('callback_SetOnMtStartStreamDone');  
function callback_SetOnMtStartStreamDone(name) {  
  
    //Check if the stream is loaded and ready for more commands  
    if (axc.mtStreamStarting())  
    {  
        axc.mtStartStreamWait(100);  
    }  
}
```

JavaScript Example

```
axc.setOnMtStartStreamDone('callback_SetOnMtStartStreamDone');  
function callback_SetOnMtStartStreamDone(name) {  
  
    //Check if the stream is loaded and ready for more commands  
    if (axc.mtStreamStarting())  
    {  
        axc.mtStartStreamWait(100);  
    }  
}
```

Related Methods[mtStartStream, page 6-3](#)[mtStreamStarting, page 6-5](#)[mtStartStreamWait, page 6-7](#)[setOnStartOfStream, page 6-84](#)

setOnPlayrateChanged

```
void setOnPlayrateChanged (String callbackFunction);
```

Purpose	Invokes the specified user-defined callback function when an archive play rate changes.
----------------	---

Arguments	<table><tr><td><i>callbackFunction</i></td><td>Name of a user-defined callback function that is invoked when an archive play rate changes. User-defined callback function signature: <pre>void callbackFunction (string name, short playRate);</pre> Arguments: • <i>name</i>—AxClient object ID. • <i>playRate</i>—The rate to which video playback was changed. Valid values are 0.05, 0.10, 0.25, 0.50, 0.75, 0.80, 1, 2, 4, 8, 16, 32, and 64: – A value of 1 is the normal play rate. – A value less than 1 is a slower play rate. – A value greater than 1 is a faster play rate. </td></tr></table>	<i>callbackFunction</i>	Name of a user-defined callback function that is invoked when an archive play rate changes. User-defined callback function signature: <pre>void callbackFunction (string name, short playRate);</pre> Arguments: • <i>name</i>—AxClient object ID. • <i>playRate</i>—The rate to which video playback was changed. Valid values are 0.05, 0.10, 0.25, 0.50, 0.75, 0.80, 1, 2, 4, 8, 16, 32, and 64: – A value of 1 is the normal play rate. – A value less than 1 is a slower play rate. – A value greater than 1 is a faster play rate.
<i>callbackFunction</i>	Name of a user-defined callback function that is invoked when an archive play rate changes. User-defined callback function signature: <pre>void callbackFunction (string name, short playRate);</pre> Arguments: • <i>name</i>—AxClient object ID. • <i>playRate</i>—The rate to which video playback was changed. Valid values are 0.05, 0.10, 0.25, 0.50, 0.75, 0.80, 1, 2, 4, 8, 16, 32, and 64: – A value of 1 is the normal play rate. – A value less than 1 is a slower play rate. – A value greater than 1 is a faster play rate.		

Return Values	None.
----------------------	-------

Exceptions	None.
-------------------	-------

Events Fired	None.
---------------------	-------

Notes	None.
--------------	-------

Examples	C# Example
-----------------	-------------------

```
axc.setOnPlayrateChanged('callback_SetOnPlayrateChanged');  
function callback_SetOnPlayrateChanged(name, playRate) {}
```

	JavaScript Example
--	---------------------------

```
axc.setOnPlayrateChanged('callback_SetOnPlayrateChanged');  
function callback_SetOnPlayrateChanged(name, playRate) {}
```

Related Methods[setPlayrateEx, page 6-17](#)[getPlayrateEx, page 6-37](#)

setOnSaveResponse

```
void setOnSaveResponse (String callbackFunction);
```

Purpose Invokes the specified user-defined callback function when a previously initiated save clip has finished.

Arguments

<i>callbackFunction</i>	Name of a user-defined callback function that is invoked when a previously initiated save clip has finished. User-defined callback function signature: <pre>void callbackFunction (string name, bool success, bool confirm, string location, string message);</pre> Arguments: • <i>name</i>—AxClient object ID. • <i>success</i>—Indicates whether the clip was saved successfully. • <i>confirm</i>—Indicates whether the clip confirmation was received. • <i>location</i>—Location where the clip is saved. Valid values can be one of the following keywords: – remote—Saves the clip to the remote VSMS host. – 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 local PC. • <i>message</i>—String. Representation of message returned by VSMS.
-------------------------	---

Return Values None.

Exceptions None.

Events Fired None.

Notes The server response is the result of the initial save methods.

Examples**C# Example**

```
axc.setOnSaveResponse('callback_SetOnSaveResponse');  
function callback_SetOnSaveResponse(name, success, confirm, location, message) {}
```

JavaScript Example

```
axc.setOnSaveResponse('callback_SetOnSaveResponse');  
function callback_SetOnSaveResponse(name, success, confirm, location, message) {}
```

Related Methods

[saveInPortableFormat](#), page 6-49

[createCiscoVideoArchive](#), page 6-51

setOnSeekTimeChanged

void setOnSeekTimeChanged (String *callbackFunction*);

Purpose Invokes the specified user-defined callback function when an archive seek time changes.

Arguments *callbackFunction* The name of the user-defined callback function that is invoked when an archive seek time changes.

User-defined callback function signature:

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

Arguments:

- *name*—AxClient object ID.
- *seekTime*—Date and time (in UTC format) of the archive after the performed seek operation.

Return Values None.

Exceptions None.

Events Fired None.

Notes The callback function is invoked when a seek method has completed.

Examples

C# Example

```
axc.setOnSeekTimeChanged('callback_SetOnSeekTimeChanged');  
function callback_SetOnSeekTimeChanged(name, seekTime) {}
```

JavaScript Example

```
axc.setOnSeekTimeChanged('callback_SetOnSeekTimeChanged');  
function callback_SetOnSeekTimeChanged(name, seekTime) {}
```

Related Methods [repeatUTCsegment](#), page 6-18
[seekToUTCTime](#), page 6-19
[seekToPercentage](#), page 6-20

setOnStartOfStream

void setOnStartOfStream (String *callbackFunction*);

Purpose Invokes the specified user-defined callback function when an archive playback is initiated.

Arguments

<i>callbackFunction</i>	Name of a user-defined callback function that is invoked when an archive playback is initiated.
	User-defined callback function signature:
	void callbackFunction (string <i>name</i>);
	Arguments:
	<i>name</i>—AxClient object ID.

Return Values None.

Exceptions None.

Events Fired None.

Notes None.

Examples

C# Example

```
axc.setOnStartOfStream('callback_SetOnStartOfStream');
function callback_SetOnStartOfStream(name) {}
```

JavaScript Example

```
axc.setOnStartOfStream('callback_SetOnStartOfStream');
function callback_SetOnStartOfStream(name) {}
```

Related Methods

[mtStartStream](#), page 6-3

[mtStreamStarting](#), page 6-5

[mtStartStreamWait](#), page 6-7

[setOnMtStartStreamDone](#), page 6-77

setOnStartTimeChanged

void setOnStartTimeChanged (String *callbackFunction*);

Purpose Invokes the specified user-defined callback function when an archive start time changes.

Arguments

<i>callbackFunction</i>	Name of the user-defined callback function that is invoked when an archive start time changes. User-defined callback function signature: void callbackFunction (string <i>name</i> , Date <i>startTime</i>); Arguments: • <i>name</i>—AxClient object ID. • <i>startTime</i>—New start time (in UTC format) of the archive.
-------------------------	---

Return Values None.

Exceptions None.

Events Fired None.

Notes This callback 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.

Examples

C# Example

```
axc.setOnStartTimeChanged('callback_SetOnStartTimeChanged');  
function callback_SetOnStartTimeChanged(name, startTime) {}
```

JavaScript Example

```
axc.setOnStartTimeChanged('callback_SetOnStartTimeChanged');  
function callback_SetOnStartTimeChanged(name, startTime) {}
```

Related Methods[repeatUTCsegment, page 6-18](#)[seekToUTCtime, page 6-19](#)[seekToPercentage, page 6-20](#)

setOnStateChanged

```
void setOnOnStateChanged (String callbackFunction);
```

Purpose	Invokes the specified user-defined callback function when an archive playback state changes.
----------------	--

Arguments	<table><tr><td><i>callbackFunction</i></td><td>Name of a user-defined callback function that is invoked when an archive playback state changes. User-defined callback function signature: <pre>void callbackFunction (string name, int state);</pre> Arguments: • <i>name</i>—AxClient object ID. • <i>state</i>—State of the archive playback. Valid values can be one of the following integers: – 0—Paused – 1—Playing Forward – 2—Playing Reverse – 3—Searching/Seeking – 4—Loading – 5—Stopped – 6—First Frame Received </td></tr></table>	<i>callbackFunction</i>	Name of a user-defined callback function that is invoked when an archive playback state changes. User-defined callback function signature: <pre>void callbackFunction (string name, int state);</pre> Arguments: • <i>name</i>—AxClient object ID. • <i>state</i>—State of the archive playback. Valid values can be one of the following integers: – 0—Paused – 1—Playing Forward – 2—Playing Reverse – 3—Searching/Seeking – 4—Loading – 5—Stopped – 6—First Frame Received
<i>callbackFunction</i>	Name of a user-defined callback function that is invoked when an archive playback state changes. User-defined callback function signature: <pre>void callbackFunction (string name, int state);</pre> Arguments: • <i>name</i>—AxClient object ID. • <i>state</i>—State of the archive playback. Valid values can be one of the following integers: – 0—Paused – 1—Playing Forward – 2—Playing Reverse – 3—Searching/Seeking – 4—Loading – 5—Stopped – 6—First Frame Received		

Return Values	None.
----------------------	-------

Exceptions	None.
-------------------	-------

Events Fired	None.
---------------------	-------

Notes	None.
--------------	-------

Examples	C# Example <pre>axc.setOnStateChanged('callback_SetOnStateChanged'); function callback_SetOnStateChanged(name, state) {}</pre>
-----------------	--

JavaScript Example

```
axc.setOnStateChanged('callback_SetOnStateChanged');  
function callback_SetOnStateChanged(name, state) {}
```

Related Methods[Methods for Controlling Video Operations, page 6-2](#)

setOnStopTimeChanged

void setOnStopTimeChanged (String *callbackFunction*);

Purpose Invokes the specified user-defined callback function when an archive stop time changes.

Arguments

<i>callbackFunction</i>	Name of a user-defined callback function that is invoked when an archive stop time changes. User-defined callback function signature: void callbackFunction (string <i>name</i>, Date <i>stopTime</i>); Arguments: • <i>name</i>—AxClient object ID. • <i>stopTime</i>—New stop time (in UTC format) of the archive.
-------------------------	--

Return Values None.

Exceptions None.

Events Fired None.

Notes None.

Examples **C# Example**

```
axc.setOnStopTimeChanged('callback_SetOnStopTimeChanged');  
function callback_SetOnStopTimeChanged(name, stopTime) {}
```

JavaScript Example

```
axc.setOnStopTimeChanged('callback_SetOnStopTimeChanged');  
function callback_SetOnStopTimeChanged(name, stopTime) {}
```

Related Methods

- [repeatUTCsegment, page 6-18](#)
- [seekToUTCTime, page 6-19](#)
- [seekToPercentage, page 6-20](#)



APPENDIX A

Turning on VMR with a C# Application

The following wrapper edits must be made in the class file that is created with AxImp if your C# application is to gain the benefit of the AxClient VMR mode.

AxImp is the Microsoft provided tool that turns a C++ DLL into an activeX DLL.

To gain this performance benefit, run AxImp with the /source parameter against imscient.dll (naming the output file aximsclient). Then edit the aximsclient.cs file that is created with the edits listed below. The edited class file must then be used to compile the activeX dll version of imscient.dll that will be used in your C# application.

The newly created activeX dll will playback video in VMR mode, which offers enormous performance gains by utilizing your video card memory and GPU. It is strongly recommended that all C# applications apply these edits to the wrapper class file in order to get the performance benefit associated with VMR.

Edits are shown below in normal font, while the original class syntax is in italics. Make sure to edit your own copy of the aximsclient.cs file by copying the new sections into it (do not copy the italic sections into your file).



Note

The following example is only a code snippet and is not intended to represent a complete compiled client code.

