

Lightspeed Live Capture Web API Reference



Vantage 7.1 UP1
Live Capture V2.4
Live Capture ComponentPac 7.1.5

June 2018

Copyrights and Trademark Notices

Copyright © 2018 by Telestream, LLC. All rights reserved worldwide. No part of this publication may be reproduced, transmitted, transcribed, altered, or translated into any languages without the written permission of Telestream. Information and specifications in this document are subject to change without notice and do not represent a commitment on the part of Telestream.

Telestream. Telestream, CaptionMaker, Episode, Flip4Mac, FlipFactory, Flip Player, Lightspeed, ScreenFlow, Switch, Vantage, Wirecast, Gameshow, GraphicsFactory, MetaFlip, and Split-and-Stitch are registered trademarks and MacCaption, e-Captioning, Pipeline, Post Producer, Tempo, TrafficManager, VidChecker, and VOD Producer are trademarks of Telestream, LLC. All other trademarks are the property of their respective owners.

Adobe. Adobe® HTTP Dynamic Streaming Copyright © 2014 Adobe Systems. All rights reserved.

Apple. QuickTime, MacOS X, and Safari are trademarks of Apple, Inc. Bonjour, the Bonjour logo, and the Bonjour symbol are trademarks of Apple, Inc.

Avid. Portions of this product Copyright 2012 Avid Technology, Inc.

Dolby. Dolby and the double-D symbol are registered trademarks of Dolby Laboratories.

Fraunhofer IIS and Thomson Multimedia. MPEG Layer-3 audio coding technology licensed from Fraunhofer IIS and Thomson Multimedia.

Google. VP6 and VP8 Copyright Google Inc. 2014 All rights Reserved.

MainConcept. MainConcept is a registered trademark of MainConcept LLC and MainConcept AG. Copyright 2004 MainConcept Multimedia Technologies.

Manzanita. Manzanita is a registered trademark of Manzanita Systems, Inc.

MCW. HEVC Decoding software licensed from MCW.

MediaInfo. Copyright © 2002-2013 MediaArea.net SARL. All rights reserved.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Microsoft. Microsoft, Windows NT|2000|XP|XP Professional|Server 2003|Server 2008 |Server 2012|Server 2016, Windows 7, Windows 8, Media Player, Media Encoder, .Net,

Internet Explorer, SQL Server 2005|2008|2012, and Windows Media Technologies are trademarks of Microsoft Corporation.

SharpSSH2. SharpSSH2 Copyright (c) 2008, Ryan Faircloth. All rights reserved. Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.

Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

Neither the name of Diversified Sales and Service, Inc. nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Telerik. RadControls for ASP.NET AJAX copyright Telerik All rights reserved.

VoiceAge. This product is manufactured by Telestream under license from VoiceAge Corporation.

x264 LLC. The product is manufactured by Telestream under license from x264 LLC.

Xceed. The Software is Copyright ©1994-2012 Xceed Software Inc., all rights reserved.

ZLIB. Copyright (C) 1995-2013 Jean-loup Gailly and Mark Adler.



Authorized Developer
Avid®



Other brands, product names, and company names are trademarks of their respective holders, and are used for identification purpose only.

MPEG Disclaimers

MPEGLA MPEG2 Patent

ANY USE OF THIS PRODUCT IN ANY MANNER OTHER THAN PERSONAL USE THAT COMPLIES WITH THE MPEG-2 STANDARD FOR ENCODING VIDEO INFORMATION FOR PACKAGED MEDIA IS EXPRESSLY PROHIBITED WITHOUT A LICENSE UNDER APPLICABLE PATENTS IN THE MPEG-2 PATENT PORTFOLIO, WHICH LICENSE IS AVAILABLE FROM MPEG LA, LLC, 4600 S. Ulster Street, Suite 400, Denver, Colorado 80237 U.S.A.

MPEGLA MPEG4 VISUAL

THIS PRODUCT IS LICENSED UNDER THE MPEG-4 VISUAL PATENT PORTFOLIO LICENSE FOR THE PERSONAL AND NON-COMMERCIAL USE OF A CONSUMER FOR (i) ENCODING VIDEO IN COMPLIANCE WITH THE MPEG-4 VISUAL STANDARD ("MPEG-4 VIDEO") AND/OR (ii) DECODING MPEG-4 VIDEO THAT WAS ENCODED BY A CONSUMER ENGAGED IN A PERSONAL AND NON-COMMERCIAL ACTIVITY AND/OR WAS OBTAINED FROM A VIDEO PROVIDER LICENSE IS GRANTED OR SHALL BE IMPLIED FOR ANY OTHER USE. ADDITIONAL INFORMATION INCLUDING THAT RELATING TO PROMOTIONAL, INTERNAL AND COMMERCIAL USES AND LICENSING MAY BE OBTAINED FROM MPEG LA, LLC. SEE [HTTP://WWW.MPEGLA.COM](http://www.mpegla.com).

MPEGLA AVC

THIS PRODUCT IS LICENSED UNDER THE AVC PATENT PORTFOLIO LICENSE FOR THE PERSONAL USE OF A CONSUMER OR OTHER USES IN WHICH IT DOES NOT RECEIVE REMUNERATION TO (i) ENCODE VIDEO IN COMPLIANCE WITH THE AVC STANDARD ("AVC VIDEO") AND/OR (ii) DECODE AVC VIDEO THAT WAS ENCODED BY A CONSUMER ENGAGED IN A PERSONAL ACTIVITY AND/OR WAS OBTAINED FROM A VIDEO PROVIDER LICENSED TO PROVIDE AVC VIDEO. NO LICENSE IS GRANTED OR SHALL BE IMPLIED FOR ANY OTHER USE. ADDITIONAL INFORMATION MAY BE OBTAINED FROM MPEG LA, L.L.C. SEE [HTTP://WWW.MPEGLA.COM](http://www.mpegla.com).

MPEG4 SYSTEMS

THIS PRODUCT IS LICENSED UNDER THE MPEG-4 SYSTEMS PATENT PORTFOLIO LICENSE FOR ENCODING IN COMPLIANCE WITH THE MPEG-4 SYSTEMS STANDARD, EXCEPT THAT AN ADDITIONAL LICENSE AND PAYMENT OF ROYALTIES ARE NECESSARY FOR ENCODING IN CONNECTION WITH (i) DATA STORED OR REPLICATED IN PHYSICAL MEDIA WHICH IS PAID FOR ON A TITLE BY TITLE BASIS AND/OR (ii) DATA WHICH IS PAID FOR ON A TITLE BY TITLE BASIS AND IS TRANSMITTED TO AN END USER FOR PERMANENT STORAGE AND/OR USE. SUCH ADDITIONAL LICENSE MAY BE OBTAINED FROM MPEG LA, LLC. SEE [HTTP://WWW.MPEGLA.COM](http://www.mpegla.com) FOR ADDITIONAL DETAILS.

Limited Warranty and Disclaimers

Telestream, LLC (the Company) warrants to the original registered end user that the product will perform as stated below for a period of one (1) year from the date of shipment from factory:

Hardware and Media—The Product hardware components, if any, including equipment supplied but not manufactured by the Company but NOT including any third party equipment that has been substituted by the Distributor for such equipment (the “Hardware”), will be free from defects in materials and workmanship under normal operating conditions and use.

Warranty Remedies

Your sole remedies under this limited warranty are as follows:

Hardware and Media—The Company will either repair or replace (at its option) any defective Hardware component or part, or Software Media, with new or like new Hardware components or Software Media. Components may not be necessarily the same, but will be of equivalent operation and quality.

Software Updates

Except as may be provided in a separate agreement between Telestream and You, if any, Telestream is under no obligation to maintain or support the Software and Telestream has no obligation to furnish you with any further assistance, technical support, documentation, software, update, upgrades, or information of any nature or kind.

Restrictions and Conditions of Limited Warranty

This Limited Warranty will be void and of no force and effect if (i) Product Hardware or Software Media, or any part thereof, is damaged due to abuse, misuse, alteration, neglect, or shipping, or as a result of service or modification by a party other than the Company, or (ii) Software is modified without the written consent of the Company.

Limitations of Warranties

THE EXPRESS WARRANTIES SET FORTH IN THIS AGREEMENT ARE IN LIEU OF ALL OTHER WARRANTIES, EXPRESS OR IMPLIED, INCLUDING, WITHOUT LIMITATION, ANY WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. No oral or written information or advice given by the Company, its distributors, dealers or agents, shall increase the scope of this Limited Warranty or create any new warranties.

Geographical Limitation of Warranty—This limited warranty is valid only within the country in which the Product is purchased/licensed.

Limitations on Remedies—YOUR EXCLUSIVE REMEDIES, AND THE ENTIRE LIABILITY OF TELESTREAM, LLC WITH RESPECT TO THE PRODUCT, SHALL BE AS STATED IN THIS LIMITED WARRANTY. Your sole and exclusive remedy for any and all breaches of any

Limited Warranty by the Company shall be the recovery of reasonable damages which, in the aggregate, shall not exceed the total amount of the combined license fee and purchase price paid by you for the Product.

Damages

TELESTREAM, LLC SHALL NOT BE LIABLE TO YOU FOR ANY DAMAGES, INCLUDING ANY LOST PROFITS, LOST SAVINGS, OR OTHER INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF YOUR USE OR INABILITY TO USE THE PRODUCT, OR THE BREACH OF ANY EXPRESS OR IMPLIED WARRANTY, EVEN IF THE COMPANY HAS BEEN ADVISED OF THE POSSIBILITY OF THOSE DAMAGES, OR ANY REMEDY PROVIDED FAILS OF ITS ESSENTIAL PURPOSE.

Further information regarding this limited warranty may be obtained by writing:

Telestream, LLC
848 Gold Flat Road
Nevada City, CA 95959 USA

You can call Telestream via telephone at (530) 470-1300.

Part number: 259440

Date: June 2018

Contents

Overview	17
Lightspeed Live Capture Web API	18
Lightspeed Live Stream Web API	19
Lightspeed Live Source Web API	19
Operation & Response Formats	20
Operation Keyword Terms	21
Use of GUIDs/UUIDs in Operations	22
Use of Timecodes in Operations	23
Response Formats	24
Capture Operation Response Formats	24
Stream and Source Operation Response Formats	24
Reserved Characters in Value Strings	25
Stream Operations	27
Introduction	28
Limits of the API	28
Using Live Stream Groups via the API	28
Ports for Live Stream Server Access	28
Using Shared vs. Dedicated Components	28
Live Stream Component Hierarchy	29
Establishing Authorization for Stream Operations	30
GetVerificationCode	31
Operation Sequence	31
Required Parameters	31
Results	31

Example	31
Typical Response	31
GetToken	32
Operation Sequence	32
Required Parameters	32
Results	32
Example	32
Typical Response	33
Obtaining Help for Stream Operations	34
Displaying a List of Live Stream Services	34
Displaying the Operations of a Live Stream Service	34
Displaying Operation Details	34
System Operations	36
GetMachines	37
Operation Sequence	37
Results	37
Example	37
Typical Response	37
GetServiceEndpoints	38
Operation Sequence	38
Results	38
Example	38
Typical Response	38
GetSystemSettings	39
Results	39
Example	39
Typical Response	39
ImportSystemSettings	40
URL Format	40
Request Body Format	40
Operation Sequence	40
Results	40
ClearAllSettings	41
URL Format	41
Request Body	41
Operation Sequence	41
Results	41
Sources Operations	42
AddRtmpSource	43
URL Format	43
Request Body Format	43
Operation Sequence	43
Request Parameters	43
Results	43
Example	43
Typical Response	44

AddTransportStreamSource	45
URL Format	45
Request Body Format	45
Operation Sequence	45
Parameters	45
Results	46
Example	46
Typical Response	46
GetSources	47
Operation Sequence	47
Required Parameters	47
Results	47
Example	47
Typical Response	47
GetSource	49
Operation Sequence	49
Required Parameters	49
Results	49
Example	49
Typical Response	49
GetSourceTracks	51
Operation Sequence	51
Required Parameters	51
Results	51
Example	51
Typical Response	51
GetSourceTrack	52
Operation Sequence	52
Required Parameters	52
Results	52
Example	52
Typical Response	52
GetTextTracks	54
Operation Sequence	54
Required Parameters	54
Results	54
Example	54
Typical Response	54
GetTextTrack	55
Operation Sequence	55
Required Parameters	55
Results	55
Example	55
Typical Response	55
GetSourceThumbnail	56
Operation Sequence	56
Required Parameters	56

Results	56
Example	56
Programs Operations	57
GetPrograms	58
Operation Sequence	58
Results	58
Example	58
Typical Response	58
GetProgram	59
Operation Sequence	59
Required Parameters	59
Results	59
Example	59
Typical Response	59
GetRenditions	60
Operation Sequence	60
Required Parameters	60
Results	60
Example	60
Typical Response	60
GetRendition	61
Operation Sequence	61
Required Parameters	61
Results	61
Example	61
Typical Response	61
GetSegments	63
Operation Sequence	63
Required Parameters	63
Results	63
Example	63
Typical Response	63
GetSegment	64
Operation Sequence	64
Required Parameters	64
Results	64
Example	64
Typical Response	64
GetMaterials	66
Operation Sequence	66
Required Parameters	66
Results	66
Example	66
Typical Response	66
GetMaterial	68
Operation Sequence	68
Required Parameters	68

Results	68
Example	68
Typical Response	68
Encoders Operations	70
GetEncoders	71
Operation Sequence	71
Results	71
Example	71
Typical Response	71
GetEncoder	72
Operation Sequence	72
Required Parameters	72
Results	72
Example	72
Typical Response	72
GetStreams	74
Operation Sequence	74
Required Parameters	74
Results	74
Example	74
Typical Response	74
GetStream	75
Operation Sequence	75
Required Parameters	75
Results	75
Example	75
Typical Response	75
Packages Operations	77
GetPackages	78
Operation Sequence	78
Results	78
Example	78
Typical Response	78
GetPackage	80
Operation Sequence	80
Required Parameters	80
Results	80
Example	80
Typical Response	80
GetVariants	82
Operation Sequence	82
Required Parameters	82
Results	82
Example	82
Typical Response	82
GetVariant	83
Operation Sequence	83

Required Parameters	83
Results	83
Example	83
Typical Response	83
Channels Operations	84
GetChannels	86
Operation Sequence	86
Results	86
Example	86
Typical Response	86
GetChannel	87
Operation Sequence	87
Required Parameters	87
Results	87
Example	87
Typical Response	87
GetChannelOutputLocations	90
Operation Sequence	90
Required Parameters	90
Results	90
Example	90
Typical Response	90
SetVariantThumbnailSize	92
URL Format	92
Request Body Format	92
Operation Sequence	92
Required Parameters	92
Results	92
Example	93
Typical Response	93
GetChannelThumbnail	94
Operation Sequence	94
Required Parameters	94
Results	94
Example	95
StartChannel	96
URL Format	96
Request Body Format	96
Operation Sequence	96
Required Parameters	96
Results	96
Example	96
Typical Response	96
StopChannel	97
URL Format	97
Request Body Format	97
Operation Sequence	97

Required Parameters	97
Results	97
Example	97
Typical Response	97
GetSourcePlaceholdersForChannel	98
Operation Sequence	98
Required Parameter	98
Results	98
Example	98
Typical Response	98
AssignSourcesForChannel	99
URL Format	99
Request Body Format	99
Operation Sequence	99
Required Parameters	99
Results	99
Example	99
Typical Response	100
GetCalendarEvents	101
Operation Sequence	101
Required Parameters	101
Results	101
Example	101
Typical Response	101
GetCalendarEvent	102
Operation Sequence	102
Required Parameters	102
Results	102
Example	102
Typical Response	102
AddCalendarEvent	104
URL Format	104
Request Body Format	104
Operation Sequence	104
Parameters	104
Results	104
Example	105
Typical Response	105
DeleteCalendarEvent	106
URL Format	106
Request Body Format	106
Operation Sequence	106
Required Parameters	106
Results	106
Example	106
Typical Response	107
GetActiveSegment	108

Operation Sequence	108
Required Parameters	108
Results	108
Example	108
Typical Response	108
SetActiveSegment	109
URL Format	109
Request Body Format	109
Operation Sequence	109
Required Parameters	109
Results	110
Example	110
Typical Response	110
SetActiveSegmentAtTime	111
URL Format	111
Request Body Format	111
Operation Sequence	111
Required Parameters	111
Results	112
Example	112
Typical Response	112
GetMachineStatistics	113
Operation Sequence	113
Required Parameters	113
Results	113
Example	113
Typical Response	113
GetChannelStatistics	114
Operation Sequence	114
Required Parameters	114
Results	114
Example	114
Typical Response	114
SetHlsOutputPackageName	115
URL Format	115
Request Body Format	115
Operation Sequence	115
Parameters	115
Results	115
Example	116
Typical Response	116
Social Media Platform Channel Management Operations	117
GetNewFacebookVerificationDeviceCode	118
Results	118
Example	118
Typical Response	118
AddFacebookChannel	119

URL Format	119
Request Body Format	119
Operation Sequence	119
Required Parameters	119
Results	120
Example	120
Typical Response	120
ConfigureFacebookChannel	122
URL Format	122
Request Body Format	122
Operation Sequence	122
Parameters	122
Results	123
Example	123
Typical Response	123
GetNewPeriscopeVerificationUserCode	125
Results	125
Example	125
Typical Response	125
AddPeriscopeChannel	126
URL Format	126
Request Body Format	126
Operation Sequence	126
Required Parameters	126
Results	127
Example	127
Typical Response	127
ConfigurePeriscopeChannel	129
URL Format	129
Request Body Format	129
Operation Sequence	129
Parameters	129
Results	130
Example	130
Typical Response	130
AddRtmpToAkamaiChannel	132
URL Format	132
Request Body Format	132
Operation Sequence	132
Parameters	132
Results	133
Example	133
Typical Response	133
AddRTMPtoCustomURLChannel	135
URL Format	135
Request Body Format	135
Operation Sequence	135

- Parameters 135
- Results 136
- Example 136
- Typical Response 136

AddRTMPtoYouTubeChannel 138

- URL Format 138
- Request Body Format 138
- Operation Sequence 138
- Parameters 138
- Results 138
- Example 138
- Typical Response 139

Source Operations 141

Introduction 142

- Port for Lightspeed Live/Capture Server Access 142
- Using Live Stream Groups via the API 142
- Using Shared vs. Dedicated Components 142
- Live Source Component Hierarchy 142

Obtaining Help for Source Operations 143

- Displaying the Operations of a Live Source Service 143
- Displaying Operation Details 143

GetMachines 145

- Operation Sequence 145
- Results 145
- Example 145
- Typical Response 145

GetSources 146

- Operation Sequence 146
- Required Parameter 146
- Results 146
- Example 146
- Typical Response 146

InsertID3Frame 147

- Operation Sequence 147
- Parameters 147
- Required Post Body 147
- Results 147
- Example 148

InsertScte35Message 149

- Operation Sequence 149
- Parameters 149
- SCTE-35 Commands 150
- Results 150
- Example 150

Overview

The three Lightspeed Live web APIs—Capture, Stream, and Source —enable you to monitor and control your Lightspeed Live system within a broader, web services-based system or create your own customized monitoring system.

You can also create a web services-based system to control streaming and capture beyond the functionality or capability of the general-purpose Vantage Capture and Stream web applications, to meet your organization's requirements.

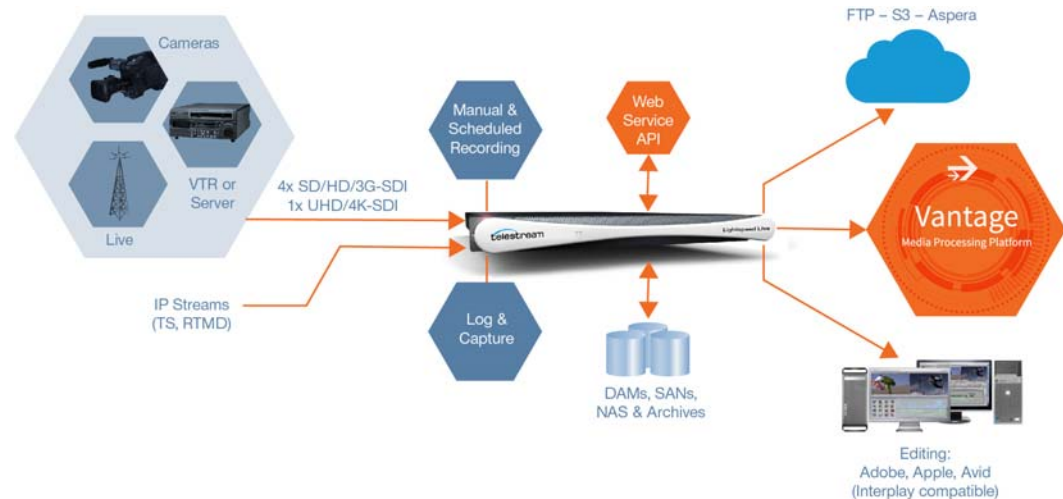
- [Lightspeed Live Capture Web API](#)
- [Lightspeed Live Stream Web API](#)
- [Lightspeed Live Source Web API](#)
- [Operation & Response Formats](#)
- [Reserved Characters in Value Strings](#)

Note: This reference assumes that the programming environment being used by the developer includes a library that abstracts the process of operation submission and responses through the HTTP protocol.

If your environment does not include a library to perform this abstraction then you will have to directly format your operations to adhere to the HTTP protocol. See [Hypertext Transfer Protocol](#) for details.

Lightspeed Live Capture Web API

Lightspeed Live Capture enables recording of live streaming media in a Vantage workflow to be controlled manually via the Lightspeed Live Capture web application or through the HTTP web service [Capture Operations](#) described in this reference. The Capture web API is implemented in the Lightspeed Live Capture web service.



Lightspeed Live Capture workflows can be configured to publish a web service that enables media clips to be recorded using this web services operation set.

In order to respond to Capture API operations, the Capture action must be configured with a Web Service trigger, and the specified port (default: 17000) must be utilized as the target port in each operation. You can optionally enable manual control (control via the Web application) and you can also enable logging.

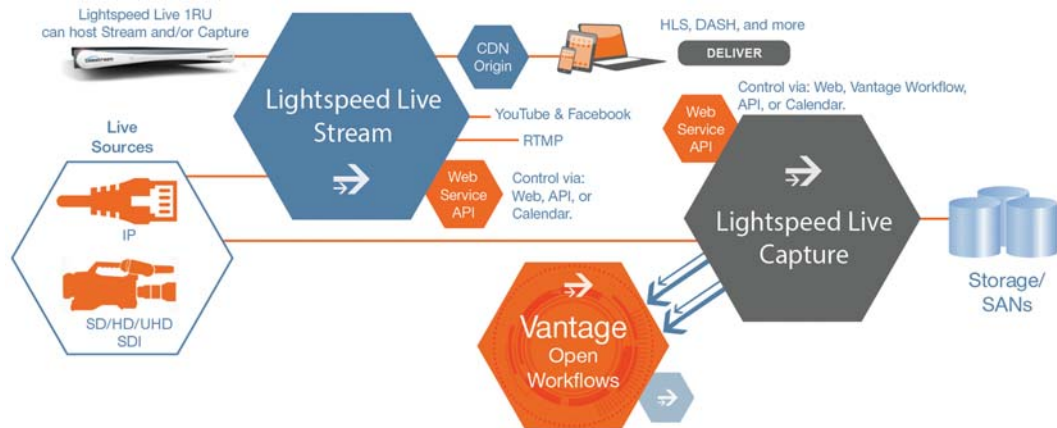
When a Capture workflow is activated in Vantage via Workflow Designer, the web service begins listening for requests on the specified port. (The user will be warned if the port is in use by another Capture workflow or other service on the host computer.)