```
//-----  
// <auto-generated>  
//     This code was generated by a tool.  
//     Runtime Version:2.0.50727.3603  
//  
//     Changes to this file may cause incorrect behavior and will be lost if  
//     the code is regenerated.  
// </auto-generated>  
//-----  
[assembly: System.Reflection.AssemblyVersion("1.0.0.0")]  
[assembly: System.Windows.Forms.AxHost.TypeLibraryTimeStamp("7/20/2010 10:43:28 AM")]  
namespace Aximsclient {  
  
    //ADDED FOR PROPERTIES  
    using System;  
    using System.Reflection;  
    using System.Security.Permissions;  
    using System.Windows.Forms;  
    //END ADD  
  
    [System.Windows.Forms.AxHost.ClsidAttribute("{41293422-93fd-443c-b848-e07edbf866c3}")]  
    [System.ComponentModel.DesignTimeVisibleAttribute(true)]
```

```

[System.ComponentModel.DefaultEvent("OnStopTimeChanged")]
public class axc : System.Windows.Forms.AxHost {

    private imsclient.IMediaPlayerCtrl ocx;
    private axcEventMulticaster eventMulticaster;
    private System.Windows.Forms.AxHost.ConnectionPointCookie cookie;
    public axc() :
        base("41293422-93fd-443c-b848-e07edbf866c3") {

        //ADDED FOR PROPERTIES
        Type propertyBagStreamType;
        ConstructorInfo propertyBagStreamCtor;
        Object propertyBag;
        MethodInfo propertyBagStreamWriteMethod;
        ConstructorInfo stateCtor;
        Object state;
        private void enableVMRMode()
        {
            setOCXProperty("EnableVMRmode", "true");
        }
        private void setOCXProperty(String strName, String strValue)
        {
            try
            {
                ReflectionPermission perms = new
                ReflectionPermission(ReflectionPermissionFlag.MemberAccess);
                perms.Assert();

                this.propertyBagStreamType =
                this.GetType().BaseType.GetNestedType("PropertyBagStream", BindingFlags.NonPublic);
                if (this.propertyBagStreamType != null)
                {
                    this.propertyBagStreamCtor =
                    this.propertyBagStreamType.GetConstructor(BindingFlags.Instance | BindingFlags.Public,
                    null, Type.EmptyTypes, null);
                    if (this.propertyBagStreamCtor != null)
                    {
                        this.propertyBag = this.propertyBagStreamCtor.Invoke(null);
                        if (this.propertyBag != null)
                        {
                            this.propertyBagStreamWriteMethod =
                            this.propertyBagStreamType.GetMethod("System.Windows.Forms.UnsafeNativeMethods.IPropertyBag
                            g.Write", BindingFlags.FlattenHierarchy | BindingFlags.Instance | BindingFlags.NonPublic);
                            if (this.propertyBagStreamWriteMethod != null)
                            {
                                this.propertyBagStreamWriteMethod.Invoke(this.propertyBag,
                                new object[2] { strName, strValue });
                                stateCtor =
                                typeof(AxHost.State).GetConstructor(BindingFlags.Instance | BindingFlags.NonPublic, null,
                                new Type[1] { this.propertyBagStreamType }, null);
                                state = stateCtor.Invoke(new object[1] { this.propertyBag
                                });

                                System.Security.CodeAccessPermission.RevertAssert();
                                this.OcxState = this.state as State;
                            }
                        }
                    }
                }
            }
            catch (Exception ex)
            {
                System.Diagnostics.Trace.WriteLine("Error in axc: " + ex.Message);
                // Your stuff here
            }
        }
    }
}

```