Your client program can control any given SDI input device on the Lightspeed server (camera, deck, etc.) by communicating with a target Vantage workflow—which in turn is configured for a specific SDI input and a specific web service port. Your program may be designed to target a specific SDI input device by communicating on a specific port (thus, a specific workflow and SDI device) on the domain, or it may be more broadly-designed (using the Vantage SDK) to obtain a list of currently-running workflows, and present those to the user for selection dynamically.

Note: Depending on the requirements of your Capture application, you may also integrate the Vantage SDK in your Capture program along with the Lightspeed Live Capture web service. Integrating the Vantage SDK enables you to programmatically control the target workflow as well as the capture operation; starting and stopping the workflow, testing its status, querying job results, and submitting jobs, for example.

Lightspeed Live Stream Web API

Lightspeed Live Stream supports adaptive bit rate encoding for SD, HD and UHD sources into AVC and HEVC. Input support is available for SDI as well as IP sources, offering future-proof operation as delivery mechanisms change. Output can be delivered via RTMP or as HTTP Live Streaming (HLS) and MPEG DASH packages.



Lightspeed Live Stream enables you to transcode source files in real-time, streaming them via the Lightspeed Live Stream web application or through the HTTP web service [Stream Operations](#) described in this reference.

Use of the Stream API requires authentication for security purpose. See [Establishing Authorization for Stream Operations](#) for details on how authentication is implemented and how to use it in your Live Stream applications.

The Live Stream web API is implemented in the Lightspeed Live Stream web service.

Lightspeed Live Source Web API

The Lightspeed Live Source Web API supports the insertion of SCTE-35 and ID3 tags. Use of the Source API to insert tags can be performed during capture or stream sessions. The Source API is implemented in the Source web service.

The Lightspeed Source web API is described in [Source Operations](#).

Operation & Response Formats

The Lightspeed Live APIs are a RESTful implementation. Most Lightspeed Live web service operations are invoked using an HTTP GET request in the following form:

```
http://<host>:<port>/<API Category>/<operation>  
[?<parameter>=<value>[&<parameter>=<value>]]
```

Operations that alter operational data in the Live Stream server are POST operations; such as *AddCalendarEvent*, for example. Live Stream POST operations use a JSON-formatted request body for transferring parameters to the Live Stream web service.

Lightspeed Live systems respond to operations with an HTTP status line (for example: *200 OK* or *404 Not Found*), HTTP headers, and either XML or JSON-formatted responses in the body. Responses vary, based on the operation and parameters.

Stream API and Source API responses are primarily in JSON format (although a few operations return XML, ZIP files, thumbnails, etc.)

The Capture API includes Post commands as well. Responses are in XML format.

Topics

- [Operation Keyword Terms](#)
- [Use of GUIDs/UUIDs in Operations](#)
- [Use of Timecodes in Operations](#)
- [Response Formats](#)

Operation Keyword Terms

Keyword terms in each operation are shown in this reference surrounded by less than and greater than symbols (<>); they are placeholders in the operation's description, to be replaced by values you specify.

Note: Operation names are not case-sensitive. For example, you can specify *Encoders/GetStreams* or you can specify *encoders/getstreams* to execute the *GetStreams* operation.

When an operation has required and/or optional parameters, they are displayed as name/value pairs in the query portion (?) of the request. Additional parameters are separated by an ampersand (&). Parameters in brackets ([]) are optional:

```
http://<host>:<port>/record/start  
[?<parameter>=<value>[&<parameter>=<value>]]
```

Here are the keyword terms you'll encounter in parameters:

Term	Description
host	The Windows domain name or the IP address of the Lightspeed Live Capture or Stream server you are targeting. For example: <i>localhost</i> <i>LightspeedServer</i> <i>192.168.1.23</i> .
port	The TCP port number assigned to the web service. When using the Live Capture API, the port number used in the operation identifies the Vantage Capture workflow whose jobs you are controlling—thus, when you have multiple Capture workflows, each workflow should be configured with a different port if they are going to run concurrently. The port (default: 17000) is displayed and configured in the Capture action inspector of the target Vantage workflow. (See the Lightspeed Live Guide or man page for the Capture action.) The following API server ports are selectable in the Live Stream web app settings. <ul style="list-style-type: none"> • Live API server port (default: 18000) • Source API server port (default: 15000) The Authorization API server port is 16000 and can not be changed.
parameter	An optional, named parameter defined by the web service operation. Parameters are listed and described in the associated table in each operation.
value	The value for the associated parameter.

Use of GUIDs/UUIDs in Operations

A GUID (Globally Unique Identifier)—referred to as UUID in Capture operations—is used in most operations to target or identify a specific instance of a component. For example, a stream or program. The important property of a GUID is that each value is globally unique, enabling you to identify a specific target using the GUID. The value is generated by an algorithm, developed by Microsoft, which assures this uniqueness.

A GUID is a 16-byte binary data type that can be logically grouped into the following subgroups: 4byte-2byte-2byte-2byte-6byte.

The standard textual representation is {12345678-1234-1234-1234-1234567890AB}.

For example, `ad1c45b7-67fb-419d-8c5b-8ba474bd6dfd`.

Note: If the GUID you supply in an operation is not properly formed, the operation fails to execute and returns an HTML XML advising of the Request Error.

Using GUIDs in Capture Operations

When you are operating on a specific clip, the GUID is first obtained when you begin capturing the clip, via the [Start](#) operation. In the response, the GUID is returned in the UUID element.

Use the clip's GUID when you use *Modify*, *MarkIn/MarkOut*, *EditIn/EditOut*, *Stop*, and *Status* operations to identify the target clip.

Using GUIDs in Stream Operations

When you are requesting information about a specific component from the system (such as a source track), the GUID is the identifier of that component.

You obtain GUIDs for specific components in your system using the *Get* operation for the component set (without a GUID): *GetPrograms*, for example.

When you are retrieving multiple components (for example, all tracks), the GUID is the identifier of the parent component.

For example, in this hierarchy: *Machine > Source > Source Track...*

If you want to retrieve a specific Source or Source Track, you provide its GUID identifier. If you want to retrieve all of the Sources that belong to a Machine or all of the Source Tracks that belong to a Source, you supply the identifier of the Machine or Source.

Use of Timecodes in Operations

Timecodes are used as parameters and are also returned as part of responses.

Timecodes can be specified in either drop frame (for example: 01:00:10;00) or non-drop frame format (for example: 01:00:10:00) but they are treated identically (assuming you are using drop frame if the source is drop frame and non-drop frame if the source is non-drop frame).

The timecode source is determined by the configuration of the source input. The timecode may be Source, Computer Clock, Free Run, Analog LTC, or RS422, as defined in the Vantage Domain Console's Capture Inventory Capture Source settings.

Response Formats

Response formats vary by API and by operation.

Capture Operation Response Formats

When you execute a Capture operation, the Capture service executes the operation and returns an XML response. For example:

```
<Response>
  <UUID>27638f24-2bb1-4ce6-8816-5a05a5e05897</UUID>
  <PercentCompleted>0</PercentCompleted>
  <Progress>0</Progress>
  <ActionDuration>3239.9733066</ActionDuration>
  <FPS>29.97002997003</FPS>
  <Start>11:05:06;09@29.97</Start>
  <End></End>
  <MarkIn>11:05:06;09@29.97</MarkIn>
  <MarkOut>11:59:06;09@29.97</MarkOut>
  <Excluding>False</Excluding>
  <Name>SDI-2 - Web UI_LSL-PM - SDI Input 2.8</Name>
  <HorizontalResolution>1920</HorizontalResolution>
  <VerticalResolution>1080</VerticalResolution>
  <FrameRate>29.97002997003</FrameRate>
  <Channels>16</Channels>
  <State>Opened</State>
  <Access-Control-Allow-Origin>*</Access-Control-Allow-Origin>
  <EngineState>Running</EngineState>
  <EngineTime>11:02:38;01</EngineTime>
  <XMLRevision>2</XMLRevision>
</Response>
```

Stream and Source Operation Response Formats

When you execute a Stream or Source operation, the Stream | Source web service returns a JSON response. For example:

```
[
  {
    "Description":null,
    "Identifier":"89d2be4b-be21-4838-a551-522cce299fbe",
    "Name":"LL-PM-1"
  },
  {
    "Description":null,
    "Identifier":"4bd2be89-2c24-3947-a432-484bca2387fba",
    "Name":"LL-PM-1"
  },
  {
    "Description":null,
    "Identifier":"43b2ff5b-ac29-48573-c443-567cad734efa",
    "Name":"LL-PM-1"
  }
]
```


Reserved Characters in Value Strings

This list of reserved characters can cause a variety of problems in parameters—you should not use them in parameter values:

" < > # % { } | \ ^ ~ [] ` ; / ? : @ = & +

During processing, certain characters are omitted; others truncate the remainder of the string or are changed to a space character. In some circumstances, this error is displayed: "Request Error - The server encountered an error processing the request. See server logs for more details."

Capture Operations

You use the Lightspeed Live Capture web service operations to record video streams. All Capture operations are GET operations; *Start* may also be submitted as a POST, when supplying runtime variables to a Vantage capture job.

Each operation in the API is presented with a brief description, including the format of the operation, and all required and optional parameters in a table. Finally, a typical *Response* is presented using an example.

Most operations return a *Response* XML. *Get-Statistics* returns a *Statistics* XML, *Message* returns a *string*, and *Variables* returns a *Customization*.

Note: In order to respond to Capture API operations, the Capture action in the capture workflow must be configured with a Web Service trigger, with a unique, user-specified port (default: 17000; min/max 1024-65536). The port must be specified as the target port for the workflow in each operation you execute. Finally, the workflow must be activated in Vantage and it must remain activated to process jobs underway and jobs awaiting execution in the clip list. You can optionally enable manual control (via the Web application) and you can also enable logging.

- [Response Elements](#)
- [Get-Statistics](#)
- [Start](#)
- [Modify](#)
- [MarkOut | MarkIn](#)
- [EditOut | EditIn](#)
- [Message](#)
- [Stop](#)
- [Status](#)
- [Variables](#)

Response Elements

For operations that return a *Response* XML, certain elements are returned in every successful *Response*. Other elements are returned only for specific operations.

- [Standard Response Elements](#)
- [Optional Response Elements](#)

Standard Response Elements

Response Element	Description
Access-Control-Allow-Origin	For use only by Telestream.
EngineState	Keyword; the current state of the Capture service. It always reports <i>Running</i> .
EngineTime	Timecode; the current Lightspeed Live Capture server clock time. For example: 12:34:56:00. Note: <i>EngineTime</i> is only returned when at least one job is in the clip list; either queued or running.
UUID	The GUID of the clip being operated on. For example: 5b1eb65c-3018-a4cf-8134-6e1c16b378a7.
XMLRevision	Integer; the <i>Response</i> XML revision number. For example: 2.

Optional Response Elements

These elements are returned in a successful *Response*, based on the operation and the parameters utilized. Some elements are always updated in the initial *Response*. Others are updated only in the *Response* to subsequent *Status* operations.

Response Element	Description
ActionDuration	Real; the estimated duration of the Capture action, in seconds. Returned by Start Stop Modify Status operations when a GUID is specified.
Channels	Integer; the number of audio channels in the target clip.
Discontinuity	Keyword (Abort Ignore); specifies how to control capturing when timecode discontinuities are observed in the video stream. When set to <i>Ignore</i> , any timecode discontinuity will be ignored and recording will continue. When set to <i>Abort</i> , recording stops and the job is terminated when a timecode discontinuity is detected.
End	Timecode; end or anticipated end of the target clip, represented in timecode format: HH:MM:SS:FF.
Error	String; reports a detailed error during a Failed state (the <i>EngineState</i> element has the keyword Failed). For example: "Writer stalled. This is often caused by the writer not being able to keep up with the reader due to I/O limitations."
Excluding	Boolean; reports <i>True</i> if frames are being excluded via the use of MarkOut MarkIn or EditIn EditOut operations. Mark operations can only be used on TIFO files. <i>Excluding</i> is <i>False</i> upon starting an initial capturing and after an In operation. <i>Excluding</i> is <i>True</i> after an Out operation.
FPS	Real; the frame rate in FPS of the target clip.
FrameRate	Real; the time scale, divided by the duration of the target clip.
HorizontalResolution VerticalResolution	Integer; the clip's horizontal and vertical frame resolution, in pixels.
Identifier	GUID; the Vantage Capture workflow's job identifier for the target clip; used in <i>Status</i> , <i>Start</i> , and <i>Stop</i> operations. Can be used in Vantage SDK queries to obtain additional status and information about the workflow associated with the clip being captured.
MarkIn MarkOut	Timecode; timecode of beginning (inclusive) or ending (exclusive) frame of a segment of the target clip.
Name	String; the file name of the clip being recorded by the workflow, including the suffix.

Response Element	Description
Path	String; fully-qualified path to the file being recorded, including the file name and suffix.
PercentCompleted	Not implemented.
Progress	Real; elapsed time (in seconds) of the capture job. Returned by Start Stop Modify Status operations when UUID specified.
Start	Timecode; starting timecode of the target clip.
State	<p>Keyword; the state of the job for the specified clip:</p> <p><i>Closed</i>—when GUID supplied, recording of the clip is complete or the job was stopped.</p> <p><i>Failed</i>—when GUID supplied, the capture job failed. The <i>Error</i> element contains a descriptive string describing the cause of the <i>Failed</i> state.</p> <p><i>Opened</i>—when GUID supplied, the clip is present in the clip list. It may be queued waiting for the specified start time or it may be actively recording.</p> <p><i>Waiting</i>—when no GUID supplied, the clip list is empty and the Capture workflow is waiting for jobs to submit.</p> <p>When monitoring a clip in the <i>Opened</i> state, you can test the <i>Progress</i> element returned from a Status (with UUID) operation to determine if it is waiting to start or is currently capturing. If State = Opened and Progress > 0, then the clip is capturing. To determine the job's progress, calculate the percent using this formula: (Progress / ActionDuration) X 100. The <i>PercentCompleted</i> element should not be used.</p>
TransmitError	String; messages or errors returned from the Capture process.
UUID	GUID; unique identifier of clip being operated upon.

Get-Statistics

The *Get-Statistics* operation returns a *Statistics* XML for the target (by host and port number) Lightspeed Live server, with source and deck information. This operation has no parameters.

Get-Statistics has the following format:

```
http://<host>:<port>/record/get-statistics
```

Statistics Elements

* Deck-Mode, Deck-State, Deck-Status are returned only when a VTR is detected on the input channel associated with the job being captured.

Response Element	Description
Audio	Integer; the number of channels in the target clip.
Bit-Depth	Integer; bit depth of the target clip.
Captions	Keyword: No Yes; indicates the presence of captions in the clip.
CPU-Usage	Integer; the current load on the CPU, expressed as a percentage.
Deck-Mode*	Keyword; reports the VTR's Remote state: <i>Remote</i> <i>Local</i> .
Deck-State*	Keyword; reports the VTR's Transport state: <i>Stopped</i> <i>Playing</i> .
Deck-Status*	Reports DATA-0 through DATA-A STATUS DATA bytes returned by a STATUS SENSE RS-422 serial protocol command. Data bytes 0 - A (2 nibbles = one byte): 00 11 22 33 44 55 66 77 88 99 AA [checksum]. Example: 00 20 00 00 00 00 00 00 00 80 3A 74
Frame-Rate	Real; the frame rate in FPS of the target clip.
Firmware-Version	String; version of the firmware installed on the target Live Stream server.
Memory	Integer; the total memory in the server, in MB.
Process	Integer; the Program Identifier (PID) of the LiveSource process assigned to the workflow/channel. (Also displayed in the LiveSource.exe row of the Task Manager Details table in the Web application.)
Resolution	String; the frame size and type of the source. For example: <i>1920x1080i</i> . This information is also displayed in the Web application's Preview panel and in the Vantage Management Console's Capture Inventory panel Preview dialog for a given SDI input.
SDI	Integer; the video port currently being streamed.
Time-Code	Timecode; the current timecode of the video on the SDI port.

Example

Here is an example of a *Get-Statistics* operation:

`http://LS-SVR:17001/record/get-statistics`

Typical Response

Issuing a *Get-Statistics* operation returned the following *Statistics* XML:

```
<Statistics>
  <Firmware-Version>2016/12/30</Firmware-Version>
  <CPU-Usage>0</CPU-Usage>
  <Memory>7933</Memory>
  <Process>8404</Process>
  <SDI>4</SDI>
  <Resolution>1920x1080i</Resolution>
  <Frame-Rate>29.97</Frame-Rate>
  <Bit-Depth>10-Bit</Bit-Depth>
  <Audio>16</Audio>
  <Captions>No</Captions>
  <Time-Code>01:01:01;10</Time-Code>
  <Deck-Status>00200000000000000000ba74</Deck-Status>
  <Deck-State>Stopped</Deck-State>
  <Deck-Mode>Remote</Deck-Mode>
</Statistics>
```


Start

The *Start* operation may be executed as a GET or a POST operation, both of which initiates the recording of a new clip by submitting a Vantage job from the target Capture workflow. a GET operation should be used when you aren't submitting runtime variables to the Vantage job.

Note: If you are submitting runtime variables to the job, you should execute use *Start* as a POST instead. (See [Supplying Runtime Variable Values](#) for a description of submitting the variables.)

The *Start* operation initiates a Vantage job from the Capture workflow identified by the port number that the Capture action's Web service trigger is configured to monitor. No GUID is provided.

You can start capturing the video immediately or you can specify a future *start* time. You can also explicitly name the file and specify an alternate file storage path, and control how video recording is handled when discontinuous timecodes are detected.

If an *end* time is not specified the clip records the video for up to 9 hours unless stopped or disk space is exhausted. For best results, specify an *end* time to and design your application to actively monitor and control the recording time.

Note: Specifying a *start* timecode that is 60 seconds or less earlier than the current time results in a capture job starting immediately. For example, if the current timecode is 02:30:00;20 and a *start* timecode of 02:29:30;00 is specified, the job starts capturing immediately.

Conversely, specifying a *start* timecode greater than 60 seconds prior to the current time causes the capture job to be queued, to start capturing in 24 hours. For example, if the current timecode is 02:30:00;20 and the *start* timecode is 02:29:00;00, the job is queued and capturing will begin at 02:29:00;00 tomorrow, nearly 24 hours later.

The job ID GUID (in the *Identifier* element) is always returned with an all-zero GUID, indicating that Vantage has not yet updated its database—it may take several seconds (typically up to ten) to update it; obtain the job ID GUID using [Status](#).

Note: Up to 16 jobs per Capture workflow can be queued in a clip list for processing. Jobs are automatically removed from the clip list when complete or stopped. If you submit more than the maximum number of jobs permitted, an error is generated. You can use the [Status](#) operation to obtain a clip list for the target workflow, so that you can determine the number of clips currently queued for processing.

Supplying Runtime Variable Values

To supply runtime variables with a job, submit the *Start* command as a post, providing a *Customization* XML enclosed in the body of the request message, with a header field in the POST to indicate the message body's Internet media type:

Note: The POST operation must be sent with a header specifying Content-Type: *application/x-www-form-urlencoded*. Failure to specify this content-type will result in an error from the IIS server.

```
<Customization xmlns:soa="urn:telestream.net:soa:core"
xmlns="urn:telestream.net:soa:live">
  <soa:Condition identifier="<Variable GUID>" name="<Variable
Name>" type="<Variable Type>">
    <soa:Value><Variable Runtime Value></soa:Value>
  </soa:Condition>
  ...
</Customization>
```

Note: The *soa:Condition* element's *type* attribute value (for example, "string") must be in lower case or the operation will fail.

These are the elements and attributes you must supply:

- *soa:Condition* element:
 - *identifier* attribute— the target variable GUID (replacing the string <Variable GUID>), in double quotes.
 - *name* attribute— the name of the target variable (replacing the string <Variable Name>), in double quotes.
 - *type* attribute— the type of the target variable (replacing the string <Variable Type>), in double quotes. For example, string, integer, or boolean.
- *soa:Value* element value—the value you want to assign the variable (replacing the string <Variable Runtime Value>). It should *not* be in quotes. The *soa:Value* element value should be updated to contain the new runtime value for each job you submit (capture you start).

Start Formats

This operation has the following formats:

http://<host>:<port>/record/start

OR

http://<host>:<port>/record/start

[?<parameter>=<value>[&<parameter>=<value>]]

Parameters

The *ActionDuration* element in the *Response* is always updated immediately when a *Start* operation is executed. Other elements are also updated when specific optional parameters are utilized. Their values can be obtained using a [Status](#) operation.

Parameter	Description
duration (optional)	<p>Timecode; specifies the duration of the clip based on the number of frames captured. Default: 9 hours.</p> <p>For example: <code>duration=00:30:00:00</code>.</p> <p>When used in conjunction with <i>end</i>, the <i>end</i> timecode overrides <i>duration</i> when the specified timecode (or one later in time) is detected.</p> <p>If no <i>duration</i> or <i>end</i> is specified, the clip is recorded for 9 hours, the capture job is stopped, or disk space is exhausted.</p> <p>When <i>duration</i> is specified, the <i>ActionDuration</i> and <i>MarkIn</i> elements are updated immediately in the <i>Response</i>. The session continues until the number of frames in the file is functionally equivalent to the <i>duration</i> parameter. This allows for timecode jumps that occur during recording.</p> <p>Error: Returns an "Unable to start recording..." <i>Error</i> if timecode contains invalid timecode characters.</p>
end (optional)	<p>Timecode; specifies the clip's <i>exclusive</i> ending timecode.</p> <p>For example: <code>end=01:42:35:00</code>.</p> <p>The <i>end</i> timecode represents the timecode of the frame after the last frame of video. Recording stops when the specified timecode (or one later in time) is detected. If no <i>duration</i> or <i>end</i> is specified, the clip records for 9 hours, the capture job is stopped, or disk space is exhausted.</p> <p>When <i>end</i> is specified, the <i>ActionDuration</i> and <i>End</i> elements are updated immediately in the <i>Response</i> to <i>Start</i>. <i>MarkIn</i> and <i>MarkOut</i> are updated in the first valid <i>Status Response</i>.</p> <p>Error: Returns an "Unable to start recording..." <i>Error</i> if timecode contains invalid timecode characters.</p>
folderPath (optional)	<p>String; specifies the path to a folder where the clip file should be written. The path must end with a \.</p> <p>If a folder path is not specified, the clip is recorded in the storage folder associated with the respective workflow. If a new folder is added to an existing path, the folder will be created. If the folder could not be created an error will be reported.</p> <p>The Capture action must have the <i>Create output file(s) at job start</i> option enabled.</p> <p>For example: <code>folderPath=[Drive letter]:\[folder path]\</code> or <code>\\[Share server]\[Share]\[Folder Path]\</code></p>