```
        // For example set a flag somewhere that indicates error condition
    }
}
//END ADD
```




APPENDIX B

Supported Media Devices

This appendix lists the media devices supported by Cisco Video Surveillance Manager (VSM) and describes the model, media type, transport, format, and resolution for each supported device. The keywords listed in [Table B-1](#) are the *device* values that specify the proxy media source for the **src***type*=*device* name-value pair used in proxy commands. For more information, see [Chapter 3, “Proxy Commands.”](#) The Model IDs listed in [Table B-1](#) are the values returned with the Get Proxy Model command. For more information, see the [“Get Proxy Model” section on page 3-20.](#)



Note

This document contains the complete list of devices that were supported at the time it was published; however, the list of supported devices frequently changes. To see the list of devices that are supported on your system, use the following URL:

http://<VSMSServerHostnameOrIP>/doc/BMS/source_types.html

Table B-1 **Media Devices Supported by VSM**

Keyword	Model ID	Model Name	Media Type	Transport	Format	Resolutions
acti2120	0	ACTi 2120 Network Camera	mpeg4-v	tcp	NTSC	4cif (720 x 480), cif (352 x 240)
autodome	0	Autodome Analog Camera	N/A	N/A	N/A	N/A
axis_driver_v1	0	Axis PTZ generic camera (v1)	N/A	N/A	N/A	N/A
axis_driver_v2	0	Axis PTZ generic camera (v2)	N/A	N/A	N/A	N/A
axis205	0	AXIS 205 Network Camera	jpeg	tcp	NTSC	4cif (640 x 480), cif (320 x 240), qcif (160 x 120)
axis206m	0	AXIS 206M Megapixel Network Camera	jpeg	tcp	NTSC	3M (1280 x 1024), 2M (1280 x 960), 1M (1280 x 720), 4cif (640 x 480), cif (320 x 240)
axis206w	0	AXIS 206W Wireless Network Camera	jpeg	tcp	NTSC	4cif (640 x 480), 2cif (640 x 360), cif (320 x 240), qcif (160 x 120)

Table B-1 Media Devices Supported by VSM

Keyword	Model ID	Model Name	Media Type	Transport	Format	Resolutions
axis207	35	AXIS 207 Network Camera	jpeg	tcp	NTSC	4cif (640 x 480), 2cif (480 x 360), cif (320 x 240), qcif (160 x 120)
axis207	35	AXIS 207 Network Camera	mpeg4-v	tcp, udp, multicast	NTSC	4cif (640 x 480), 2cif (480 x 360), cif (320 x 240), qcif (160 x 120)
axis207mw	93	AXIS 207MW MegaPixel Wireless Network Camera	audio	tcp	N/A	N/A
axis207mw	93	AXIS 207MW MegaPixel Wireless Network Camera	mpeg4-v	tcp, udp, multicast	NTSC	1M (1280 x 720), 4cif (640 x 480), 2cif (480 x 360), cif (320 x 240), qcif (160 x 120)
axis207mw	93	AXIS 207MW MegaPixel Wireless Network Camera	jpeg	tcp	NTSC	1M (1280 x 720), 4cif (640 x 480), 2cif (480 x 360), cif (320 x 240), qcif (160 x 120)
axis209m	146	AXIS 209M MegaPixel Network Camera	jpeg	tcp	NTSC	1M (1280 x 720), 4cif (640 x 480), 2cif (480 x 360), cif (320 x 240), qcif (160 x 120)
axis209m	146	AXIS 209M MegaPixel Network Camera	mpeg4-v	tcp, udp, multicast	NTSC	1M (1280 x 720), 4cif (640 x 480), 2cif (480 x 360), cif (320 x 240), qcif (160 x 120)
axis210	20	AXIS 210 Network Camera	jpeg	tcp	NTSC	4cif (640 x 480), 2cif (480 x 360), cif (320 x 240), qcif (160 x 120)
axis210	20	AXIS 210 Network Camera	mpeg4-v	tcp, udp, multicast	NTSC	4cif (640 x 480), 2cif (480 x 360), cif (320 x 240), qcif (160 x 120)
axis2100	12	AXIS 2100 Network Camera	jpeg	tcp	NTSC	4cif (640 x 480), cif (320 x 240)
axis210a	36	AXIS 210A Network Camera	audio	tcp	N/A	N/A
axis210a	36	AXIS 210A Network Camera	mpeg4-v	tcp, udp, multicast	NTSC	4cif (640 x 480), 2cif (480 x 360), cif (320 x 240), qcif (160 x 120)
axis210a	36	AXIS 210A Network Camera	jpeg	tcp	NTSC	4cif (640 x 480), 2cif (480 x 360), cif (320 x 240), qcif (160 x 120)
axis211	21	AXIS 211 Network Camera	jpeg	tcp	NTSC	4cif (640 x 480), 2cif (480 x 360), cif (320 x 240), qcif (160 x 120)
axis211	21	AXIS 211 Network Camera	mpeg4-v	tcp, udp, multicast	NTSC	4cif (640 x 480), 2cif (480 x 360), cif (320 x 240), qcif (160 x 120)
axis211a	31	AXIS 211A Network Camera	audio	tcp	N/A	N/A
axis211a	31	AXIS 211A Network Camera	mpeg4-v	tcp, udp, multicast	NTSC	4cif (640 x 480), 2cif (480 x 360), cif (320 x 240), qcif (160 x 120)

Table B-1 **Media Devices Supported by VSM**

Keyword	Model ID	Model Name	Media Type	Transport	Format	Resolutions
axis211a	31	AXIS 211A Network Camera	jpeg	tcp	NTSC	4cif (640 x 480), 2cif (480 x 360), cif (320 x 240), qcif (160 x 120)
axis211m	145	AXIS 211M Network Camera	audio	tcp	N/A	N/A
axis211m	145	AXIS 211M Network Camera	mpeg4-v	tcp, udp, multicast	NTSC	1.2M (1280 x 1024), 1M (1280 x 720), 4cif (640 x 480), 2cif (480 x 360), cif (320 x 240), qcif (160 x 120)
axis211m	145	AXIS 211M Network Camera	jpeg	tcp	NTSC	1.2M (1280 x 1024), 1M (1280 x 720), 4cif (640 x 480), 2cif (480 x 360), cif (320 x 240), qcif (160 x 120)
axis212	40	AXIS 212 Network Camera	jpeg	tcp	NTSC	4cif (640 x 480), 2cif (480 x 360), cif (320 x 240), qcif (160 x 120)
axis212	40	AXIS 212 Network Camera	mpeg4-v	tcp, udp, multicast	NTSC	4cif (640 x 480), 2cif (480 x 360), cif (320 x 240)
axis2120	8	AXIS 2120 Network Camera	jpeg	tcp	NTSC	4cif (704 x 480), cif (352 x 240)
axis213	30	AXIS 213 PTZ Network Camera	audio	tcp	N/A	N/A
axis213	30	AXIS 213 PTZ Network Camera	mpeg4-v	tcp, udp, multicast	NTSC	4cif (704 x 480), 2cif (704 x 240), cif (352 x 240), qcif (176 x 120)
axis213	30	AXIS 213 PTZ Network Camera	mpeg4-v	tcp, udp, multicast	PAL	4cif (704 x 576), 2cif (704 x 288), cif (352 x 288), qcif (176 x 144)
axis213	30	AXIS 213 PTZ Network Camera	jpeg	tcp	NTSC	4cif (704 x 480), 2cif (704 x 240), cif (352 x 240), qcif (176 x 120)
axis213	30	AXIS 213 PTZ Network Camera	jpeg	tcp	PAL	4cif (704 x 576), 2cif (704 x 288), cif (352 x 288), qcif (176 x 144)
axis2130	10	AXIS 2130 PTZ Network Camera	jpeg	tcp	NTSC	4cif (704 x 480), cif (352 x 240), qcif (176 x 120)
axis214	41	AXIS 214 PTZ Network Camera	audio	tcp	N/A	N/A
axis214	41	AXIS 214 PTZ Network Camera	mpeg4-v	tcp, udp, multicast	NTSC	4cif (704 x 480), 2cif (704 x 240), cif (352 x 240), qcif (176 x 120)
axis214	41	AXIS 214 PTZ Network Camera	mpeg4-v	tcp, udp, multicast	PAL	4cif (704 x 576), 2cif (704 x 288), cif (352 x 288), qcif (176 x 144)
axis214	41	AXIS 214 PTZ Network Camera	jpeg	tcp	NTSC	4cif (704 x 480), 2cif (704 x 240), cif (352 x 240), qcif (176 x 120)
axis214	41	AXIS 214 PTZ Network Camera	jpeg	tcp	PAL	4cif (704 x 576), 2cif (704 x 288), cif (352 x 288), qcif (176 x 144)
axis215	119	AXIS 215 PTZ Network Camera	audio	tcp	N/A	N/A

Table B-1 Media Devices Supported by VSM

Keyword	Model ID	Model Name	Media Type	Transport	Format	Resolutions
axis215	119	AXIS 215 PTZ Network Camera	mpeg4-v	tcp, udp, multicast	NTSC	4cif (704 x 480), 2cif (704 x 240), cif (352 x 240), qcif (176 x 120)
axis215	119	AXIS 215 PTZ Network Camera	mpeg4-v	tcp, udp, multicast	PAL	4cif (704 x 576), 2cif (704 x 288), cif (352 x 288), qcif (176 x 144)
axis215	119	AXIS 215 PTZ Network Camera	jpeg	tcp	NTSC	4cif (704 x 480), 2cif (704 x 240), cif (352 x 240), qcif (176 x 120)
axis215	119	AXIS 215 PTZ Network Camera	jpeg	tcp	PAL	4cif (704 x 576), 2cif (704 x 288), cif (352 x 288), qcif (176 x 144)
axis216	39	AXIS 216FD Network Camera	jpeg	tcp	NTSC	4cif (640 x 480), 2cif (480 x 360), cif (320 x 240), qcif (160 x 120)
axis216	39	AXIS 216FD Network Camera	mpeg4-v	tcp, udp, multicast	NTSC	4cif (640 x 480), 2cif (480 x 360), cif (320 x 240), qcif (160 x 120)
axis216m	144	AXIS 216MFD Network Camera	jpeg	tcp	NTSC	1.2M (1280 x 1024), 1M (1280 x 720), 4cif (640 x 480), 2cif (480 x 360), cif (320 x 240), qcif (160 x 120)
axis216m	144	AXIS 216MFD Network Camera	mpeg4-v	tcp, udp, multicast	NTSC	1.2M (1280 x 1024), 1M (1280 x 720), 4cif (640 x 480), 2cif (480 x 360), cif (320 x 240), qcif (160 x 120)
axis221	165	AXIS 221 Day and Night Camera	jpeg	tcp	NTSC	4cif (640 x 480), 2cif (480 x 360), cif (320 x 240), qcif (160 x 120)
axis221	165	AXIS 221 Day and Night Camera	mpeg4-v	tcp, udp, multicast	NTSC	4cif (640 x 480), 2cif (480 x 360), cif (320 x 240), qcif (160 x 120)
axis223m	0	AXIS 223M Network Camera	audio	tcp	N/A	N/A
axis223m	0	AXIS 223M Network Camera	mpeg4-v	tcp, udp, multicast	NTSC	4cif (640 x 480), cif (352 x 240), qcif (176 x 120)
axis223m	0	AXIS 223M Network Camera	jpeg	tcp	NTSC	2M (1600 x 1200), 1M (1280 x 960), 4cif (704 x 480), 2cif (704 x 240), cif (352 x 240), qcif (176 x 120)
axis225	92	AXIS 225FD Network Camera	jpeg	tcp	NTSC	4cif (640 x 480), 2cif (480 x 360), cif (320 x 240), qcif (160 x 120)
axis225	92	AXIS 225FD Network Camera	mpeg4-v	tcp, udp, multicast	NTSC	4cif (640 x 480), 2cif (480 x 360), cif (320 x 240), qcif (160 x 120)
axis230	22	AXIS 230 MPEG-2 Network Camera	mpeg2-v	tcp	NTSC	4cif (704 x 480), cif (352 x 240), qcif (160 x 112)

Table B-1 **Media Devices Supported by VSM**

Keyword	Model ID	Model Name	Media Type	Transport	Format	Resolutions
axis231	53	AXIS 231D Network Dome Camera	jpeg	tcp	NTSC	4cif (704 x 480), 2cif (704 x 240), cif (352 x 240), qcif (176 x 120)
axis231	53	AXIS 231D Network Dome Camera	jpeg	tcp	PAL	4cif (704 x 576), 2cif (704 x 288), cif (352 x 288), qcif (176 x 144)
axis231	53	AXIS 231D Network Dome Camera	mpeg4-v	tcp, udp, multicast	NTSC	4cif (704 x 480), 2cif (704 x 240), cif (352 x 240), qcif (176 x 120)
axis231	53	AXIS 231D Network Dome Camera	mpeg4-v	tcp, udp, multicast	PAL	4cif (704 x 576), 2cif (704 x 288), cif (352 x 288), qcif (176 x 144)
axis232	29	AXIS 232D Network Dome Camera	jpeg	tcp	NTSC	4cif (704 x 480), 2cif (704 x 240), cif (352 x 240), qcif (176 x 120)
axis232	29	AXIS 232D Network Dome Camera	jpeg	tcp	PAL	4cif (704 x 576), 2cif (704 x 288), cif (352 x 288), qcif (176 x 144)
axis232	29	AXIS 232D Network Dome Camera	mpeg4-v	tcp, udp, multicast	NTSC	4cif (704 x 480), 2cif (704 x 240), cif (352 x 240), qcif (176 x 120)
axis232	29	AXIS 232D Network Dome Camera	mpeg4-v	tcp, udp, multicast	PAL	4cif (704 x 576), 2cif (704 x 288), cif (352 x 288), qcif (176 x 144)
axis233	111	AXIS 233D Network Dome Camera	jpeg	tcp	NTSC	4cif (704 x 480), 2cif (704 x 240), cif (352 x 240), qcif (176 x 120)
axis233	111	AXIS 233D Network Dome Camera	jpeg	tcp	PAL	4cif (704 x 576), 2cif (704 x 288), cif (352 x 288), qcif (176 x 144)
axis233	111	AXIS 233D Network Dome Camera	mpeg4-v	tcp, udp, multicast	NTSC	4cif (704 x 480), 2cif (704 x 240), cif (352 x 240), qcif (176 x 120)
axis233	111	AXIS 233D Network Dome Camera	mpeg4-v	tcp, udp, multicast	PAL	4cif (704 x 576), 2cif (704 x 288), cif (352 x 288), qcif (176 x 144)
axis2400	1	AXIS 2400 Video Server (v1)	jpeg	tcp	NTSC	4cif (704 x 480), cif (352 x 240), qcif (176 x 112)
axis2400l	2	AXIS 2400 Video Server (v2)	jpeg	tcp	NTSC	4cif (704 x 480), cif (352 x 240), qcif (176 x 112)
axis2400plus	16	AXIS 2400+ Video Server	jpeg	tcp	NTSC	4cif (704 x 480), cif (352 x 240), qcif (176 x 112)
axis2400plus	16	AXIS 2400+ Video Server	jpeg	tcp	PAL	4cif (704 x 576), cif (352 x 288), qcif (176 x 144)

Table B-1 Media Devices Supported by VSM

Keyword	Model ID	Model Name	Media Type	Transport	Format	Resolutions
axis2401	5	AXIS 2401 Video Server (v1)	jpeg	tcp	NTSC	4cif (704 x 480), cif (352 x 240), qcif (176 x 112)
axis2401	5	AXIS 2401 Video Server (v1)	jpeg	tcp	PAL	4cif (704 x 576), cif (352 x 288), qcif (176 x 144)
axis2401l	6	AXIS 2401L Video Server (v2)	jpeg	tcp	NTSC	4cif (704 x 480), cif (352 x 240), qcif (176 x 112)
axis2401l	6	AXIS 2401L Video Server (v2)	jpeg	tcp	PAL	4cif (704 x 576), cif (352 x 288), qcif (176 x 144)
axis2401plus	17	AXIS 2401+ Video Server	jpeg	tcp	NTSC	4cif (704 x 480), cif (352 x 240), qcif (176 x 112)
axis2401plus	17	AXIS 2401+ Video Server	jpeg	tcp	PAL	4cif (704 x 576), cif (352 x 288), qcif (176 x 144)
axis240q	37	AXIS 240Q Video Server	jpeg	tcp	NTSC	4cif (704 x 480), 2cif (704 x 240), cif (352 x 240), qcif (176 x 120)
axis240q	37	AXIS 240Q Video Server	jpeg	tcp	PAL	4cif (704 x 576), cif (352 x 288), qcif (176 x 144)
axis2411	0	AXIS 2411 Video Server	jpeg	tcp	NTSC	4cif (704 x 480), cif (352 x 240), qcif (176 x 112)
axis2411	0	AXIS 2411 Video Server	jpeg	tcp	PAL	4cif (704 x 576), cif (352 x 288), qcif (176 x 144)
axis241q	19	AXIS 241Q Video Server	jpeg	tcp	NTSC	4cif (704 x 480), 2cif (704 x 240), cif (352 x 240), qcif (176 x 120)
axis241q	19	AXIS 241Q Video Server	jpeg	tcp	PAL	4cif (704 x 576), 2cif (704 x 288), cif (352 x 288), qcif (176 x 144)
axis241q	19	AXIS 241Q Video Server	mpeg4-v	tcp, udp, multicast	NTSC	4cif (704 x 480), 2cif (704 x 240), cif (352 x 240), qcif (176 x 112)
axis241q	19	AXIS 241Q Video Server	mpeg4-v	tcp, udp, multicast	PAL	4cif (704 x 576), 2cif (704 x 288), cif (352 x 288), qcif (176 x 144)
axis241qa	33	AXIS 241QA Video Server	audio	tcp	N/A	N/A
axis241qa	33	AXIS 241QA Video Server	mpeg4-v	tcp, udp, multicast	NTSC	4cif (704 x 480), 2cif (704 x 240), cif (352 x 240), qcif (176 x 112)
axis241qa	33	AXIS 241QA Video Server	mpeg4-v	tcp, udp, multicast	PAL	4cif (704 x 576), 2cif (704 x 288), cif (352 x 288), qcif (176 x 144)
axis241qa	33	AXIS 241QA Video Server	jpeg	tcp	NTSC	4cif (704 x 480), 2cif (704 x 240), cif (352 x 240), qcif (176 x 112)
axis241qa	33	AXIS 241QA Video Server	jpeg	tcp	PAL	4cif (704 x 576), 2cif (704 x 288), cif (352 x 288), qcif (176 x 144)
axis241s	18	AXIS 241S Video Server	jpeg	tcp	NTSC	4cif (704 x 480), 2cif (704 x 240), cif (352 x 240), qcif (176 x 120)
axis241s	18	AXIS 241S Video Server	jpeg	tcp	PAL	4cif (704 x 576), 2cif (704 x 288), cif (352 x 288), qcif (176 x 144)

Table B-1 **Media Devices Supported by VSM**

Keyword	Model ID	Model Name	Media Type	Transport	Format	Resolutions
axis241s	18	AXIS 241S Video Server	mpeg4-v	tcp, udp, multicast	NTSC	4cif (704 x 480), 2cif (704 x 240), cif (352 x 240), qcif (176 x 112)
axis241s	18	AXIS 241S Video Server	mpeg4-v	tcp, udp, multicast	PAL	4cif (704 x 576), 2cif (704 x 288), cif (352 x 288), qcif (176 x 144)
axis241sa	34	AXIS 241SA Video Server	audio	tcp	N/A	N/A
axis241sa	34	AXIS 241SA Video Server	mpeg4-v	tcp, udp, multicast	NTSC	4cif (704 x 480), 2cif (704 x 240), cif (352 x 240), qcif (176 x 112)
axis241sa	34	AXIS 241SA Video Server	mpeg4-v	tcp, udp, multicast	PAL	4cif (704 x 576), 2cif (704 x 288), cif (352 x 288), qcif (176 x 144)
axis241sa	34	AXIS 241SA Video Server	jpeg	tcp	NTSC	4cif (704 x 480), 2cif (704 x 240), cif (352 x 240), qcif (176 x 120)
axis241sa	34	AXIS 241SA Video Server	jpeg	tcp	PAL	4cif (704 x 576), 2cif (704 x 288), cif (352 x 288), qcif (176 x 144)
axis2420	9	AXIS 2420 Network Camera	jpeg	tcp	NTSC	4cif (704 x 480), cif (352 x 240)
axis243q	95	AXIS 243Q Video Server	jpeg	tcp	NTSC	4cif (704 x 480), 2cif (704 x 240), cif (352 x 240), qcif (176 x 120)
axis243q	95	AXIS 243Q Video Server	mpeg4-v	tcp, udp, multicast	NTSC	4cif (704 x 480), 2cif (704 x 240), cif (352 x 240), qcif (176 x 120)
axis243sa	94	AXIS 243SA Video Server	audio	tcp	N/A	N/A
axis243sa	94	AXIS 243SA Video Server	mpeg4-v	tcp, udp, multicast	NTSC	4cif (704 x 480), 2cif (704 x 240), cif (352 x 240), qcif (176 x 120)
axis243sa	94	AXIS 243SA Video Server	mpeg4-v	tcp, udp, multicast	PAL	4cif (704 x 576), 2cif (704 x 288), cif (352 x 288), qcif (176 x 144)
axis243sa	94	AXIS 243SA Video Server	jpeg	tcp	NTSC	4cif (704 x 480), 2cif (704 x 240), cif (352 x 240), qcif (176 x 120)
axis243sa	94	AXIS 243SA Video Server	jpeg	tcp	PAL	4cif (704 x 576), 2cif (704 x 288), cif (352 x 288), qcif (176 x 144)
axis247s	147	AXIS 247S Video Server	audio	tcp	N/A	N/A
axis247s	147	AXIS 247S Video Server	mpeg4-v	tcp, udp, multicast	NTSC	4cif (704 x 480), 2cif (704 x 240), cif (352 x 240), qcif (176 x 120)
axis247s	147	AXIS 247S Video Server	mpeg4-v	tcp, udp, multicast	PAL	4cif (704 x 576), 2cif (704 x 288), cif (352 x 288), qcif (176 x 144)
axis247s	147	AXIS 247S Video Server	jpeg	tcp	NTSC	4cif (704 x 480), 2cif (704 x 240), cif (352 x 240), qcif (176 x 120)
axis247s	147	AXIS 247S Video Server	jpeg	tcp	PAL	4cif (704 x 576), 2cif (704 x 288), cif (352 x 288), qcif (176 x 144)

Table B-1 Media Devices Supported by VSM

Keyword	Model ID	Model Name	Media Type	Transport	Format	Resolutions
axis250s	11	AXIS 250S MPEG-2 Video Server	mpeg2-v	tcp	NTSC	4cif (704 x 480), 2cif (704 x 240), cif (352 x 240), qcif (160 x 112)
axis250s	11	AXIS 250S MPEG-2 Video Server	mpeg2-v	tcp	PAL	4cif (704 x 576), cif (352 x 288), qcif (160 x 144)
axisp3301	149	AXIS P3301 Network Camera	audio	tcp	N/A	N/A
axisp3301	149	AXIS P3301 Network Camera	jpeg	tcp	NTSC	4cif (640 x 480), 2cif (480 x 360), cif (320 x 240), qcif (240 x 180)
axisp3301	149	AXIS P3301 Network Camera	h264-v	udp, multicast	NTSC	4cif (640 x 480), 2cif (480 x 360), cif (320 x 240), qcif (240 x 180)
axisq7401	135	AXIS Q7401/Q7406 Video Server	audio	tcp	N/A	N/A
axisq7401	135	AXIS Q7401/Q7406 Video Server	jpeg	tcp	NTSC	4cif (704 x 480), 2cif (704 x 240), cif (352 x 240), qcif (176 x 120)
axisq7401	135	AXIS Q7401/Q7406 Video Server	jpeg	tcp	PAL	4cif (704 x 576), 2cif (704 x 288), cif (352 x 288), qcif (176 x 144)
axisq7401	135	AXIS Q7401/Q7406 Video Server	h264-v	udp	NTSC	4cif (704 x 480), 2cif (704 x 240), cif (352 x 240), qcif (176 x 120)
axisq7401	135	AXIS Q7401/Q7406 Video Server	h264-v	udp	PAL	4cif (704 x 576), 2cif (704 x 288), cif (352 x 288), qcif (176 x 144)
bosch_autodome_300	0	Bosch AutoDome 300 Network Camera	jpeg	tcp	NTSC	4cif (704 x 480), 2cif (704 x 240), cif (352 x 240), qcif (176 x 112)
bosch_autodome_300	0	Bosch AutoDome 300 Network Camera	jpeg	tcp	PAL	4cif (704 x 576), 2cif (704 x 288), cif (352 x 288), qcif (176 x 144)
bosch_autodome_300	0	Bosch AutoDome 300 Network Camera	mpeg4-v	tcp, udp	NTSC	4cif (704 x 480), 2cif (704 x 240), cif (352 x 240), qcif (176 x 112)
bosch_autodome_300	0	Bosch AutoDome 300 Network Camera	mpeg4-v	tcp, udp	PAL	4cif (704 x 576), 2cif (704 x 288), cif (352 x 288), qcif (176 x 144)
bosch_dnip	0	Bosch Dinion-IP-NWC- 0495 Network Camera	mpeg4-v	tcp, udp	NTSC	4cif (704 x 480), cif (352 x 240), qcif (176 x 112)

Table B-1 **Media Devices Supported by VSM**

Keyword	Model ID	Model Name	Media Type	Transport	Format	Resolutions
bosch_dnip	0	Bosch Dinion-IP-NWC- 0495 Network Camera	mpeg4-v	tcp, udp	PAL	4cif (704 x 576), cif (352 x 288), qcif (176 x 144)
cisco_242x	172	Cisco SD IP Camera Indoor Dome 2420 Series	jpeg	tcp, udp	NTSC	d1 (720 x 480), 4cif (704 x 480), cif (352 x 240)
cisco_242x	172	Cisco SD IP Camera Indoor Dome 2420 Series	jpeg	tcp, udp	PAL	d1 (720 x 576), 4cif (704 x 576), cif (352 x 288)
cisco_242x	172	Cisco SD IP Camera Indoor Dome 2420 Series	mpeg4-v	tcp, udp	NTSC	d1 (720 x 480), 4cif (704 x 480), cif (352 x 240)
cisco_242x	172	Cisco SD IP Camera Indoor Dome 2420 Series	mpeg4-v	tcp, udp	PAL	d1 (720 x 576), 4cif (704 x 576), cif (352 x 288)
cisco_252xV	173	Cisco SD IP Camera Indoor Vandal Dome 2520 Series	jpeg	tcp, udp	NTSC	d1 (720 x 480), 4cif (704 x 480), cif (352 x 240)
cisco_252xV	173	Cisco SD IP Camera Indoor Vandal Dome 2520 Series	jpeg	tcp, udp	PAL	d1 (720 x 576), 4cif (704 x 576), cif (352 x 288)
cisco_252xV	173	Cisco SD IP Camera Indoor Vandal Dome 2520 Series	mpeg4-v	tcp, udp	NTSC	d1 (720 x 480), 4cif (704 x 480), cif (352 x 240)
cisco_252xV	173	Cisco SD IP Camera Indoor Vandal Dome 2520 Series	mpeg4-v	tcp, udp	PAL	d1 (720 x 576), 4cif (704 x 576), cif (352 x 288)
cisco_253xV	174	Cisco SD IP Camera Ruggedized Dome 2530 Series	jpeg	tcp, udp	NTSC	d1 (720 x 480), 4cif (704 x 480), cif (352 x 240)
cisco_253xV	174	Cisco SD IP Camera Ruggedized Dome 2530 Series	jpeg	tcp, udp	PAL	d1 (720 x 576), 4cif (704 x 576), cif (352 x 288)

Table B-1 **Media Devices Supported by VSM**

Keyword	Model ID	Model Name	Media Type	Transport	Format	Resolutions
cisco_253xV	174	Cisco SD IP Camera Ruggedized Dome 2530 Series	mpeg4-v	tcp, udp	NTSC	d1 (720 x 480), 4cif (704 x 480), cif (352 x 240)
cisco_253xV	174	Cisco SD IP Camera Ruggedized Dome 2530 Series	mpeg4-v	tcp, udp	PAL	d1 (720 x 576), 4cif (704 x 576), cif (352 x 288)
cisco_4300	151	Cisco HD IP Camera 4300 Series	h264-v	udp, multicast	NTSC	1080p (1920 x 1080), 720p (1280 x 720), d1 (720 x 480), 4cif (704 x 480), cif (352 x 240)
cisco_4300	151	Cisco HD IP Camera 4300 Series	h264-v	udp, multicast	PAL	1080p (1920 x 1080), 720p (1280 x 720), d1 (720 x 576), 4cif (704 x 576), cif (352 x 288)
cisco_4300	151	Cisco HD IP Camera 4300 Series	jpeg	udp, multicast	NTSC	d1 (720 x 480), 4cif (704 x 480), cif (352 x 240)
cisco_4300	151	Cisco HD IP Camera 4300 Series	jpeg	udp, multicast	PAL	d1 (720 x 576), 4cif (704 x 576), cif (352 x 288)
cisco_4500	168	Cisco HD IP Camera 4500 Series	h264-v	udp, multicast	NTSC	1080p (1920 x 1080), 720p (1280 x 720), d1 (720 x 480), 4cif (704 x 480), cif (352 x 240)
cisco_4500	168	Cisco HD IP Camera 4500 Series	h264-v	udp, multicast	PAL	1080p (1920 x 1080), 720p (1280 x 720), d1 (720 x 576), 4cif (704 x 576), cif (352 x 288)
cisco_4500	168	Cisco HD IP Camera 4500 Series	jpeg	udp, multicast	NTSC	d1 (720 x 480), 4cif (704 x 480), cif (352 x 240)
cisco_4500	168	Cisco HD IP Camera 4500 Series	jpeg	udp, multicast	PAL	d1 (720 x 576), 4cif (704 x 576), cif (352 x 288)
cisco_avg	195	Cisco Analog Video Gateway	h264-v	udp	NTSC	4cif (704 x 480), 2cif (704 x 240), cif (352 x 240)
cisco_avg	195	Cisco Analog Video Gateway	h264-v	udp	PAL	4cif (704 x 576), 2cif (704 x 288), cif (352 x 288)
cisco_avg	195	Cisco Analog Video Gateway	mpeg4-v	udp	NTSC	4cif (704 x 480), 2cif (704 x 240), cif (352 x 240)

Table B-1 **Media Devices Supported by VSM**

Keyword	Model ID	Model Name	Media Type	Transport	Format	Resolutions
cisco_avg	195	Cisco Analog Video Gateway	mpeg4-v	udp	PAL	4cif (704 x 576), 2cif (704 x 288), cif (352 x 288)
cisco_avg	195	Cisco Analog Video Gateway	jpeg	tcp	NTSC	4cif (704 x 480), 2cif (704 x 240), cif (352 x 240)
cisco_avg	195	Cisco Analog Video Gateway	jpeg	tcp	PAL	4cif (704 x 576), 2cif (704 x 288), cif (352 x 288)
cisco_smsg	122	Cisco IP Gateway Encoder (CIVS-SGxx)	h264-v	udp	NTSC	d1 (720 x 480), 4cif (704 x 480), 2cif (704 x 240), hhr (352 x 480), cif (352 x 240)
cisco_smsg	122	Cisco IP Gateway Encoder (CIVS-SGxx)	h264-v	udp	PAL	d1 (720 x 576), 4cif (704 x 576), 2cif (704 x 288), hhr (352 x 576), cif (352 x 288)