Parameter	Description
name (optional)	<p>String; specifies a name for the clip. If a clip name is not specified, then a random name is generated.</p> <p>Note: For this name to be utilized, you must configure the Capture action to include the Base Name token in the Primary or Secondary Output's file name creation pattern in the Filename Pattern Editor (see the Capture Primary and Secondary Output Panels topic in the Lightspeed Live Guide for more detail). The appropriate file extension is automatically added to the file name. If a file exists, the web service will append an incremental integer to the file name.</p> <p>The Capture action must have the <i>Create output file(s) at job start</i> option enabled.</p> <p>For example: <code>name=LiveInOakland_Cam7_16062016_1342_PQ_234</code>.</p> <p>Error: Returns an error if the name contains Reserved Characters in Value Strings.</p>
discontinuity (optional)	<p>Keyword (Abort Ignore); specifies how to control recording when timecode discontinuities are observed in the video stream. When set to <i>Ignore</i>, any timecode discontinuity will be ignored and recording will continue. When set to <i>Abort</i>, the recording stops and the job is terminated when a timecode discontinuity is detected.</p>
rs422delay (optional)	<p>Integer; specifies the number of frames delay when capturing from a VTR deck, adjusting the RS-422 time code by the specified number of frames. Negative numbers indicate early frames (an advance); positive numbers indicate latent frames (a true delay). Default 0; minimum -3 frames; maximum +3 frames).</p> <p>For example: <code>rs422delay=-1</code>.</p> <p>RS-422 time code delay is used to compensate for VTR decks requiring frame timing adjustments. Frames can appear early or late, depending on the type and/or age of the deck used. This adjustment compensates for decks that are not 100% frame-accurate (or for systems with routing latency to the associated timecode). Most decks do not require an offset adjustment. If you have a deck that requires an adjustment, use a trial-and-error process to determine the correct offset.</p> <p>Note: The Capture action must have <i>Create output file(s) at job start</i> enabled.</p> <p>Error: Returns an error if <code>rs422delay</code> is not an integer or contains values out of range.</p>
start (optional)	<p>Timecode; specifies the clip's <i>inclusive</i> starting timecode.</p> <p>For example: <code>start=01:12:35:00</code>.</p> <p>The <i>start</i> time represents the at which to record the first frame. If a <i>start</i> time is not specified, the clip begins recording immediately.</p> <p>When a <i>start</i> parameter is supplied, the <i>ActionDuration</i> and <i>Start</i> elements are updated in the <i>Start Response</i>. The <i>MarkIn</i> element is updated in the first valid <i>Status Response</i>.</p> <p>Error: Returns an "Unable to start recording..." <i>Error</i> if timecode contains invalid timecode characters.</p>

Parameter	Description
tape (optional)	String; enables automatic RS422 control, based on the <i>start</i> and <i>end</i> time codes. Any string value may be entered. For example: <code>tape=LiveInOakland_Cam7</code> . Error: Returns an error if the string contains Reserved Characters in Value Strings .

Using Time-based Response Elements

These timecode-based *Response* elements are updated based on the parameters specified in the *Start* operation. Elements that are not updated (populated or changed value) on the initial *Response* are noted.

Start (without any parameters)

- *ActionDuration*

Start (start)

- *ActionDuration*
- *Start*

Start (duration)

- *ActionDuration*
- *MarkIn*—on the first valid Status operation

Start (start, duration)

- *ActionDuration*
- *Start*
- *End*
- *MarkIn*—on the first valid Status operation

Example—Immediate Start, No Duration

In this example, recording starts immediately and runs until it is stopped, 9 hours has elapsed or disk space is exhausted:

```
http://LS-SVR:17001/record/start
```

Typical Response

In this response, the clip GUID is in the *UUID* element. You can use this for [Status](#) or other operations on the clip. The *State* element indicates *Opened*, meaning that recording is ready to start or has started. Note that the *Identifier* is an all-zero GUID, indicating that Vantage has not yet updated its database with the job GUID—it may take several seconds (typically up to ten) to obtain the job GUID, using [Status](#). The *ActionDuration* is 32,400 seconds—9 hours—and typically is updated in the *Start* operation's *Response*.

```

<Response>
  <UUID>d35d4e41-1c87-478d-9c9e-edb64d3adf6e</UUID>
  <PercentCompleted>0</PercentCompleted>
  <Progress>0</Progress>
  <ActionDuration>32400</ActionDuration>
  <FPS>29.97002997003</FPS>
  <Start></Start>
  <End></End>
  <MarkIn></MarkIn>
  <MarkOut></MarkOut>
  <Excluding>False</Excluding>
  <Name>capture_default_undefined_TeamBUndefined.mxf</Name>
  <Path>\\LS-SVR\Live\31a5a309-3fde-43be-8855-
a1a39e63ccb1\capture_default_undefined_TeamBUndefined.mxf</Path>
  <HorizontalResolution>1920</HorizontalResolution>
  <VerticalResolution>1080</VerticalResolution>
  <FrameRate>29.97002997003</FrameRate>
  <Channels>16</Channels>
  <Identifier>00000000-0000-0000-0000-000000000000</Identifier>
  <State>Opened</State>
  <Access-Control-Allow-Origin>*</Access-Control-Allow-Origin>
  <EngineState>Running</EngineState>
  <EngineTime>12:52:28;13</EngineTime>
  <XMLRevision>2</XMLRevision>
</Response>

```

Example—Start with Time and Duration

In this example, recording starts later today at 2PM (14:00), and runs for 10 seconds:

```

http://LS-SVR:17001/record/
start?start=14:00:00:00?duration=00:00:10:00

```

Typical Response

In this *Response*, note that the recording lasts for 10.01 seconds:

```

<Response>
  <UUID>7db7c9bc-e2b9-4ec2-93e2-dfe89b876045</UUID>
  <PercentCompleted>0</PercentCompleted>
  <Progress>0</Progress>
  <ActionDuration>10.01</ActionDuration>
  <FPS>29.97002997003</FPS>
  <Start>14:00:00;00@29.97</Start>
  <End>14:00:10;00@29.97</End>
  <MarkIn/>
  <MarkOut/>
  <Name>capture_default_undefined_TeamBUndefined.mxf</Name>
  <Path>\\LS-SVR\Live\3c003eda-8222-476d-aafe-51770a1dd791/
capture_default_undefined_TeamBUndefined.mxf</Path>
  <HorizontalResolution>1920</HorizontalResolution>
  <VerticalResolution>1080</VerticalResolution>
  <FrameRate>29.97002997003</FrameRate>
  <Channels>16</Channels>
  <Identifier>00000000-0000-0000-0000-000000000000</Identifier>
  <State>Opened</State>

```

```
<Access-Control-Allow-Origin>*</Access-Control-Allow-Origin>
<EngineState>Running</EngineState>
<EngineTime>13:26:04;20</EngineTime>
<XMLRevision>2</XMLRevision>
</Response>
```

Modify

The *Modify* operation is used to modify the start time and/or duration of a specific clip, specified by its GUID or all clips in the clip list, when no GUID is provided. The clip may be in any state. The *start* parameter is always required. If a clip has already started recording, it does not affect the job.

This operation has the following formats:

```
http://<host>:<port>/record/modify  
?&start=<TimeCode>
```

OR

```
http://<host>:<port>/record/modify  
?uuid=<GUID>[&start=<TimeCode>[&duration=<TimeCode>]]
```

Note: If you only have one clip in the clip list (or you want to modify all clips in the clip list), you can issue a *Modify* without a UUID. It returns a *Response* with only the standard elements.

Parameters

Parameter	Description
uuid (optional)	GUID; the unique identifier of the target clip. When not supplied, all clips in the list are updated. For example: <code>uuid=21EC2020-3AEA-4069-A2DD-08002B30309D</code> Error: Returns <i>404 Not Found</i> if the specified clip is not in the clip list.
start (required)	Timecode; specifies the clip's new <i>inclusive</i> starting timecode—the <i>start</i> timecode of the first frame to be recorded—when executed before the scheduled <i>start</i> timecode. If recording is underway, you can specify any value—it does not affect the job, but it does update the <i>Start</i> element. For example: <code>start=01:12:35:00</code> . When a start timecode is specified, the <i>ActionDuration</i> and <i>Start</i> elements are updated in the <i>Response</i> . Error: Returns an "Unable to start recording..." <i>Error</i> if timecode contains invalid timecode characters.
duration (optional)	Timecode; specifies the new duration of the clip. For example: <code>duration=01:12:35:00</code> . When a duration timecode is specified, the <i>ActionDuration</i> , <i>Start</i> , <i>MarkIn</i> , and <i>MarkOut</i> elements are updated in the <i>Response</i> . Error: Returns an "Unable to start recording..." <i>Error</i> if timecode contains invalid timecode characters.

Errors

If you don't supply a *start* parameter, this error is returned:

```
<Response>
  <Error>Unable to modify recording. Start time code must be
    specified</Error>
  <Access-Control-Allow-Origin>*</Access-Control-Allow-Origin>
</Response>
```

Example

In this example, a new clip is created, to start at 18:00:00:00 for a duration of 10 minutes. Before 18:00 (recording has not started yet), the start time is modified to start at 18:30:00:00.

Here are the operations executed in this example:

Starting the Clip: `http://LS-SVR:17001/record/start?start=14:00:00:00&duration=00:10:00:00.`

Modifying the Clip Start Time: `http://LS-SVR:17001/record/modify?uuid=cd76fe46-700f-47b1-b483-7eaa1ed32241&start=11:30:00:00.`

Typical Response

Issuing the example *Modify* operation with a UUID and new start timecode results in the following *Response*:

```
<Response>
  <UUID>cd76fe46-700f-47b1-b483-7eaa1ed32241</UUID>
  <PercentCompleted>0</PercentCompleted>
  <Progress>0</Progress>
  <ActionDuration>10.01</ActionDuration>
  <FPS>29.97002997003</FPS>
  <Start>14:30:00;00@29.97</Start>
  <End>14:30:10;00@29.97</End>
  <MarkIn>14:00:00;00@29.97</MarkIn>
  <MarkOut>14:30:10;00@29.97</MarkOut>
  <Excluding>False</Excluding>
  <Name>capture_default_undefined_TeamBUndefined.mxf</Name>
  <Path>\\LS-SVR\Live\6220fd60-05c2-4b47-b481-f7a628708567/
capture_default_undefined_TeamBUndefined.mxf</Path>
  <HorizontalResolution>1920</HorizontalResolution>
  <VerticalResolution>1080</VerticalResolution>
  <FrameRate>29.97002997003</FrameRate>
  <Channels>16</Channels>
  <Identifier>8ee1376a-e1c0-490b-bea8-6ee42ee68fb2</Identifier>
  <State>Opened</State>
  <Access-Control-Allow-Origin>*</Access-Control-Allow-Origin>
  <EngineState>Running</EngineState>
  <EngineTime>15:25:07;27</EngineTime>
  <XMLRevision>2</XMLRevision>
</Response>
```

MarkOut | MarkIn

The MarkOut and MarkIn operations enable you to mark a specific segment of all clips in the clip list or a specific clip, specified by its GUID, for deletion during processing by a Vantage workflow.

Note: Mark In and Mark Out operations are only valid for TIFO format encoding. Using MarkOut/MarkIn with formats other than TIFO may have unintended consequences. Processing TIFO frames marked as OUT requires Vantage Transcoder 2012.1 or later in your Vantage domain. Alternatively, to temporarily halt streaming media to the Capture workflow, use [EditOut](#) | [EditIn](#).