cisco_smsg	122	Cisco IP Gateway Encoder (CIVS-SGxx)	mpeg4-v	udp	NTSC	d1 (720 x 480), 4cif (704 x 480), 2cif (704 x 240), hhr (352 x 480), cif (352 x 240)
cisco_smsg	122	Cisco IP Gateway Encoder (CIVS-SGxx)	mpeg4-v	udp	PAL	d1 (720 x 576), 4cif (704 x 576), 2cif (704 x 288), hhr (352 x 576), cif (352 x 288)
cisco-2500	110	Cisco SD IP Camera 2500/W Series	jpeg	tcp, udp	NTSC	d1 (720 x 480), 4cif (704 x 480), cif (352 x 240)
cisco-2500	110	Cisco SD IP Camera 2500/W Series	jpeg	tcp, udp	PAL	d1 (720 x 576), 4cif (704 x 576), cif (352 x 288)
cisco-2500	110	Cisco SD IP Camera 2500/W Series	mpeg4-v	tcp, udp	NTSC	d1 (720 x 480), 4cif (704 x 480), cif (352 x 240)
cisco-2500	110	Cisco SD IP Camera 2500/W Series	mpeg4-v	tcp, udp	PAL	d1 (720 x 576), 4cif (704 x 576), cif (352 x 288)
cohu	0	Cohu Analog Camera	N/A	N/A	N/A	N/A
cornet_cdx350	0	Cornet CDX-350	mpeg2-e	udp, multicast	NTSC	d1 (720 x 480), 4cif (640 x 480), cif (360 x 240), qcif (160 x 112)
cornet_cdx350	0	Cornet CDX-350	mpeg2-e	udp, multicast	PAL	d1 (720 x 576), qcif (160 x 144)
cornet_ivdo	0	Cornet iVDO Streamer 2/4	mpeg2-e	udp, multicast	NTSC	d1 (720 x 480), cif (352 x 240)
cornet_ivdo	0	Cornet iVDO Streamer 2/4	mpeg2-e	udp, multicast	PAL	d1 (720 x 576), cif (352 x 288)
cornet_ivdo	0	Cornet iVDO Streamer 2/4	mpeg4-v	udp, multicast	NTSC	d1 (720 x 480), cif (352 x 240)

Table B-1 Media Devices Supported by VSM

Keyword	Model ID	Model Name	Media Type	Transport	Format	Resolutions
cornet_ivdo	0	Cornet iVDO Streamer 2/4	mpeg4-v	udp, multicast	PAL	d1 (720 x 576), cif (352 x 288)
cornet_ivdo	0	Cornet iVDO Streamer 2/4	mpeg2-e	udp, multicast	NTSC	d1 (720 x 480), cif (352 x 240)
cornet_ivdo	0	Cornet iVDO Streamer 2/4	mpeg2-e	udp, multicast	PAL	d1 (720 x 576), cif (352 x 288)
cornet_ivdo	0	Cornet iVDO Streamer 2/4	mpeg4-v	udp, multicast	NTSC	d1 (720 x 480), cif (352 x 240)
cornet_ivdo	0	Cornet iVDO Streamer 2/4	mpeg4-v	udp, multicast	PAL	d1 (720 x 576), cif (352 x 288)
cornet_mpeg2	0	Cornet CDX-350	mpeg2-e	udp, multicast	NTSC	d1 (720 x 480), 4cif (640 x 480), cif (360 x 240), qcif (160 x 112)
cornet_mpeg2	0	Cornet CDX-350	mpeg2-e	udp, multicast	PAL	d1 (720 x 576), qcif (160 x 144)
exacq		Cisco Video Surveillance Video Encoder	jpeg	tcp	NTSC	4cif (720 x 480), 2cif (720 x 240), cif (352 x 240)
exacq		Cisco Video Surveillance Video Encoder	jpeg	tcp	PAL	4cif (720 x 576), 2cif (720 x 288), cif (352 x 288)
exacq		Cisco Video Surveillance Video Encoder	mpeg4-v	tcp	NTSC	4cif (720 x 480), 2cif (720 x 240), cif (352 x 240)
exacq		Cisco Video Surveillance Video Encoder	mpeg4-v	tcp	PAL	4cif (720 x 576), 2cif (720 x 288), cif (352 x 288)
generic	15	Pseudo-device for soft-triggered events	N/A	N/A	N/A	N/A
generic_mp4	0	Generic MPEG-4 Video (RTP/AVP/UDP)	mpeg4-v	udp, multicast	NTSC	d1 (720 x 480), 4cif (704 x 480), cif (352 x 240), qcif (176 x 120)
generic_mp4	0	Generic MPEG-4 Video (RTP/AVP/UDP)	mpeg4-v	udp, multicast	PAL	d1 (720 x 576), 4cif (704 x 576), cif (352 x 288), qcif (176 x 144)
icx_di5000	0	Icx DII IR/Visible Camera	N/A	N/A	N/A	N/A
indigo	0	IndigoVision Indigo8000	mpeg4-v	udp, multicast	NTSC	4cif (704 x 480), cif (352 x 240), qcif (176 x 120)
indigo	0	IndigoVision Indigo8000	mpeg4-v	udp, multicast	PAL	4cif (704 x 576), cif (352 x 288), qcif (176 x 144)

Table B-1 **Media Devices Supported by VSM**

Keyword	Model ID	Model Name	Media Type	Transport	Format	Resolutions
iqeye501	0	IQeye 501 Network Camera	jpeg	tcp	NTSC	1M (1280 x 1024), 4cif (640 x 512), cif (320 x 256), qcif (160 x 128)
iqeye510	0	IQeye 510 Network Camera	jpeg	tcp	NTSC	4cif (752 x 480), cif (376 x 240), qcif (188 x 120)
iqeye511	0	IQeye 511 Network Camera	jpeg	tcp	NTSC	1M (1280 x 1024), 4cif (640 x 512), cif (320 x 256), qcif (160 x 128)
iqeye602	97	IQeye 602 Network Camera	jpeg	tcp	NTSC	2M (1600 x 1200), 4cif (800 x 600), cif (400 x 296), qcif (200 x 144)
iqeye702	126	IQeye 702 Network Camera	jpeg	tcp	NTSC	2M (1600 x 1200), 4cif (800 x 600), cif (400 x 296), qcif (200 x 144)
iqeye703	98	IQeye 703 Network Camera	jpeg	tcp	NTSC	3M (2048 x 1536), 1M (1024 x 768), 2cif (512 x 384), cif (256 x 192)
iqeye705	131	IQeye 705 Network Camera	jpeg	tcp	NTSC	5M (2560 x 1920), 1M (1280 x 960), 4cif (640 x 480), cif (320 x 240)
iqeye711	127	IQeye 711 Network Camera	jpeg	tcp	NTSC	1M (1280 x 1024), 4cif (640 x 512), cif (320 x 256), qcif (160 x 128)
iqeye712d	134	IQeye 712D Network Camera	jpeg	tcp	NTSC	2M (1600 x 1200), 4cif (800 x 600), cif (400 x 300), qcif (200 x 150)
iqeye752	136	IQeye 752 Network Camera	jpeg	tcp	NTSC	2M (1600 x 1200), 4cif (800 x 600), cif (400 x 296), qcif (200 x 144)
iqeye753	128	IQeye 753 Network Camera	jpeg	tcp	NTSC	3M (2048 x 1536), 1M (1024 x 768), 2cif (512 x 384), cif (256 x 192)
iqeye755	132	IQeye 755 Network Camera	jpeg	tcp	NTSC	5M (2560 x 1920), 1M (1280 x 960), 4cif (640 x 480), cif (320 x 240)
iqeye802	138	IQeye 802 Network Camera	jpeg	tcp	NTSC	2M (1600 x 1200), 4cif (800 x 600), cif (400 x 296), qcif (200 x 144)
iqeye803	139	IQeye 803 Network Camera	jpeg	tcp	NTSC	3M (2048 x 1536), 1M (1024 x 768), 2cif (512 x 384), cif (256 x 192)
iqeye805	140	IQeye 805 Network Camera	jpeg	tcp	NTSC	5M (2560 x 1920), 1M (1280 x 960), 4cif (640 x 480), cif (320 x 240)

Table B-1 Media Devices Supported by VSM

Keyword	Model ID	Model Name	Media Type	Transport	Format	Resolutions
iqeye811	137	IQeye 811 Network Camera	jpeg	tcp	NTSC	1M (1280 x 1024), 4cif (640 x 512), cif (320 x 256), qcif (160 x 128)
iqeye852	141	IQeye 852 Network Camera	jpeg	tcp	NTSC	2M (1600 x 1200), 4cif (800 x 600), cif (400 x 296), qcif (200 x 144)
iqeye853	142	IQeye 853 Network Camera	jpeg	tcp	NTSC	3M (2048 x 1536), 1M (1024 x 768), 2cif (512 x 384), cif (256 x 192)
iqeye855	143	IQeye 855 Network Camera	jpeg	tcp	NTSC	5M (2560 x 1920), 1M (1280 x 960), 4cif (640 x 480), cif (320 x 240)
ivc_vs2101	0	IVnC VS2101	jpeg	tcp	NTSC	4cif (704 x 480), cif (352 x 240), qcif (176 x 112)
lumenara	0	Lumenara Ethernet Camera	jpeg	tcp	NTSC	sxga (1376 x 1032), 4cif (640 x 480), cif (320 x 240), qcif (160 x 120)
mango	0	Mango Raven-HX Video Server	audio	udp, multicast	N/A	N/A
mango	0	Mango Raven-HX Video Server	mpeg4-v	udp, multicast	NTSC	4cif (704 x 480), 2cif (704 x 240), cif (352 x 240), qcif (176 x 112)
mango	0	Mango Raven-HX Video Server	mpeg4-v	udp, multicast	PAL	4cif (704 x 576), 2cif (704 x 288), cif (352 x 288), qcif (176 x 144)
mango_hx	0	Mango Raven-HX Video Server	audio	udp, multicast	N/A	N/A
mango_hx	0	Mango Raven-HX Video Server	mpeg4-v	udp, multicast	NTSC	4cif (704 x 480), 2cif (704 x 240), cif (352 x 240), qcif (176 x 112)
mango_hx	0	Mango Raven-HX Video Server	mpeg4-v	udp, multicast	PAL	4cif (704 x 576), 2cif (704 x 288), cif (352 x 288), qcif (176 x 144)
mango_hx104	0	Mango Raven-HX104 Video Server	audio	udp, multicast	N/A	N/A
mango_hx104	0	Mango Raven-HX104 Video Server	mpeg4-v	udp, multicast	NTSC	4cif (704 x 480), 2cif (704 x 240), cif (352 x 240), qcif (176 x 112)
mango_hx104	0	Mango Raven-HX104 Video Server	mpeg4-v	udp, multicast	PAL	4cif (704 x 576), 2cif (704 x 288), cif (352 x 288), qcif (176 x 144)

Table B-1 **Media Devices Supported by VSM**

Keyword	Model ID	Model Name	Media Type	Transport	Format	Resolutions
mango_m	0	Mango Raven-M Video Server	audio	udp, multicast	N/A	N/A
mango_m	0	Mango Raven-M Video Server	mpeg4-v	udp, multicast	NTSC	4cif (704 x 480), 2cif (704 x 240), cif (352 x 240), qcif (176 x 112)
mango_m	0	Mango Raven-M Video Server	mpeg4-v	udp, multicast	PAL	4cif (704 x 576), 2cif (704 x 288), cif (352 x 288), qcif (176 x 144)
net300	0	Pelco PelcoNet NET300T	audio	udp	N/A	N/A
net300	0	Pelco PelcoNet NET300T	mpeg4-v	tcp, udp	NTSC	4cif (704 x 480), cif (352 x 240), qcif (176 x 128)
net300	0	Pelco PelcoNet NET300T	mpeg4-v	tcp, udp	PAL	4cif (704 x 576), cif (352 x 288), qcif (176 x 144)
net300-e	27	Pelco PelcoNet NET300T (events)	mpeg4-v	tcp, udp	NTSC	4cif (704 x 480), cif (352 x 240), qcif (176 x 128)
net300-e	27	Pelco PelcoNet NET300T (events)	mpeg4-v	tcp, udp	PAL	4cif (704 x 576), cif (352 x 288), qcif (176 x 144)
net350	0	Pelco PelcoNet NET350T	audio	udp	N/A	N/A
net350	0	Pelco PelcoNet NET350T	mpeg4-v	tcp, udp	NTSC	4cif (704 x 480), cif (352 x 240), qcif (176 x 128)
net350	0	Pelco PelcoNet NET350T	mpeg4-v	tcp, udp	PAL	4cif (704 x 576), cif (352 x 288), qcif (176 x 144)
net350-e	28	Pelco PelcoNet NET350T (events)	mpeg4-v	tcp, udp	NTSC	4cif (704 x 480), cif (352 x 240), qcif (176 x 128)
net350-e	28	Pelco PelcoNet NET350T (events)	mpeg4-v	tcp, udp	PAL	4cif (704 x 576), cif (352 x 288), qcif (176 x 144)
net4001a	0	Pelco PelcoNet NET4001A	audio	udp	N/A	N/A
net4001a-e	14	Pelco PelcoNet NET4001A (events)	mpeg2-v	tcp	NTSC	4cif (704 x 576), cif (352 x 288)
net4001a-e	14	Pelco PelcoNet NET4001A (events)	mpeg2-v	tcp	PAL	4cif (704 x 576), cif (352 x 288)
net4001a-e	14	Pelco PelcoNet NET4001A (events)	mpeg4-v	tcp	NTSC	4cif (704 x 576), cif (352 x 240), qcif (176 x 144)

Table B-1 Media Devices Supported by VSM

Keyword	Model ID	Model Name	Media Type	Transport	Format	Resolutions
net4001a-e	14	Pelco PelcoNet NET4001A (events)	mpeg4-v	tcp	PAL	4cif (704 x 576), cif (352 x 288), qcif (176 x 144)
nice_nvr	0	Nice DVR	jpeg	tcp	NTSC	4cif (704 x 480), cif (352 x 240), qcif (176 x 120)
optelecom_c20	0	Optelecom C20	audio	udp	N/A	N/A
optelecom_c20	0	Optelecom C20	mpeg2-e	udp, multicast	NTSC	d1 (720 x 480), 2/3d1 (480 x 480), 1/2d1 (352 x 480), cif (352 x 240)
optelecom_c20	0	Optelecom C20	mpeg2-e	udp, multicast	PAL	d1 (720 x 576), 2/3d1 (480 x 576), 1/2d1 (352 x 576), cif (352 x 288)
optelecom_c40	0	Optelecom C40	audio	udp	N/A	N/A
optelecom_c40	0	Optelecom C40	mpeg4-v	udp, multicast	NTSC	d1 (720 x 480), 2cif (720 x 240), cif (352 x 240)
optelecom_c40	0	Optelecom C40	mpeg4-v	udp, multicast	PAL	d1 (720 x 576), 2cif (720 x 288), cif (352 x 288)
optelecom_c44	0	Optelecom C44	mpeg4-v	udp, multicast	NTSC	d1 (720 x 480), 2cif (720 x 240), vga (640 x 480), 2/3d1 (480 x 480), 1/2d1 (352 x 480), cif (352 x 240), qvga (320 x 240), qcif (176 x 128)
optelecom_c44	0	Optelecom C44	mpeg4-v	udp, multicast	PAL	d1 (720 x 576), 2cif (720 x 288), vga (640 x 576), 2/3d1 (480 x 576), 1/2d1 (352 x 576), cif (352 x 288), qvga (352 x 240), qcif (176 x 144)
panasonic_cs954	0	Panasonic WV-CS954 Analog Camera	N/A	N/A	N/A	N/A
panasonic_nf_284	105	Panasonic WV-NF284 Network Camera	jpeg	tcp	NTSC	4cif (640 x 480), cif (320 x 240)
panasonic_nf_284	105	Panasonic WV-NF284 Network Camera	mpeg4-v	udp, multicast	NTSC	4cif (640 x 480), cif (320 x 240)
panasonic_nf_302	133	Panasonic WV-NF302 Network Camera	jpeg	tcp	NTSC	1M (1280 x 960), 4cif (640 x 480), cif (320 x 240)
panasonic_nf_302	133	Panasonic WV-NF302 Network Camera	mpeg4-v	udp, multicast	NTSC	1M (1280 x 960), 4cif (640 x 480), cif (320 x 240)
panasonic_np_1004	103	Panasonic WV-NP1004 Network Camera	jpeg	tcp	NTSC	2M (1280 x 960), 1M (960 x 720), 4cif (640 x 480), cif (320 x 240)

Table B-1 **Media Devices Supported by VSM**

Keyword	Model ID	Model Name	Media Type	Transport	Format	Resolutions
panasonic_np_1004	103	Panasonic WV-NP1004 Network Camera	mpeg4-v	udp, multicast	NTSC	4cif (640 x 480), cif (320 x 240)
panasonic_np_244	107	Panasonic WV-NP244 Network Camera	jpeg	tcp	NTSC	4cif (640 x 480), cif (320 x 240)
panasonic_np_244	107	Panasonic WV-NP244 Network Camera	mpeg4-v	udp, multicast	NTSC	4cif (640 x 480), cif (320 x 240)
panasonic_np_304	148	Panasonic WV-NP304 Network Camera	jpeg	tcp	NTSC	1M (1280 x 960), 4cif (640 x 480), cif (320 x 240)
panasonic_np_304	148	Panasonic WV-NP304 Network Camera	mpeg4-v	udp, multicast	NTSC	1M (1280 x 960), 4cif (640 x 480), cif (320 x 240)
panasonic_ns_202	104	Panasonic WV-NS202 Network Camera	jpeg	tcp	NTSC	4cif (640 x 480), cif (320 x 240)
panasonic_ns_202	104	Panasonic WV-NS202 Network Camera	mpeg4-v	udp, multicast	NTSC	4cif (640 x 480), cif (320 x 240)
panasonic_ns_202a	129	Panasonic WV-NS202A Network Camera	jpeg	tcp	NTSC	4cif (640 x 480), cif (320 x 240)
panasonic_ns_202a	129	Panasonic WV-NS202A Network Camera	mpeg4-v	udp, multicast	NTSC	4cif (640 x 480), cif (320 x 240)
panasonic_ns_954	152	Panasonic WV-NS954 Network Camera	jpeg	tcp	NTSC	4cif (640 x 480), cif (320 x 240)
panasonic_ns_954	152	Panasonic WV-NS954 Network Camera	mpeg4-v	udp, multicast	NTSC	4cif (640 x 480), cif (320 x 240)
panasonic_ns_964	153	Panasonic WV-NW964 Network Camera	jpeg	tcp	NTSC	4cif (640 x 480), cif (320 x 240)
panasonic_ns_964	153	Panasonic WV-NW964 Network Camera	mpeg4-v	udp, multicast	NTSC	4cif (640 x 480), cif (320 x 240)
panasonic_nw_484s	130	Panasonic WV-NW484S Network Camera	jpeg	tcp	NTSC	4cif (640 x 480), cif (320 x 240)

Table B-1 Media Devices Supported by VSM

Keyword	Model ID	Model Name	Media Type	Transport	Format	Resolutions
panasonic_nw_484s	130	Panasonic WV-NW484S Network Camera	mpeg4-v	udp, multicast	NTSC	4cif (640 x 480), cif (320 x 240)
pelco_d	0	Pelco Analog Camera (D protocol)	N/A	N/A	N/A	N/A
pelco_endura_analog	0	Pelco Endura Analog Camera)	N/A	N/A	N/A	N/A
pelco_ip_spectra_iv	0	Pelco Spectra IV IP Camera	mpeg4-v	udp	NTSC	4cif (704 x 480), 2cif (704 x 240), cif (352 x 240), qcif (176 x 112)
pelco_ip_spectra_iv	0	Pelco Spectra IV IP Camera	mpeg4-v	udp	PAL	4cif (704 x 576), 2cif (704 x 288), cif (352 x 288), qcif (176 x 144)
Note Cisco VSM supports the Pelco Spectra IV IP Cameras only with the Pelco TXB-IP encoder board. Cisco VSM does NOT support these cameras with the Pelco TXB-N encoder board (the cameras will fail to communicate with Cisco VSM using a Pelco TXB-N encoder).						
pelco_ip110	0	Pelco IP110 Network Camera	jpeg	tcp	NTSC	4cif (704 x 480)
pelco_ip110	0	Pelco IP110 Network Camera	jpeg	tcp	PAL	4cif (704 x 480)
pelco_ip110	0	Pelco IP110 Network Camera	mpeg4-v	udp	NTSC	4cif (704 x 480), 2cif (704 x 240), cif (352 x 240), qcif (176 x 112)
pelco_ip110	0	Pelco IP110 Network Camera	mpeg4-v	udp	PAL	4cif (704 x 576), 2cif (704 x 288), cif (352 x 288), qcif (176 x 144)
pelco_miniSpectra	0	Pelco Mini Spectra Dome Analog Camera	N/A	N/A	N/A	N/A
pelco_net53xxt	0	Pelco NET53XXT Encoder	mpeg4-v	udp	NTSC	4cif (704 x 480), 2cif (704 x 240), cif (352 x 240), qcif (176 x 112)
pelco_net53xxt	0	Pelco NET53XXT Encoder	mpeg4-v	udp	PAL	4cif (704 x 576), 2cif (704 x 288), cif (352 x 288), qcif (176 x 144)
pelco_p	0	Pelco Analog Camera (P protocol)	N/A	N/A	N/A	N/A
pelcoSM	0	Pelco System Manager	mpeg4-v	udp	NTSC	4cif (704 x 480), 2cif (704 x 240), cif (352 x 240), qcif (176 x 112)
pelcoSM	0	Pelco System Manager	mpeg4-v	udp	PAL	4cif (704 x 576), 2cif (704 x 288), cif (352 x 288), qcif (176 x 144)
proxy	0	Existing BroadWare Proxy Source	audio	tcp	N/A	N/A

Table B-1 **Media Devices Supported by VSM**

Keyword	Model ID	Model Name	Media Type	Transport	Format	Resolutions
smartsight1500	0	SmartSight 15xx	mpeg4-v.h	udp, multicast	NTSC	2cif (704 x 240), cif (352 x 240), qcif (176 x 128)
smartsight1500	0	SmartSight 15xx	mpeg4-v.h	udp, multicast	PAL	2cif (704 x 288), cif (352 x 288), qcif (176 x 144)
smartsight1700	0	SmartSight 17xx	mpeg4-v	udp, multicast	NTSC	4cif (704 x 480), 2cif (704 x 240), cif (352 x 240)
smartsight1700	0	SmartSight 17xx	mpeg4-v	udp, multicast	PAL	4cif (704 x 576), 2cif (704 x 288), cif (352 x 288)
smartsight1700	0	SmartSight 17xx	mpeg4-v.h	udp, multicast	NTSC	4cif (704 x 480), 2cif (704 x 240), cif (352 x 240)
smartsight1700	0	SmartSight 17xx	mpeg4-v.h	udp, multicast	PAL	4cif (704 x 576), 2cif (704 x 288), cif (352 x 288)
smartsight1900	0	SmartSight 19xx	mpeg4-v.h	udp, multicast	NTSC	4cif (704 x 480), cif (352 x 240)
smartsight1900	0	SmartSight 19xx	mpeg4-v.h	udp, multicast	PAL	4cif (704 x 576), cif (352 x 288)
smartsight2600	0	SmartSight 26xx Network Camera	mpeg4-v	udp, multicast	NTSC	4cif (640 x 480), cif (352 x 240)
smartsight2600	0	SmartSight 26xx Network Camera	mpeg4-v	udp, multicast	PAL	4cif (640 x 576), cif (352 x 288)
smartsight2600	0	SmartSight 26xx Network Camera	mpeg4-v.h	udp, multicast	NTSC	4cif (640 x 480), cif (352 x 240)
smartsight2600	0	SmartSight 26xx Network Camera	mpeg4-v.h	udp, multicast	PAL	4cif (640 x 576), cif (352 x 288)
sony_snc_cm120	155	Sony SNC-CM120 Network Camera	jpeg	tcp	NTSC	1M (1280 x 960), 4cif (640 x 480), cif (320 x 240)
sony_snc_cm120	155	Sony SNC-CM120 Network Camera	jpeg	tcp	PAL	4cif (640 x 480), cif (320 x 240)
sony_snc_cm120	155	Sony SNC-CM120 Network Camera	mpeg4-v	tcp, udp	NTSC	4cif (640 x 480), cif (320 x 240)
sony_snc_cm120	155	Sony SNC-CM120 Network Camera	mpeg4-v	tcp, udp	PAL	4cif (640 x 480), cif (320 x 240)
sony_snc_cs10	166	Sony SNC-CS10 Network Camera	jpeg	tcp	NTSC	4cif (640 x 480), cif (320 x 240), qcif (160 x 120)
sony_snc_cs10	166	Sony SNC-CS10 Network Camera	jpeg	tcp	PAL	4cif (640 x 480), cif (320 x 240), qcif (160 x 120)
sony_snc_cs10	166	Sony SNC-CS10 Network Camera	mpeg4-v	udp	NTSC	4cif (640 x 480), cif (320 x 240), qcif (160 x 120)

Table B-1 Media Devices Supported by VSM

Keyword	Model ID	Model Name	Media Type	Transport	Format	Resolutions
sony_snc_cs10	166	Sony SNC-CS10 Network Camera	mpeg4-v	udp	PAL	4cif (640 x 480), cif (320 x 240), qcif (160 x 120)
sony_snc_cs11	75	Sony SNC-CS11 Network Camera	jpeg	tcp	NTSC	4cif (640 x 480), cif (320 x 240), qcif (160 x 120)
sony_snc_cs11	75	Sony SNC-CS11 Network Camera	jpeg	tcp	PAL	4cif (640 x 480), cif (320 x 240), qcif (160 x 120)
sony_snc_cs11	75	Sony SNC-CS11 Network Camera	mpeg4-v	udp	NTSC	4cif (640 x 480), cif (320 x 240), qcif (160 x 120)
sony_snc_cs11	75	Sony SNC-CS11 Network Camera	mpeg4-v	udp	PAL	4cif (640 x 480), cif (320 x 240), qcif (160 x 120)
sony_snc_cs20	154	Sony SNC-CS20 Network Camera	jpeg	tcp	NTSC	4cif (640 x 480), cif (320 x 240)
sony_snc_cs20	154	Sony SNC-CS20 Network Camera	jpeg	tcp	PAL	4cif (640 x 480), cif (320 x 240)
sony_snc_cs20	154	Sony SNC-CS20 Network Camera	mpeg4-v	tcp, udp	NTSC	4cif (640 x 480), cif (320 x 240)
sony_snc_cs20	154	Sony SNC-CS20 Network Camera	mpeg4-v	tcp, udp	PAL	4cif (640 x 480), cif (320 x 240)
sony_snc_cs3	49	Sony SNC-CS3 Network Camera	jpeg	tcp	NTSC	4cif (640 x 480), cif (320 x 240), qcif (160 x 120)
sony_snc_cs3	49	Sony SNC-CS3 Network Camera	jpeg	tcp	PAL	4cif (640 x 480), cif (320 x 240), qcif (160 x 120)
sony_snc_cs50	74	Sony SNC-CS50 Network Camera	h264-v	tcp, udp	NTSC	4cif (640 x 480), cif (320 x 240), qcif (160 x 120)
sony_snc_cs50	74	Sony SNC-CS50 Network Camera	h264-v	tcp, udp	PAL	4cif (640 x 480), cif (320 x 240), qcif (160 x 120)
sony_snc_cs50	74	Sony SNC-CS50 Network Camera	mpeg4-v	tcp, udp	NTSC	4cif (640 x 480), cif (320 x 240), qcif (160 x 120)
sony_snc_cs50	74	Sony SNC-CS50 Network Camera	mpeg4-v	tcp, udp	PAL	4cif (640 x 480), cif (320 x 240), qcif (160 x 120)
sony_snc_cs50	74	Sony SNC-CS50 Network Camera	jpeg	tcp	NTSC	4cif (640 x 480), cif (320 x 240), qcif (160 x 120)
sony_snc_cs50	74	Sony SNC-CS50 Network Camera	jpeg	tcp	PAL	4cif (640 x 480), cif (320 x 240), qcif (160 x 120)
sony_snc_df40	47	Sony SNC-DF40 Network Camera	jpeg	tcp	NTSC	4cif (640 x 480), cif (320 x 240), qcif (160 x 120)
sony_snc_df40	47	Sony SNC-DF40 Network Camera	jpeg	tcp	PAL	4cif (640 x 480), cif (320 x 240), qcif (160 x 120)
sony_snc_df40	47	Sony SNC-DF40 Network Camera	mpeg4-v	udp	NTSC	4cif (640 x 480), cif (320 x 240), qcif (160 x 120)
sony_snc_df40	47	Sony SNC-DF40 Network Camera	mpeg4-v	udp	PAL	4cif (640 x 480), cif (320 x 240), qcif (160 x 120)

Table B-1 **Media Devices Supported by VSM**

Keyword	Model ID	Model Name	Media Type	Transport	Format	Resolutions
sony_snc_df50	165	Sony SNC-DF50 Network Camera	h264-v	tcp, udp	NTSC	4cif (640 x 480), cif (320 x 240), qcif (160 x 120)
sony_snc_df50	165	Sony SNC-DF50 Network Camera	h264-v	tcp, udp	PAL	4cif (640 x 480), cif (320 x 240), qcif (160 x 120)
sony_snc_df50	165	Sony SNC-DF50 Network Camera	mpeg4-v	tcp, udp	NTSC	4cif (640 x 480), cif (320 x 240), qcif (160 x 120)
sony_snc_df50	165	Sony SNC-DF50 Network Camera	mpeg4-v	tcp, udp	PAL	4cif (640 x 480), cif (320 x 240), qcif (160 x 120)
sony_snc_df50	165	Sony SNC-DF50 Network Camera	jpeg	tcp	NTSC	4cif (640 x 480), cif (320 x 240), qcif (160 x 120)
sony_snc_df50	165	Sony SNC-DF50 Network Camera	jpeg	tcp	PAL	4cif (640 x 480), cif (320 x 240), qcif (160 x 120)
sony_snc_df70	48	Sony SNC-DF70 Network Camera	jpeg	tcp	NTSC	4cif (640 x 480), cif (320 x 240), qcif (160 x 120)
sony_snc_df70	48	Sony SNC-DF70 Network Camera	jpeg	tcp	PAL	4cif (640 x 480), cif (320 x 240), qcif (160 x 120)
sony_snc_df70	48	Sony SNC-DF70 Network Camera	mpeg4-v	udp	NTSC	4cif (640 x 480), cif (320 x 240), qcif (160 x 120)
sony_snc_df70	48	Sony SNC-DF70 Network Camera	mpeg4-v	udp	PAL	4cif (640 x 480), cif (320 x 240), qcif (160 x 120)
sony_snc_df80	121	Sony SNC-DF80 Network Camera	h264-v	tcp, udp	NTSC	4cif (640 x 480), cif (320 x 240), qcif (160 x 120)
sony_snc_df80	121	Sony SNC-DF80 Network Camera	h264-v	tcp, udp	PAL	4cif (640 x 480), cif (320 x 240), qcif (160 x 120)
sony_snc_df80	121	Sony SNC-DF80 Network Camera	mpeg4-v	tcp, udp	NTSC	4cif (640 x 480), cif (320 x 240), qcif (160 x 120)
sony_snc_df80	121	Sony SNC-DF80 Network Camera	mpeg4-v	tcp, udp	PAL	4cif (640 x 480), cif (320 x 240), qcif (160 x 120)
sony_snc_df80	121	Sony SNC-DF80 Network Camera	jpeg	tcp	NTSC	4cif (640 x 480), cif (320 x 240), qcif (160 x 120)
sony_snc_df80	121	Sony SNC-DF80 Network Camera	jpeg	tcp	PAL	4cif (640 x 480), cif (320 x 240), qcif (160 x 120)
sony_snc_df85	121	Sony SNC-DF85 Network Camera	jpeg	tcp	NTSC	4cif (640 x 480), cif (320 x 240), qcif (160 x 120)
sony_snc_df85	121	Sony SNC-DF85 Network Camera	jpeg	tcp	PAL	4cif (640 x 480), cif (320 x 240), qcif (160 x 120)
sony_snc_df85	121	Sony SNC-DF85 Network Camera	mpeg4-v	tcp, udp	NTSC	4cif (640 x 480), cif (320 x 240), qcif (160 x 120)
sony_snc_df85	121	Sony SNC-DF85 Network Camera	mpeg4-v	tcp, udp	PAL	4cif (640 x 480), cif (320 x 240), qcif (160 x 120)

Table B-1 Media Devices Supported by VSM