All TIFO frames after a *MarkOut* is executed are marked with an OUT metadata tag. The TIFO decoder included in a Vantage workflow ignores frames marked as OUT and prevents them from being passed to a transcoder for encoding.

If a filler file is specified in the workflow, markout frames are replaced by frames in the same ordinal position in the filler file until the *MarkIn* operation is received.

All frames are marked as IN after a *MarkIn* has been executed.

You can use the [Status](#) command to determine the current state of these operations. The Excluding element in the Response has two values:

- Excluding = *False* means frames are not being affected by a Mark/Edit command.
- Excluding = *True* means frames are being affected by a Mark/Edit command.

These operations have the following formats:

```
http://<host>:<port>/record/<markout | markin>  
?[%timecode=<TimeCode>]
```

OR

```
http://<host>:<port>/record/<markout | markin>  
?[uuid=<GUID>[%timecode=<TimeCode>]]
```

On success, these operations return a *Response* element. If the uuid is not in the clip list, returns this error: "*The server encountered an error processing the request. See server logs for more details.*"

Parameters

Parameter	Description
uuid (optional)	GUID; the unique identifier of the clip on which to set the MarkOut/In point. When not supplied, all clips in the list are updated. For example: <code>uuid=21EC2020-3AEA-4069-A2DD-08002B30309D</code> Returns <i>404 Not Found</i> if the specified clip is not in the clip list.
timecode (optional)	Timecode; specifies the timecode in the future for a MarkOut/In point in the clip. For example: <code>timecode=01:12:35:00</code> . If a timecode is not specified the MarkOut/In point starts or stops immediately. MarkOut timecode is INCLUSIVE; the frame will be marked OUT. MarkIn timecode is EXCLUSIVE; the frame will be marked IN. Error: Returns an "Unable to start recording..." <i>Error</i> if timecode contains invalid timecode characters.

MarkOut Example

In this example, a new clip recording was started. Next, a MarkOut was executed, and finally, a MarkIn was executed:

Starting the Clip: `http://LS-SVR:17001/record/start`

Marking Out the Clip: `http://LS-SVR:17001/record/markout?uuid=d769249e-8bd8-4579-91b6-beb58f19d859`

Marking In the Clip: `http://LS-SVR:17001/record/markin?uuid=d769249e-8bd8-4579-91b6-beb58f19d859`

Typical MarkOut Response

Issuing the *MarkOut* operation returned this *Response*:

```
<Response>
  <UUID>d769249e-8bd8-4579-91b6-beb58f19d859</UUID>
  <State>Opened</State>
  <Access-Control-Allow-Origin>*</Access-Control-Allow-Origin>
  <EngineState>Running</EngineState>
  <EngineTime>15:35:02;20</EngineTime>
  <XMLRevision>2</XMLRevision>
</Response>
```

Typical MarkIn Response

Issuing the *MarkIn* operation returned this *Response*:

```
<Response>
  <UUID>d769249e-8bd8-4579-91b6-beb58f19d859</UUID>
  <State>Opened</State>
  <Access-Control-Allow-Origin>*</Access-Control-Allow-Origin>
  <EngineState>Running</EngineState>
```

```
<EngineTime>15:36:05;04</EngineTime>  
<XMLRevision>2</XMLRevision>  
</Response>
```

EditOut | EditIn

The EditOut and EditIn operations temporarily halt streaming to the Capture workflow, to prevent recording a segment between the EditOut and the EditIn of all clips in the clip list or a specific clip, specified by its GUID, where recording of the streamed clip resumes.

Note: As an alternative, when capturing TIFO format, you can use [MarkOut](#) | [MarkIn](#) operations and perform the cuts directly in the workflow.

EditOut/EditIn functionality differs, based on whether a Lightspeed Live Capture workflow's Filler feature has been enabled in the Capture action, and a file specified. This feature is identical to using the Out and In buttons in the Capture Portal with a manually-triggered capture.

If Filler is *not checked* in the Capture action, issuing an EditOut operation prevents source frames from being added to the file; issuing an EditIn operation returns to adding source frames to be written to the file.

If Filler is *checked* in the Capture action, issuing an *EditOut* enable frames from the Filler file to be written to the file. Issuing an *EditIn* returns to adding source frames to be written to the file. If the filler is longer than the edit out period, it will loop.

Note: See the Lightspeed Live Guide for details on enabling and using Filler.

You can use the [Status](#) command to determine the current state of these operations. The Excluding element in the Response has two values:

- Excluding = *False* means frames are not being affected by a Mark/Edit command.
- Excluding = *True* means frames are being affected by a Mark/Edit command.

These operations have the following format:

```
http://<host>:<port>/record/<editin | editout>  
[?timecode=<TimeCode>]
```

OR

```
http://<host>:<port>/record/<editin | editout>  
?[uuid=<GUID>&timecode=<TimeCode>]
```

On success, these operations return a *Response* element.

Parameters

Operation	Description
uuid (optional)	GUID; the unique identifier of the clip on which to set the EditOut EditIn point. When not supplied, all clips in the list are updated. For example: <code>uuid=21EC2020-3AEA-4069-A2DD-08002B30309D</code> Returns <i>404 Not Found</i> if the specified clip is not in the clip list.
timecode (optional)	Timecode; specifies the timecode for an EditOut EditIn point in the clip. If a timecode is not specified, the EditOut EditIn operations are issued immediately. For example: <code>timecode=01:12:35:00</code> . EditOut timecode is <i>inclusive</i> ; the frame will be edited out. EditIn timecode is <i>exclusive</i> ; the frame will be edited in. Error: Returns an "Unable to start recording..." <i>Error</i> if timecode contains invalid timecode characters.

Example

In this example, a new clip was started. Next, an EditOut was executed, and finally, an EditIn was executed:

Starting the Clip: `http://LS-SVR:17001/record/start`

Editing Out the Clip: `http://LS-SVR:17001/record/
EditOut?uuid=ee74da56-6e01-4e2f-b1c3-ed3449878934`

Editing In the Clip: `http://LS-SVR:17001/record/editin?uuid=ee74da56-
6e01-4e2f-b1c3-ed3449878934`

Whether you supply a timecode or not, the *Response* is essentially the same.

Typical EditOut Response

Issuing this *EditOut* operation returned this *Response*:

```
<Response>  
  <UUID>ee74da56-6e01-4e2f-b1c3-ed3449878934</UUID>  
  <State>Opened</State>  
  <Access-Control-Allow-Origin>*</Access-Control-Allow-Origin>  
  <EngineState>Running</EngineState>  
  <EngineTime>15:45:23;09</EngineTime>  
  <XMLRevision>2</XMLRevision>  
</Response>
```

Typical EditIn Response

Issuing this *EditIn* operation returned this *Response*:

```
<Response>  
  <UUID>ee74da56-6e01-4e2f-b1c3-ed3449878934</UUID>  
  <State>Opened</State>
```

```
<Access-Control-Allow-Origin>*</Access-Control-Allow-Origin>  
<EngineState>Running</EngineState>  
<EngineTime>15:47:04;15</EngineTime>  
<XMLRevision>2</XMLRevision>  
</Response>
```

Message

The *Message* operation enables an application to control a VTR by sending Sony VTR operations to the connected VTR from the Lightspeed Live server via RS-422 and retrieving the *Response*.

Note: The Sony 9-pin operations are specified in the Sony Video Cassette Recorder/Player Protocol of Remote (9-pin) Connector, 2nd Edition.

The SDI Source Input associated with the Capture workflow receiving this operation must be connected to a VTR device via EIA RS-422A.

You can not use this operation unless the target workflow is in an active state with a Monitor Status of VTR connected. (View the Monitor Status tab in Workflow Designer).

This operation has the following format:

`http://<host>:<port>/record/Message?request=<Sony Operation>`

Parameters

Parameter	Description
request (required)	<p>String, reserved numeric strings identified by Sony for 9-pin device operations.</p> <p>For example: <code>request=2000</code></p> <p>Typical Sony operations (supported by most VTRs) include:</p> <ul style="list-style-type: none">• Stop = 2000• Play = 2001• Fast Forward = 2010• Step Forward = 2014• Rewind = 2020• Step Backward = 2024• Cue with DATA (Go To) = 2431[TC DATA] - TC = 09:59:59:00; TC DATA = 00595909 <p>Reference your VTR guide for supported operations.</p>

Example

In this example, the *LS_Live* server is contacted on port 17001, issuing a Sony VTR Stop operation:

`http://LS-SVR:17001/record/Message?request=2000`

Typical Response

The XML responses from *Message* operations do not follow the pattern of the other responses. Instead, they are specified by a Microsoft schema. *Message* operations return a *string* element, whose data value is a string. The last two characters are a checksum. The remaining string is the return value.

Issuing this *Message* operation with a 2001 request (Play):

```
http://LS-SVR:17001/record/message?request=2001
```

returned the following XML:

```
<string xmlns="http://schemas.microsoft.com/2003/10/Serialization/">100111</string>
```

The response string 1001 (dropping the last 2 numbers—the checksum) = ACK.

In this example, the *Message* operation is issued for a Status Sense:

```
http://LS-SVR:17001/record/message?request=612002
```

The response is:

```
<string xmlns="http://schemas.microsoft.com/2003/10/Serialization/">7220000193</string>
```

This response string (7220000193) is de-constructed as:

7220 (7X20) = Status Data returning 2 data bytes (Data No. 0 through Data No. 1)

Data No. 0 = 20 = 00100000 (Bit 5 = 1 which means Tape or Cassette IN; more properly NOT OUT)

Data No. 1 = 01 = 00000001 (Bit 0 = 1 which indicates a status of PLAY).

Stop

Stops a queued or in-progress job and removes the specified clip from the clip list or—when no GUID is provided—stops and removes all clips from the clip list.

This operation has the following format:

```
http://<host>:<port>/record/stop
```

OR

```
http://<host>:<port>/record/stop  
?uuid=<GUID>[&end=<TimeCode>]
```

The Response that is returned contains one *UUID* element with the GUID for each clip that was stopped.

Parameters

Parameter	Description
uuid (required)	GUID; the unique identifier of the clip to stop. For example: <code>uuid=21EC2020-3AEA-4069-A2DD-08002B30309D</code> Error: Returns <i>404 Not Found</i> if the clip is not in the clip list.
end (optional)	Timecode; specifies the clip's <i>exclusive</i> new end timecode. For example: <code>end=01:42:35:00</code> . The end timecode represents the timecode of the frame after the last frame of video. If an end time is not specified the clip stops recording immediately. This parameter has no affect if the duration parameter was used within the job's start operation. Error: Returns <i>400 Bad Request</i> if the end time results in a clip with a duration longer than initially specified. Note: You can only change the end timecode on files recorded without an initial duration or when recording QuickTime Closed or MXF OP1a Closed container formats.

Example

```
http://LS-SVR:17001/record/stop  
?uuid=d35d4e41-1c87-478d-9c9e-edb64d3adf6e
```

Typical Response

Issuing this *Stop* operation on the same clip started in the example in the [Start](#) operation returned the following *Response*:

```
<<Response>
  <UUID>d35d4e41-1c87-478d-9c9e-edb64d3adf6e</UUID>
  <PercentCompleted>0</PercentCompleted>
  <Progress>383.783395833333</Progress>
  <ActionDuration>32400</ActionDuration>
  <FPS>29.97002997003</FPS>
  <Start/>
  <End/>
  <MarkIn>12:52:28;13@29.97</MarkIn>
  <MarkOut/>
  <Excluding>False</Excluding>
  <Name>capture_default_undefined_TeamBUndefined.mxf</Name>
  <Path>\\LS-SVR\\Live\\31a5a309-3fde-43be-8855-a1a39e63ccb1/
capture_default_undefined_TeamBUndefined.mxf</Path>
  <HorizontalResolution>1920</HorizontalResolution>
  <VerticalResolution>1080</VerticalResolution>
  <FrameRate>29.97002997003</FrameRate>
  <Channels>16</Channels>
  <Identifier>f66d4a1d-0c30-4829-90c3-bed03454b3f3</Identifier>
  <State>Opened</State>
  <Access-Control-Allow-Origin>*</Access-Control-Allow-Origin>
  <EngineState>Running</EngineState>
  <EngineTime>12:58:52;19</EngineTime>
  <XMLRevision>2</XMLRevision>
</Response>
```

Status

The *Status* operation supplies status information for a clip when the clip's GUID is supplied. When you don't specify a GUID, the Status operation returns the clip list (the list of clips currently in the capture queue).

You can obtain the clip GUID from its *Start* operation or from a separate Status operation that returns the clip list.

You must use the *Status* operation to obtain the job ID GUID in the *Identifier* element, which identifies the Vantage capture job. The *Start* operation does not return the job ID GUID in its Response. After starting a clip capture, you may have to wait to issue the *Status* operation or a few seconds (or poll repeatedly and test) for the status to be fully-updated and available. For example, the Job ID (the value in the *Identifier* element) is returned as zeros until the process is fully updated.

This operation has the following formats:

```
http://<host>:<port>/record/status
```

OR

```
http://<host>:<port>/record/status?uuid=<GUID>
```

Parameters

Parameter	Description
uuid (optional)	GUID; the unique identifier of the clip to obtain status information from. For example: <code>uuid=21EC2020-3AEA-4069-A2DD-08002B30309D</code> When this operation is issued with a GUID, the <i>Response</i> provides status details for the associated clip. Error: Returns <i>404 Not Found</i> if the clip is not in the clip list.

Example

```
http://LS-SVR:17001/record/status  
?uuid=d35d4e41-1c87-478d-9c9e-edb64d3adf6e
```

In this example, the uuid is the clip GUID associated with the job created in the example provided in the *Start* operation topic.

Typical Response

Issuing this *Status* operation returned the following *Response*.

Note: You may have to wait (or poll) for several seconds (typically up to ten) to obtain a valid job ID GUID in the *Identifier* element.

The GUID in the *Identifier* element is the job ID GUID of the job created by issuing the *Start* command. The *Progress* value of 173 is the totaled elapsed recording time (in seconds) so far.

```
<Response>
  <UUID>d35d4e41-1c87-478d-9c9e-edb64d3adf6e</UUID>
  <PercentCompleted>0</PercentCompleted>
  <Progress>173.773604166667</Progress>
  <ActionDuration>32400</ActionDuration>
  <FPS>29.97002997003</FPS>
  <Start/>
  <End/>
  <MarkIn>12:52:28;13@29.97</MarkIn>
  <MarkOut/>
  <Excluding>False</Excluding>
  <Name>capture_default_undefined_TeamBUndefined.mxf</Name>
  <Path>\\LS-SVR\Live\31a5a309-3fde-43be-8855-a1a39e63ccb1/
capture_default_undefined_TeamBUndefined.mxf</Path>
  <HorizontalResolution>1920</HorizontalResolution>
  <VerticalResolution>1080</VerticalResolution>
  <FrameRate>29.97002997003</FrameRate>
  <Channels>16</Channels>
  <Identifier>f66d4a1d-0c30-4829-90c3-bed03454b3f3</Identifier>
  <State>Opened</State>
  <Access-Control-Allow-Origin>*</Access-Control-Allow-Origin>
  <EngineState>Running</EngineState>
  <EngineTime>12:55:22;20</EngineTime>
  <XMLRevision>2</XMLRevision>
</Response>
```

Variables

The purpose of the *Variables* operation is to obtain the variables defined in the Capture action of the target workflow, so that you can start a job using the [Start](#) POST operation and supply the required run-time variable values.

The *Variables* operation returns the variables as a Customization XML describing the variables in the Capture action of the target workflow. The XML contains the variables and current values in the `/Customization/Conditions/a:Condition/` elements.

Note: If a variable was added to the Capture action, it uses namespace b:, not a:.

This operation has the following format:

`http://<host>:<port>/record/variables`

Example

Here is an example of a *Variables* operation:

`http://LS-SVR:17001/record/variables`

Typical Response

Issuing this *Variables* operation on the target workflow illustrated in the [Start](#) example returned the following *Response*, where *Condition* elements defined each variable and its attributes:

- TeamA
- TeamB

```
<Customization xmlns="http://schemas.datacontract.org/2004/07/
Telestream.Soa.Facility.Live.Vocabulary" xmlns:i="http://
www.w3.org/2001/XMLSchema-instance">
  <xmlns xmlns="http://schemas.datacontract.org/2004/07/
Telestream.Soa.Vocabulary" xmlns:a="http://
schemas.datacontract.org/2004/07/System.Xml.Serialization"
i:nil="true"/>
  <Identifier xmlns="http://Telestream.Vantage.Sdk/2010/
07">42c887d0-a06c-4e66-96d7-82cf75ba37b7</Identifier>
  <Categories xmlns="http://Telestream.Vantage.Sdk/2010/07"
xmlns:a="http://schemas.datacontract.org/2004/07/
Telestream.Soa.Vocabulary"/>
  <Components xmlns="http://Telestream.Vantage.Sdk/2010/07"/>
  <CustomEditorType xmlns="http://Telestream.Vantage.Sdk/2010/07"
i:nil="true"/>
  <Description xmlns="http://Telestream.Vantage.Sdk/2010/07"
i:nil="true"/>
  <Name xmlns="http://Telestream.Vantage.Sdk/2010/07" i:nil="true"/>
  <ParameterSelections xmlns="http://Telestream.Vantage.Sdk/2010/07"
xmlns:a="http://schemas.datacontract.org/2004/07/
Telestream.Soa.Vocabulary"/>
```

```
<ParameterSetCollections xmlns="http://Telestream.Vantage.Sdk/
2010/07" xmlns:a="http://schemas.datacontract.org/2004/07/
Telestream.Soa.Vocabulary"/>
<Parameters xmlns="http://Telestream.Vantage.Sdk/2010/07"/>
<Summary xmlns="http://Telestream.Vantage.Sdk/2010/07"
i:nil="true"/>
<Conditions xmlns:a="http://Telestream.Vantage.Sdk/2010/07">
<a:Condition>
<xmlns xmlns="http://schemas.datacontract.org/2004/07/
Telestream.Soa.Vocabulary" xmlns:b="http://
schemas.datacontract.org/2004/07/System.Xml.Serialization"
i:nil="true"/>
<a:Identifier>d27afbf9-8ed3-45fb-9ae5-0992a1757d7f</a:Identifier>
<a:Categories xmlns:b="http://schemas.datacontract.org/2004/07/
Telestream.Soa.Vocabulary"/>
<a:Components/>
<a:CustomEditorType i:nil="true"/>
<a:Description/>
<a:Name>TeamB</a:Name>
<a:ParameterSelections xmlns:b="http://schemas.datacontract.org/
2004/07/Telestream.Soa.Vocabulary"/>
<a:ParameterSetCollections xmlns:b="http://
schemas.datacontract.org/2004/07/Telestream.Soa.Vocabulary"/>
<a:Parameters/>
<a:Summary i:nil="true"/>
<a:Instance>106c7a70-42b0-4ae4-9d7d-7bf54e9d6a78</a:Instance>
<a:ConditionValue>
<a:ComplexValue i:nil="true"/>
<a:Text xmlns:b="http://schemas.microsoft.com/2003/10/
Serialization/Arrays" i:nil="true"/>
<a:Default>
<a:ComplexValue i:nil="true"/>
<a:Text xmlns:b="http://schemas.microsoft.com/2003/10/
Serialization/Arrays">
<b:string>TeamBUndefined</b:string>
</a:Text>
</a:Default>
</a:ConditionValue>
<a:TypeCode>String</a:TypeCode>
</a:Condition>
<a:Condition>
<xmlns xmlns="http://schemas.datacontract.org/2004/07/
Telestream.Soa.Vocabulary" xmlns:b="http://
schemas.datacontract.org/2004/07/System.Xml.Serialization"
i:nil="true"/>
<a:Identifier>d2907504-ee16-4af2-bfcd-eb18e0bde7b0</a:Identifier>
<a:Categories xmlns:b="http://schemas.datacontract.org/2004/07/
Telestream.Soa.Vocabulary"/>
<a:Components/>
<a:CustomEditorType i:nil="true"/>
<a:Description/>
<a:Name>TeamA</a:Name>
<a:ParameterSelections xmlns:b="http://schemas.datacontract.org/
2004/07/Telestream.Soa.Vocabulary"/>
<a:ParameterSetCollections xmlns:b="http://
schemas.datacontract.org/2004/07/Telestream.Soa.Vocabulary"/>
<a:Parameters/>
```

```
<a:Summary i:nil="true"/>
<a:Instance>6ad3eec9-28cd-499c-8927-0710f1b5622a</a:Instance>
<a:ConditionValue>
<a:ComplexValue i:nil="true"/>
<a:Text xmlns:b="http://schemas.microsoft.com/2003/10/
Serialization/Arrays" i:nil="true"/>
<a:Default>
<a:ComplexValue i:nil="true"/>
<a:Text xmlns:b="http://schemas.microsoft.com/2003/10/
Serialization/Arrays">
<b:string>default_undefined</b:string>
</a:Text>
</a:Default>
</a:ConditionValue>
<a:TypeCode>String</a:TypeCode>
</a:Condition>
</Conditions>
<Labels xmlns:a="http://Telestream.Vantage.Sdk/2010/07"/>
</Customization>
```

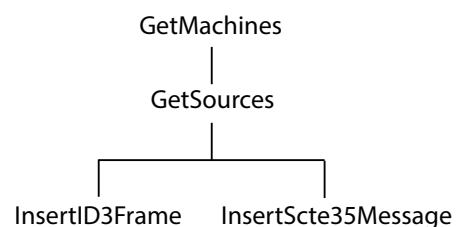

Source Operations

You can use the Lightspeed Source web service operations in a program to identify Live servers by GUID, obtain a list of sources to operate on, and insert various SCTE-35 triggers and ID3 frame tags into a source. Source operations can be used in both capture and stream processes.

- [Introduction](#)
- [Obtaining Help for Source Operations](#)
- [GetMachines](#)
- [GetSources](#)
- [InsertID3Frame](#)
- [InsertScte35Message](#)

Note: All Source operations are located at the root level: `http://<host>:<port>/`.

In this diagram, the operations are organized hierarchically, by machine GUID and Source GUID requirement.



Note: To display help for Source operations, enter `http://<host>:<port>/help`.

Introduction

The Live Source interface is implemented over HTTP as a RESTful interface, and results (if any) are returned in JSON format.

The following topics provide general information about the Source API and how operations are presented. This reference is intended for readers who understand how to use Live Stream and Capture; for information, read the Lightspeed Live Guide.

Port for Lightspeed Live/Capture Server Access

Port 15000 is used to access the Source API on a Lightspeed Live or Capture server.

This port can be changed in the Lightspeed Live Stream web app. Under Settings, update the Source REST API server port. After changing a port, you must restart the Live Source service for the new port number to take effect.

Using Live Stream Groups via the API

A Live Stream group is a set of Live servers organized into a single, functional system, enabling you to increase scalability without significantly increasing complexity. Where reference is made to a Live server, it may be a single server or a group of servers—there is no difference in functionality or reporting.