Keyword	Model ID	Model Name	Media Type	Transport	Format	Resolutions
sony_snc_dm110	157	Sony SNC-DM110 Network Camera	jpeg	tcp	NTSC	1M (1280 x 960), 4cif (640 x 480), cif (320 x 240)
sony_snc_dm110	157	Sony SNC-DM110 Network Camera	jpeg	tcp	PAL	4cif (640 x 480), cif (320 x 240)
sony_snc_dm110	157	Sony SNC-DM110 Network Camera	mpeg4-v	tcp, udp	NTSC	4cif (640 x 480), cif (320 x 240)
sony_snc_dm110	157	Sony SNC-DM110 Network Camera	mpeg4-v	tcp, udp	PAL	4cif (640 x 480), cif (320 x 240)
sony_snc_dm160	159	Sony SNC-DM160 Network Camera	jpeg	tcp	NTSC	1M (1280 x 960), 4cif (640 x 480), cif (320 x 240)
sony_snc_dm160	159	Sony SNC-DM160 Network Camera	jpeg	tcp	PAL	4cif (640 x 480), cif (320 x 240)
sony_snc_dm160	159	Sony SNC-DM160 Network Camera	mpeg4-v	tcp, udp	NTSC	4cif (640 x 480), cif (320 x 240)
sony_snc_dm160	159	Sony SNC-DM160 Network Camera	mpeg4-v	tcp, udp	PAL	4cif (640 x 480), cif (320 x 240)
sony_snc_ds10	156	Sony SNC-DS10 Network Camera	jpeg	tcp	NTSC	4cif (640 x 480), cif (320 x 240)
sony_snc_ds10	156	Sony SNC-DS10 Network Camera	jpeg	tcp	PAL	4cif (640 x 480), cif (320 x 240)
sony_snc_ds10	156	Sony SNC-DS10 Network Camera	mpeg4-v	tcp, udp	NTSC	4cif (640 x 480), cif (320 x 240)
sony_snc_ds10	156	Sony SNC-DS10 Network Camera	mpeg4-v	tcp, udp	PAL	4cif (640 x 480), cif (320 x 240)
sony_snc_ds60	158	Sony SNC-DS60 Network Camera	jpeg	tcp	NTSC	4cif (640 x 480), cif (320 x 240)
sony_snc_ds60	158	Sony SNC-DS60 Network Camera	jpeg	tcp	PAL	4cif (640 x 480), cif (320 x 240)
sony_snc_ds60	158	Sony SNC-DS60 Network Camera	mpeg4-v	tcp, udp	NTSC	4cif (640 x 480), cif (320 x 240)
sony_snc_ds60	158	Sony SNC-DS60 Network Camera	mpeg4-v	tcp, udp	PAL	4cif (640 x 480), cif (320 x 240)
sony_snc_p1	163	Sony SNC-P1 Network Camera	jpeg	tcp	NTSC	4cif (640 x 480), cif (320 x 240), qcif (160 x 120)

Table B-1 **Media Devices Supported by VSM**

Keyword	Model ID	Model Name	Media Type	Transport	Format	Resolutions
sony_snc_p1	163	Sony SNC-P1 Network Camera	jpeg	tcp	PAL	4cif (640 x 480), cif (320 x 240), qcif (160 x 120)
sony_snc_p1	163	Sony SNC-P1 Network Camera	mpeg4-v	udp	NTSC	4cif (640 x 480), cif (320 x 240), qcif (160 x 120)
sony_snc_p1	163	Sony SNC-P1 Network Camera	mpeg4-v	udp	PAL	4cif (640 x 480), cif (320 x 240), qcif (160 x 120)
sony_snc_rx530	120	Sony SNC-RX530 Network Camera	h264-v	tcp, udp	NTSC	4cif (640 x 480), cif (320 x 240), qcif (160 x 120)
sony_snc_rx530	120	Sony SNC-RX530 Network Camera	h264-v	tcp, udp	PAL	4cif (640 x 480), cif (320 x 240), qcif (160 x 120)
sony_snc_rx530	120	Sony SNC-RX530 Network Camera	mpeg4-v	tcp, udp	NTSC	4cif (640 x 480), cif (320 x 240), qcif (160 x 120)
sony_snc_rx530	120	Sony SNC-RX530 Network Camera	mpeg4-v	tcp, udp	PAL	4cif (640 x 480), cif (320 x 240), qcif (160 x 120)
sony_snc_rx530	120	Sony SNC-RX530 Network Camera	jpeg	tcp	NTSC	4cif (640 x 480), cif (320 x 240), qcif (160 x 120)
sony_snc_rx530	120	Sony SNC-RX530 Network Camera	jpeg	tcp	PAL	4cif (640 x 480), cif (320 x 240), qcif (160 x 120)
sony_snc_rx550	72	Sony SNC-RX550 Network Camera	h264-v	tcp, udp	NTSC	4cif (640 x 480), cif (320 x 240), qcif (160 x 120)
sony_snc_rx550	72	Sony SNC-RX550 Network Camera	h264-v	tcp, udp	PAL	4cif (640 x 480), cif (320 x 240), qcif (160 x 120)
sony_snc_rx550	72	Sony SNC-RX550 Network Camera	mpeg4-v	tcp, udp	NTSC	4cif (640 x 480), cif (320 x 240), qcif (160 x 120)
sony_snc_rx550	72	Sony SNC-RX550 Network Camera	mpeg4-v	tcp, udp	PAL	4cif (640 x 480), cif (320 x 240), qcif (160 x 120)
sony_snc_rx550	72	Sony SNC-RX550 Network Camera	jpeg	tcp	NTSC	4cif (640 x 480), cif (320 x 240), qcif (160 x 120)
sony_snc_rx550	72	Sony SNC-RX550 Network Camera	jpeg	tcp	PAL	4cif (640 x 480), cif (320 x 240), qcif (160 x 120)

Table B-1 Media Devices Supported by VSM

Keyword	Model ID	Model Name	Media Type	Transport	Format	Resolutions
sony_snc_rx550-e	38	Sony SNC-RX550 (events) Network Camera	h264-v	tcp, udp	NTSC	4cif (640 x 480), cif (320 x 240), qcif (160 x 120)
sony_snc_rx550-e	38	Sony SNC-RX550 (events) Network Camera	h264-v	tcp, udp	PAL	4cif (640 x 480), cif (320 x 240), qcif (160 x 120)
sony_snc_rx550-e	38	Sony SNC-RX550 (events) Network Camera	mpeg4-v	tcp, udp	NTSC	4cif (640 x 480), cif (320 x 240), qcif (160 x 120)
sony_snc_rx550-e	38	Sony SNC-RX550 (events) Network Camera	mpeg4-v	tcp, udp	PAL	4cif (640 x 480), cif (320 x 240), qcif (160 x 120)
sony_snc_rx550-e	38	Sony SNC-RX550 (events) Network Camera	jpeg	tcp	NTSC	4cif (640 x 480), cif (320 x 240), qcif (160 x 120)
sony_snc_rx550-e	38	Sony SNC-RX550 (events) Network Camera	jpeg	tcp	PAL	4cif (640 x 480), cif (320 x 240), qcif (160 x 120)
sony_snc_rx570	123	Sony SNC-RX570 Network Camera	h264-v	tcp, udp	NTSC	4cif (640 x 480), cif (320 x 240), qcif (160 x 120)
sony_snc_rx570	123	Sony SNC-RX570 Network Camera	h264-v	tcp, udp	PAL	4cif (640 x 480), cif (320 x 240), qcif (160 x 120)
sony_snc_rx570	123	Sony SNC-RX570 Network Camera	mpeg4-v	tcp, udp	NTSC	4cif (640 x 480), cif (320 x 240), qcif (160 x 120)
sony_snc_rx570	123	Sony SNC-RX570 Network Camera	mpeg4-v	tcp, udp	PAL	4cif (640 x 480), cif (320 x 240), qcif (160 x 120)
sony_snc_rx570	123	Sony SNC-RX570 Network Camera	jpeg	tcp	NTSC	4cif (640 x 480), cif (320 x 240), qcif (160 x 120)
sony_snc_rx570	123	Sony SNC-RX570 Network Camera	jpeg	tcp	PAL	4cif (640 x 480), cif (320 x 240), qcif (160 x 120)
sony_snc_rz25	70	Sony SNC-RZ25 Network Camera	jpeg	tcp	NTSC	4cif (640 x 480), cif (320 x 240), qcif (160 x 120)

Table B-1 **Media Devices Supported by VSM**

Keyword	Model ID	Model Name	Media Type	Transport	Format	Resolutions
sony_snc_rz25	70	Sony SNC-RZ25 Network Camera	jpeg	tcp	PAL	4cif (640 x 480), cif (320 x 240), qcif (160 x 120)
sony_snc_rz25	70	Sony SNC-RZ25 Network Camera	mpeg4-v	udp	NTSC	4cif (640 x 480), cif (320 x 240)
sony_snc_rz25	70	Sony SNC-RZ25 Network Camera	mpeg4-v	udp	PAL	4cif (640 x 480), cif (320 x 240)
sony_snc_rz30	43	Sony SNC-RZ30 Network Camera	jpeg	tcp	NTSC	4cif (640 x 480), cif (320 x 240), qcif (160 x 120)
sony_snc_rz30	43	Sony SNC-RZ30 Network Camera	jpeg	tcp	PAL	4cif (640 x 480), cif (320 x 240), qcif (160 x 120)
sony_snc_rz50	71	Sony SNC-RZ50 Dome PTZ Network Camera	h264-v	tcp, udp	NTSC	4cif (640 x 480), cif (320 x 240), qcif (160 x 120)
sony_snc_rz50	71	Sony SNC-RZ50 Dome PTZ Network Camera	h264-v	tcp, udp	PAL	4cif (640 x 480), cif (320 x 240), qcif (160 x 120)
sony_snc_rz50	71	Sony SNC-RZ50 Dome PTZ Network Camera	mpeg4-v	tcp, udp	NTSC	4cif (640 x 480), cif (320 x 240), qcif (160 x 120)
sony_snc_rz50	71	Sony SNC-RZ50 Dome PTZ Network Camera	mpeg4-v	tcp, udp	PAL	4cif (640 x 480), cif (320 x 240), qcif (160 x 120)
sony_snc_rz50	71	Sony SNC-RZ50 Dome PTZ Network Camera	jpeg	tcp	NTSC	4cif (640 x 480), cif (320 x 240), qcif (160 x 120)
sony_snc_rz50	71	Sony SNC-RZ50 Dome PTZ Network Camera	jpeg	tcp	PAL	4cif (640 x 480), cif (320 x 240), qcif (160 x 120)
sony_snc_z20	164	Sony SNC-Z20 Network Camera	jpeg	tcp	NTSC	4cif (640 x 480), cif (320 x 240), qcif (160 x 120)
sony_snc_z20	164	Sony SNC-Z20 Network Camera	jpeg	tcp	PAL	4cif (640 x 480), cif (320 x 240), qcif (160 x 120)
teleste_ipe	0	Teleste IPE	mpeg2-t	udp, multicast	NTSC	4cif (720 x 480)
teleste_mpce4	0	Teleste MPC-E4	audio	udp	N/A	N/A
teleste_mpce4	0	Teleste MPC-E4	mpeg4-v	udp, multicast	NTSC	4cif (704 x 480), 2cif (704 x 240), cif (352 x 240), qcif (176 x 120)
teleste_mpce4	0	Teleste MPC-E4	mpeg4-v	udp, multicast	PAL	4cif (704 x 576), 2cif (704 x 288), cif (352 x 288), qcif (176 x 144)
teleste_mpce4	0	Teleste MPC-E4	jpeg	udp	NTSC	4cif (704 x 480), 2cif (704 x 240), cif (352 x 240), qcif (176 x 120)
teleste_mpce4	0	Teleste MPC-E4	jpeg	udp	PAL	4cif (704 x 576), 2cif (704 x 288), cif (352 x 288), qcif (176 x 144)

Table B-1 *Media Devices Supported by VSM*

Keyword	Model ID	Model Name	Media Type	Transport	Format	Resolutions
toshiba_wb21a	0	Toshiba IK-WB21A Network Camera	jpeg	tcp	NTSC	1M (1280 x 960), 4cif (640 x 480), cif (320 x 240), qcif (160 x 120)
toshiba_wr01	0	Toshiba IK-WR01A Network Camera	jpeg	tcp	NTSC	4cif (640 x 480), cif (320 x 240), qcif (160 x 120)
vbrick	0	VBrick MPEG-4 Encoder	mpeg4-v	tcp, udp, multicast	NTSC	cif (352 x 240), qcif (176 x 128)
vbrick	0	VBrick MPEG-4 Encoder	mpeg4-v	tcp, udp, multicast	PAL	cif (352 x 288), qcif (176 x 144)
videojet10	0	Bosch VideoJet 10	audio	udp	N/A	N/A
videojet10	0	Bosch VideoJet 10	mpeg4-v	tcp, udp	NTSC	4cif (704 x 480), cif (352 x 240), qcif (176 x 128)
videojet10	0	Bosch VideoJet 10	mpeg4-v	tcp, udp	PAL	4cif (704 x 576), cif (352 x 288), qcif (176 x 144)
videojet10	0	Bosch VideoJet 10	jpeg	tcp	NTSC	cif (352 x 288)
videojet10	0	Bosch VideoJet 10	jpeg	tcp	PAL	cif (352 x 288)
videojet1000	0	Bosch VideoJet 1000	audio	udp	N/A	N/A
videojet1000-e	13	Bosch VideoJet 1000 (events)	mpeg2-v	tcp	NTSC	4cif (720 x 576), cif (352 x 288), qcif (176 x 144)
videojet1000-e	13	Bosch VideoJet 1000 (events)	mpeg2-v	tcp	PAL	4cif (720 x 576), cif (352 x 288), qcif (176 x 144)
videojet1000-e	13	Bosch VideoJet 1000 (events)	mpeg4-v	tcp, udp	NTSC	4cif (704 x 576), cif (352 x 288), qcif (176 x 144)
videojet1000-e	13	Bosch VideoJet 1000 (events)	mpeg4-v	tcp, udp	PAL	4cif (704 x 576), cif (352 x 288), qcif (176 x 144)
videojet10-e	24	Bosch VideoJet 10 (events)	mpeg4-v	tcp, udp	NTSC	4cif (704 x 480), cif (352 x 240), qcif (176 x 128)
videojet10-e	24	Bosch VideoJet 10 (events)	mpeg4-v	tcp, udp	PAL	4cif (704 x 576), cif (352 x 288), qcif (176 x 144)
vip_x1	0	Bosch VIP X1	mpeg4-v	tcp, udp	NTSC	d1 (704 x 480), cif (352 x 288)
vip_x1	0	Bosch VIP X1	mpeg4-v	tcp, udp	PAL	d1 (704 x 480), cif (352 x 288)
vip_x2	0	Bosch VIP X2	mpeg4-v	tcp, udp	NTSC	d1 (704 x 480), cif (352 x 240)
vip_x2	0	Bosch VIP X2	mpeg4-v	tcp, udp	PAL	d1 (704 x 576), cif (352 x 288)
vip10	0	Bosch VIP 10	audio	udp	N/A	N/A
vip10	0	Bosch VIP 10	mpeg4-v	tcp, udp	NTSC	4cif (704 x 480), cif (352 x 240), qcif (176 x 128)

Table B-1 **Media Devices Supported by VSM**

Keyword	Model ID	Model Name	Media Type	Transport	Format	Resolutions
vip10	0	Bosch VIP 10	mpeg4-v	tcp, udp	PAL	4cif (704 x 576), cif (352 x 288), qcif (176 x 144)
vip10	0	Bosch VIP 10	jpeg	tcp	NTSC	cif (352 x 288)
vip10	0	Bosch VIP 10	jpeg	tcp	PAL	cif (352 x 288)
vip1000	0	Bosch VIP 1000	audio	udp	N/A	N/A
vip1000	0	Bosch VIP 1000	mpeg4-v	tcp, udp	NTSC	4cif (704 x 480), cif (352 x 240), qcif (176 x 128)
vip1000	0	Bosch VIP 1000	mpeg4-v	tcp, udp	PAL	4cif (704 x 576), cif (352 x 288), qcif (176 x 144)
vip1000	0	Bosch VIP 1000	jpeg	tcp	NTSC	cif (352 x 288)
vip1000	0	Bosch VIP 1000	jpeg	tcp	PAL	cif (352 x 288)
vip1000-e	26	Bosch VIP 1000 (events)	mpeg4-v	tcp, udp	NTSC	4cif (704 x 480), cif (352 x 240), qcif (176 x 128)
vip1000-e	26	Bosch VIP 1000 (events)	mpeg4-v	tcp, udp	PAL	4cif (704 x 576), cif (352 x 288), qcif (176 x 144)
vip10-e	25	Bosch VIP 10 (events)	mpeg4-v	tcp, udp	NTSC	4cif (704 x 480), cif (352 x 240), qcif (176 x 128)
vip10-e	25	Bosch VIP 10 (events)	mpeg4-v	tcp, udp	PAL	4cif (704 x 576), cif (352 x 288), qcif (176 x 144)
vjet_x40	0	Bosch VideoJet X40	mpeg4-v	tcp, udp	NTSC	4cif (704 x 480), 2cif (704 x 240), cif (352 x 240), qcif (176 x 120)



INDEX

A

addToSync [6-22](#)
archive
 clip, creating [4-22](#)
 creating [6-51](#)
 details [4-15, 4-18](#)
 listing
 all [4-12](#)
 running [4-13](#)
 mediatype [4-14](#)
 monitoring
 detail [4-20](#)
 summary [4-21](#)
 play rate
 getting [6-37](#)
 setting [6-17](#)
 removing [4-11](#)
 renaming [4-8](#)
 seeking to
 percentage [6-20](#)
 UTC time [6-19](#)
 segment loop, playing [6-18](#)
 starting [4-2](#)
 stopping [4-10](#)
 updating [4-5](#)
archive commands
 overview [1-2](#)
 summary [4-1](#)
archive video
 one frame in playback direction [6-11](#)
 one frame opposite playback direction [6-12](#)
AxClient

information [6-25](#)
version number [6-27](#)
AxClient API methods
 overview [1-2](#)
 programming notes
 C# [1-3](#)
 JavaScript [1-3](#)
 summary [6-1](#)

B

base rectangle color
 getting [6-63](#)
 setting [6-62](#)

C

C#, programming notes [1-3](#)
callback
 invoking when
 archive starts [6-77](#)
 archive stops [6-76](#)
 playback starts [6-84](#)
 playback state changes [6-87](#)
 play rate changes [6-79](#)
 save clip finishes [6-81](#)
 seek time changes [6-83](#)
 start time changes [6-85](#)
 stop time changes [6-89](#)
 summary [6-75](#)
camera control [2-5](#)
changing view rectangle size [6-71](#)
clip

- creating [6-48](#)
 - saving in WMV format [6-49](#)
 - starting [5-18](#)
 - stopping [5-18](#)
- close [6-16](#)
- closing stream [6-16](#)
- codec subtype [6-41](#)
- color, transparent
 - getting [6-61](#)
 - setting [6-60](#)
- controlling
 - video operations [6-2](#)
 - VMR display [6-57](#)
- createCiscoVideoArchive [6-51](#)
- createSyncId [6-24](#)
- creating
 - archive [6-51](#)
 - archive clip [4-22](#)
 - clips [6-48](#)
 - snapshots [6-48](#)
 - sync ID [6-24](#)
- current time in UTC format [6-31](#)

D

- deltaMove [6-72](#)
- DIB snapshot [6-55](#)
- DIB snapshot, properly formatted [6-54](#)
- disabling event [5-10](#)
- display
 - height [6-45](#)
 - width [6-44](#)

E

- enabling event [5-9](#)
- error text description [6-38](#)
- event

- clip
 - starting [5-18](#)
 - stopping [5-18](#)
- configuration
 - motion [5-21](#)
 - single alarm [5-22](#)
 - soft trigger [5-22](#)
- disabling [5-10](#)
- enabling [5-9](#)
- handling
 - motion [5-21](#)
 - single alarm [5-22](#)
 - soft trigger [5-22](#)
- information [5-19](#)
- removing [5-11](#)
- setup [5-2](#)
- triggering [5-13](#)
- event commands
 - overview [1-2](#)
 - summary [5-1](#)

F

- framerate or bitrate, proxy [3-16](#)
- frame size, proxy [3-23](#)

G

- getAlpha [6-59](#)
- getBaseRectColor [6-63](#)
- getCiscoHD [6-26](#)
- getContentType [6-34](#)
- getCurrentSource [6-35](#)
- getDisplayHeight [6-45](#)
- getDisplayWidth [6-44](#)
- getErrorText [6-38](#)
- getPlayrateEx [6-37](#)
- getProfiles [6-39](#)

- getProfilesSSV [6-40](#)
- getRecordrateEx [6-36](#)
- getSnapshotDIB [6-54](#)
- getSnapshotWin32DIB [6-55](#)
- getState [6-33](#)
- getStreamCodecSubtype [6-41](#)
- getting
 - archive
 - details [4-15](#)
 - mediatype [4-14](#)
 - monitoring detail [4-20](#)
 - monitoring summary [4-21](#)
 - play rate [6-37](#)
 - array of supported WMV profiles [6-40](#)
 - AxClient version number [6-27](#)
 - base rectangle color [6-63](#)
 - codec subtype [6-41](#)
 - current time in UTC format [6-31](#)
 - DIB snapshot [6-55](#)
 - display
 - height [6-45](#)
 - width [6-44](#)
 - error text description [6-38](#)
 - event information [5-19](#)
 - HD camera indication [6-26](#)
 - list of supported WMV profiles [6-39](#)
 - media type [6-34](#)
 - original start time in UTC format [6-32](#)
 - player state [6-33](#)
 - properly formatted DIB snapshot [6-54](#)
 - proxy
 - device [3-22](#)
 - framerate or bitrate [3-16](#)
 - frame size [3-23](#)
 - media type [3-15](#)
 - model [3-20](#)
 - quality, MJPEG [3-17](#)
 - source [3-14](#)
 - status [3-21](#)
 - video height [3-19](#)
 - video width [3-18](#)
 - record rate [6-36](#)
 - RTSP stream from VSMS [2-13](#)
 - SDP information for video stream [2-3](#)
 - seek time in UTC format [6-28](#)
 - source URL [6-35](#)
 - start time in UTC format [6-29](#)
 - stop time in UTC format [6-30](#)
 - stream height [6-43](#)
 - stream width [6-42](#)
 - transparent color [6-61](#)
 - view rectangle
 - x coordinate [6-46](#)
 - y coordinate [6-47](#)
 - VMR display mode [6-68](#)
 - VMR transparency level [6-59](#)
 - VSMS version [2-2](#)
 - zoom rectangle
 - color [6-65](#)
 - factor [6-70](#)
- getTransparent [6-61](#)
- getUTCCurrentTime [6-31](#)
- getUTCOriginalStartTime [6-32](#)
- getUTCSeekTime [6-28](#)
- getUTCStartTime [6-29](#)
- getUTCStopTime [6-30](#)
- getVersion [6-27](#)
- getVideoHeight [6-43](#)
- getVideoWidth [6-42](#)
- getVmrDisplayMode [6-68](#)
- getX [6-46](#)
- getY [6-47](#)
- getZoomFactor [6-70](#)
- getZoomRectColor [6-65](#)

H

- HD camera indication [6-26](#)

height, stream [6-43](#)
 hiding time stamp [6-21](#)

I

invoking callback when
 archive starts [6-77](#)
 archive stops [6-76](#)
 playback starts [6-84](#)
 playback state changes [6-87](#)
 play rate changes [6-79](#)
 save clip finishes [6-81](#)
 seek time changes [6-83](#)
 start time changes [6-85](#)
 stop time changes [6-89](#)

J

JavaScript, programming notes [1-3](#)

L

listing
 all archives [4-12](#)
 all proxies [3-12](#)
 running archives [4-13](#)
 looping archive segment [6-18](#)

M

media devices, supported [B-1](#)
 media type [6-34](#)
 MJPEG proxy quality [3-17](#)
 motion event configuration [5-21](#)
 move [6-71](#)
 moving zoom rectangle [6-72](#)
 mtStartStream [6-3](#)
 mtStartStreamWait [6-7](#)

mtStreamStarting [6-5](#)

O

obtaining
 AxClient information [6-25](#)
 video stream information [6-25](#)
 original start time in UTC format [6-32](#)

P

pause [6-13](#)
 pausing
 video stream [6-13](#)
 video stream after start [6-7](#)
 player state [6-33](#)
 playForward [6-8](#)
 playing
 archive segment loop [6-18](#)
 paused video stream [6-14](#)
 video stream in forward direction [6-8](#)
 video stream in reverse direction [6-10](#)
 playResume [6-14](#)
 playRewind [6-10](#)
 programming notes
 C# [1-3](#)
 JavaScript [1-3](#)
 proxy
 device [3-22](#)
 framerate or bitrate [3-16](#)
 frame size [3-23](#)
 listing all [3-12](#)
 media type [3-15](#)
 model [3-20](#)
 quality, MJPEG [3-17](#)
 source [3-14](#)
 starting [3-2](#)
 status [3-21](#)

- stopping [3-10](#)
- updating [3-6](#)
- video height [3-19](#)
- video width [3-18](#)

proxy commands

- overview [1-2](#)
- summary [3-1](#)

R

- record rate [6-36](#)
- removeFromSync [6-23](#)
- removing
 - archive [4-11](#)
 - event [5-11](#)
 - playback from sync control [6-23](#)
- renaming archive [4-8](#)
- repeatUTCsegment [6-18](#)
- resizeVideoWindow [6-73](#)
- resizing video window [6-73](#)
- returning status of started video stream [6-5](#)
- RTSP stream, getting from VSMS [2-13](#)

S

- saveInPortableFormat [6-49](#)
- saving clip in WMV format [6-49](#)
- seek archive to
 - archive percentage [6-20](#)
 - UTC time [6-19](#)
- seek time in UTC format [6-28](#)
- seekToPercentage [6-20](#)
- seekToUTCTime [6-19](#)
- sending
 - event notification [5-13](#)
 - event notification from trigger device to VSMS [5-13](#)
- setAlpha [6-58](#)
- setBaseRectColor [6-62](#)

- setOnEndOfStream [6-76](#)
- setOnMtStartStreamDone [6-77](#)
- setOnPlayrateChanged [6-79](#)
- setOnSaveResponse [6-81](#)
- setOnSeekTimeChanged [6-83](#)
- setOnStartOfStream [6-84](#)
- setOnStartTimeChanged [6-85](#)
- setOnStateChanged [6-87](#)
- setOnStopTimeChanged [6-89](#)
- setPlayrateEx [6-17](#)
- setTimeStampRect [6-66](#)
- setting
 - archive play rate [6-17](#)
 - base rectangle color [6-62](#)
 - time stamp rectangle [6-66](#)
 - transparent color [6-60](#)
 - VMR display mode [6-67](#)
 - VMR transparency level [6-58](#)
 - zoom rectangle
 - color [6-64](#)
 - factor [6-69](#)
- setting up
 - callback [6-75](#)
 - event [5-2](#)
- setTransparent [6-60](#)
- setVmrDisplayMode [6-67](#)
- setZoomFactor [6-69](#)
- setZoomRectColor [6-64](#)
- showing time stamp [6-21](#)
- showTimestamp [6-21](#)
- single alarm event configuration [5-22](#)
- snapshot
 - creating [6-48](#)
 - from video frame [6-53](#)
- soft trigger event configuration [5-22](#)
- source URL [6-35](#)
- starting
 - archive [4-2](#)
 - event clip [5-18](#)

- proxy [3-2](#)
- video stream [6-3](#)
- start time in UTC format [6-29](#)
- status, proxy [3-21](#)
- stepForward [6-11](#)
- stepping archive video
 - one frame in playback direction [6-11](#)
 - one frame opposite playback direction [6-12](#)
- stepRewind [6-12](#)
- stopping
 - archive [4-10](#)
 - event clip [5-18](#)
 - proxy [3-10](#)
 - streaming of live or archive video [6-15](#)
- stop time in UTC format [6-30](#)
- stream
 - closing [6-16](#)
 - height [6-43](#)
 - SDP information [2-3](#)
 - width [6-42](#)
- subtype, codec [6-41](#)
- supported media devices [B-1](#)
- sync ID [6-24](#)

T

- time stamp, showing or hiding [6-21](#)
- time stamp rectangle, setting [6-66](#)
- transparency level, VMR
 - getting [6-59](#)
 - setting [6-58](#)
- transparent color
 - getting [6-61](#)
- transparent color
 - setting [6-60](#)
- triggering VSMS event [5-13](#)

U

- updating
 - archive [4-5](#)
 - proxy [3-6](#)
- URL-based media server API commands
 - archive commands [1-2](#)
 - event commands [1-2](#)
 - proxy commands [1-2](#)
 - VSMS commands [1-1](#)
- URL of loaded source [6-35](#)
- UTC
 - current time [6-31](#)
 - original start time [6-32](#)
 - playing an archive segment loop [6-18](#)
 - seek archive to [6-19](#)
 - seek time [6-28](#)
 - start time [6-29](#)
 - stop time [6-30](#)

V

- version number, AxClient [6-27](#)
- video
 - height [3-19](#)
 - operations [6-2](#)
 - width [3-18](#)
- video stream
 - information [6-25](#)
 - pausing [6-13](#)
 - pausing after start [6-7](#)
 - playing
 - forward direction [6-8](#)
 - reverse direction [6-10](#)
 - resume playing after pause [6-14](#)
 - returning start status [6-5](#)
 - SDP information [2-3](#)
 - starting [6-3](#)
- viewing JPEG frames [3-11](#)

view rectangle, changing size [6-71](#)

VMR

display [6-57](#)

turning on with C# [A-1](#)

VMR display mode

getting [6-68](#)

setting [6-67](#)

VMR transparency level

getting [6-59](#)

setting [6-58](#)

VSMS commands

overview [1-1](#)

summary [2-1](#)

VSMS version [2-2](#)

setting [6-69](#)

moving [6-72](#)

W

width, stream [6-42](#)

WMV profiles

array of supported [6-40](#)

list of supported [6-39](#)

X

x coordinate, view rectangle [6-46](#)

Y

y coordinate, view rectangle [6-47](#)

Z

zoom rectangle

color

getting [6-65](#)

setting [6-64](#)

factor

getting [6-70](#)