Using Shared vs. Dedicated Components

Unlike most other categories of components in a Live server, media sources are associated with a port directly on a specific server. Thus, Source operations must be executed directly on that server. When you are operating on a media source, you must execute the operation using the DNS name or the IP address of the Live server where the source was created.

Live Source Component Hierarchy

When operating on sources, you must first obtain a list of machines and then identify the machine you want to access by GUID: Machine > Source. Next, obtain a list of Sources and then identify the source you want to access by GUID.

Obtaining Help for Source Operations

You can obtain information about these operations in a variety of ways:

Displaying the Operations of a Live Source Service

To list all of the operations in a Live Source service, execute this command:

`http://<host>:<port>/help.`

For example: `http://10.0.25.158:15000/help`

Help returns a web page listing all Source operations:

Operations at http://10.0.5.126:15000/		
This page describes the service operations at this endpoint.		
Uri	Method	Description
GetMachines	GET	Get a list of machines in this system.
GetSources	GET	Get a list of sources for the specified machine.
InsertID3Frame	POST	Insert an ID3 tag with a PRIV frame with the specified type and value in the given source. If a time code is not specified, the ID3 tag will be inserted immediately.
InsertScte35Message	GET	Insert a SCTE-35 message in the given source. If a time is not specified, the message will be inserted immediately.

Hover over the GET or POST link to view the format of the operation.

Displaying Operation Details

To display details about a specific operation and view the syntax of the JSON response:

- Click on GET or POST link for the operation
- or
- Execute the operation in the /help/operations directory, (no parameters).

This form of Help has the following format:

`http://<host>:<port>/help/operations/<Operation>.`

For example:

`http://10.0.25.158:15000/help/operations/GetSources`

Help returns a web page illustrating the operation and its method plus example responses:

Reference for `http://ll-pm:15000/Sources/GetSources?machine={MACHINE}`

Get a list of sources for the specified machine.

Url: `http://ll-pm:15000/Sources/GetSources?machine={MACHINE}`

HTTP Method: GET

Message direction	Format	Body
Request	N/A	The Request body is empty.
Response	Xml	Example,Schema
Response	Json	Example

The following is an example response Xml body:

```
<ArrayOfBrief xmlns="http://schemas.datacontract.org/2004/07/Telestream.Soa.Live.Vocabulary">
  <Brief>
    <Description>String content</Description>
    <Identifier>1627aea5-8e0a-4371-9022-9b504344e724</Identifier>
    <Name>String content</Name>
  </Brief>
  <Brief>
    <Description>String content</Description>
    <Identifier>1627aea5-8e0a-4371-9022-9b504344e724</Identifier>
    <Name>String content</Name>
  </Brief>
</ArrayOfBrief>
```

The following is an example response Json body:

```
[{
  "Description": "String content",
  "Identifier": "1627aea5-8e0a-4371-9022-9b504344e724",
  "Name": "String content"
}]
```

The help page displays two types of responses: XML and JSON. JSON responses are returned in all Source operations.

GetMachines

The purpose of this GET operation is to identify the Live server (or in the case of a group, all of the Live servers in the group) by GUID.

This operation is a prerequisite for other operations which require a machine GUID.

To execute this operation, use the Windows domain name or the IP address of the Live server or any Live server in the group.

GetMachines has the following format:

```
http://<host>:<port>/GetMachines
```

Operation Sequence

No other operations are required before you can execute this operation.

Results

Upon success, *GetMachines* returns an array, with a record of the Live server (or, in the case of a group, each Live server) GUID and Name.

Example

```
http://10.9.9.9:15000/GetMachines
```

Typical Response

In this response, three server records are listed with their Identifier and Name. You can extract each machine's GUID from the Identifier and use it to connect and monitor or control its resources. Of course, in a standalone system, only one record is returned.

```
[
  {
    "Description":null,
    "Identifier":"89d2be4b-be21-4838-a551-522cce299fbe",
    "Name":"LL-PM-1"
  },
  {
    "Description":null,
    "Identifier":"4bd2be89-2c24-3947-a432-484bca2387fba",
    "Name":"LL-PM-1"
  },
  {
    "Description":null,
    "Identifier":"43b2ff5b-ac29-48573-c443-567cad734efa",
    "Name":"LL-PM-1"
  }
]
```

GetSources

The purpose of this GET operation is to identify all of the video sources that are available on the target Live server, and return them in a list for further use.

Note: Sources are defined for a specified hardware port on a specific server. Thus, the host that you specify must be the server where the Source was added.

GetSources has the following format:

```
http://<host>:<port>/Sources/GetSources?machine={MACHINE GUID}
```

Operation Sequence

Execute the following operation to obtain the required GUID for this operation:

[GetMachines](#) (machine GUID)

Required Parameter

Parameter	Description
machine	GUID; string that identifies a specific Live server.

Results

On success, *GetSources* returns a set of records; one for each source in the Live server.

Example

```
http://10.9.9.9:15000/Sources/GetSources  
?machine=1129625b-0d7d-490a-aa3f-315214a0b6f2
```

Typical Response

In this response, all of the Sources that are operational on the target server are listed along with their Identifier and Name values, which you can use to query and use them.

```
[  
  {  
    "Description":null,  
    "Identifier":"4ad20640-a5d8-45d1-a035-3a38227f21d5",  
    "Name":"QA-VL-LIVE-9 - SDI Input 3"  
  },  
  {  
    "Description":null,  
    "Identifier":"8a7bc656-6704-41a2-8161-f4d0d5a5f8de",  
    "Name":"Test1"  
  }  
]
```

InsertID3Frame

This POST operation inserts an ID3 tag with a single PRIV frame and the specified type and value in the source at the specified timecode. If a time code is not specified, the tag is inserted immediately. Sources are defined for a specified hardware port on a specific server. Thus, the host that you specify must be the server where the Source was added.

InsertID3Frame has the following format:

```
http://<host>:<port>/InsertID3Frame?  
source={SOURCE GUID}&type={FRAME TYPE}&timeCode={TIMECODE VALUE}  
Body <MIME TYPE>: {type value}
```

Operation Sequence

Execute the following operation to obtain the required GUID for this operation:

[GetMachines](#) (machine GUID) > [GetSources](#) (Source GUID)

Parameters

Parameter	Description
source	GUID; string that identifies a specific source on the Live server. For example: <code>source=442082bf-bde8-44b5-9bce-832b6d0fd885</code>
type	String; the type of PRIV frame. For example: <code>type=com.cisco.streaming.SplicePoint.0</code>
timeCode (optional)	Timecode; time code value in the source at which to insert the tag. If a time code is not specified, the tag is inserted immediately. For example: <code>source=01:03:15:00@29.97</code>

Required Post Body

Type Value	String; appropriately-encoded string that is the value for the type key-pair value inserted into the video. This operation supports four MIME types: <ul style="list-style-type: none">- text/plain (base 64-encoded binary data)- application/xml or text/xml (un-encoded XML)- application/octet-stream (raw binary data).
------------	---

Results

Upon success, *InsertID3Frame* adds an ID3 tag with a PRIV frame of the specified type and value to the source at the indicated time frame (or immediately) and returns a record with the string "Success":

```
[
  {
    "Success"
  }
]
```

Example

```
http://10.9.9.9:15000/InsertID3Frame?
source=442082bf-bde8-44b5-9bce-
832b6d0fd885&type="com.cisco.streaming.SplicePoint.0"
```

Body (text/plain MIME type with base-64-encoded binary data)

```
PD94bWwgdmVyc2lvbj0iMS4wIiBlbmNvZGluZz0iVVRGLTgiPz4NCjxNYXJrZXI+DQ
o8TWfYa2VySUQ+MHgxNWQxZTg8L01hcmtlccklEPg0KPFByZXJvbGxITlM+MDwvUHJl
cm9sbEhOUz4NCjxTcGxpY2U+DQo8RXZlbnRJRd4xNDI5OTkyPC9FdmVudElEPg0KPE
9ldE9mTmV0d29yaz4wPC9PdXRPZk5ldHdvcm5+DQo8UHJvZ3JhbVNwbG1jZT4xPC9Q
cm9ncmFtU3BsaWNlPg0KPERlcmF0aW9uPjA8L0RlcmF0aW9uPg0KPFNwbG1jZU1tbW
VkaWF0ZT4wPC9TcGxpY2VJbW1lZG1hdGU+DQo8VGltZVNwZWNPZml1ZD4xPC9UaW1l
U3BlY2lmaWVhPg0KPEF1dG9SZXR1cm4+MDwvQXV0b1JldHVybj4NCjwvU3BsaWNlPg
0KPFByb2dyYW0+DQo8UHJvZ3JhbUlEPjE5NDwvUHJvZ3JhbUlEPg0KPEF2YWlsTnVt
PjA8L0F2YWlsTnVtPg0KPEF2YWlsRXhwZWNOZWQ+MDwvQXZhaWxFeHB1Y3RlZD4NCj
wvUHJvZ3JhbT4NCjxUaW1lPg0KPFNwbG1jZVRpbWVITlM+MTAxOTE4MjEzMDAwMDA8
L1NwbG1jZVRpbWVITlM+DQo8U3BsaWNlRGF0ZVRpbWU+MjAxNy0xMC0yNlQwMDoxNz
oyM1o8L1NwbG1jZURhdGVUaW1lPg0KPERlcmF0aW9uSE5TPjA8L0RlcmF0aW9uSE5T
Pg0KPC9UaW1lPg0KPC9NYXJrZXI+
```


InsertScte35Message

The purpose of this GET operation is to insert a SCTE-35 message in the source. If a time is not specified, the message will be inserted immediately.

Note: Sources are defined for a specified hardware port on a specific server. Thus, the host that you specify must be the server where the Source was added.

InsertScte35Message has the following format:

```
http://<host>:<port>/  
InsertScte35Message?source={SOURCE}&scte35Message={SCTE35MESSAGE}&  
time={TIME}&eventID={EVENTID}
```

Operation Sequence

Execute the following operation to obtain the required GUID for this operation:

[GetMachines](#) (machine GUID) > [GetSources](#) (Source GUID)

Parameters

Parameter	Description
source	GUID; string that identifies a specific source on the Live server. For example: <code>source=442082bf-bde8-44b5-9bce-832b6d0fd885</code>
scte35 Message	String; keyword that defines the SCTE-35 message. You can enter a string representing the SCTE segmentation type ID in plain text or as a string representation of the hex value. Capitalization is ignored, and with or without spaces. For example: "Program Start", "programstart", and "0x10" all refer to the same message; creating a SCTE trigger and setting its segmentation type ID to 0x10 (16 in base 10). For example: <code>scte35Message="Program Start"</code> See the SCTE Commands table below.
timeCode (optional)	Timecode; time code value in the source at which to insert the SCTE-35 message. If a time code is not specified, the message is inserted immediately. For example: <code>timeCode=01:03:15:00@29.97</code>
eventID	Integer value; a 32-bit unsigned integer value which specifies the ID of the message.

SCTE-35 Commands

Command	Hex Value	Text Value
Program Start	0x10	ProgramStart
Program End	0x11	ProgramEnd
National Break Start	0x30	ProviderAdStart
National Break End	0x31	ProviderAdEnd
Local Break Start	0x32	DistributorAdStart
Local Break End	0x33	DistributorAdEnd
Ad Break Start	0x34	AdBreakStart
Ad Break End	0x35	AdBreakEnd

Results

Upon success, *InsertScte35Message* adds the specified message and its type value in the body, to the source at the indicated time frame (or immediately) and returns a record with the string "Success".

```
[
  {
    "Success"
  }
]
```

Example

```
http://10.9.9.9:15000/InsertScte35Message?
source=442082bf-bde8-44b5-9bce-832b6d0fd885&scte35Message="Program
Start"&eventID=4967295
```