# Lightspeed Live Capture Web API Reference

Vantage 7.1
Live Capture V2.3
Live Capture ComponentPac 7.1.3

March 2018

242602

# Copyrights and Trademark Notices

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.

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.

## 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.

## 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 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:** 242603

**Date:** March 2018

# Contents

# 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.

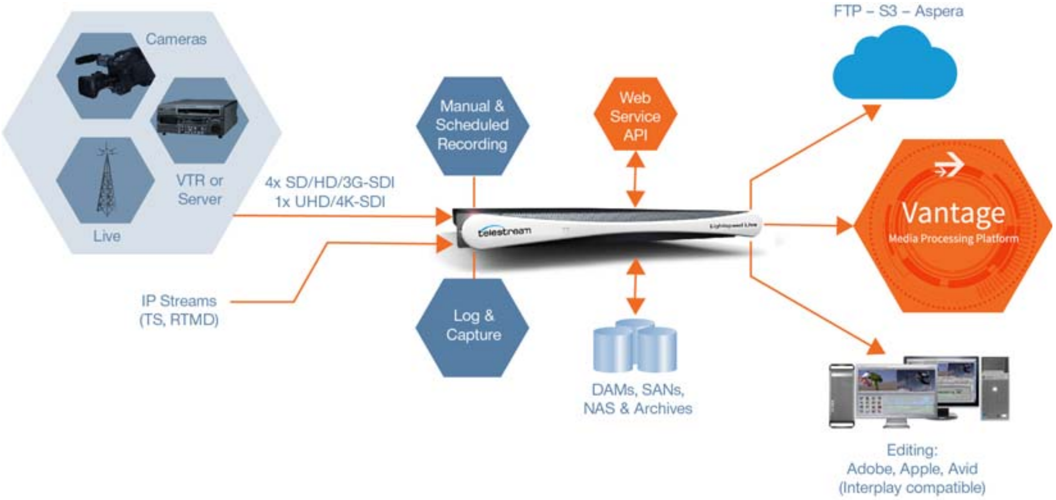
You can also create a web services-based system to control streaming and capture beyond the functionality or capability of the general-purpose Vantage Capture and Stream web applications, to meet your organization's requirements.

- Lightspeed Live Capture Web API
- Lightspeed Live Stream Web API
- Lightspeed Live Source Web API
- Operation & Response Formats
- Reserved Characters in Value Strings
- Adjusting Chrome Browser Prediction Settings

**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.



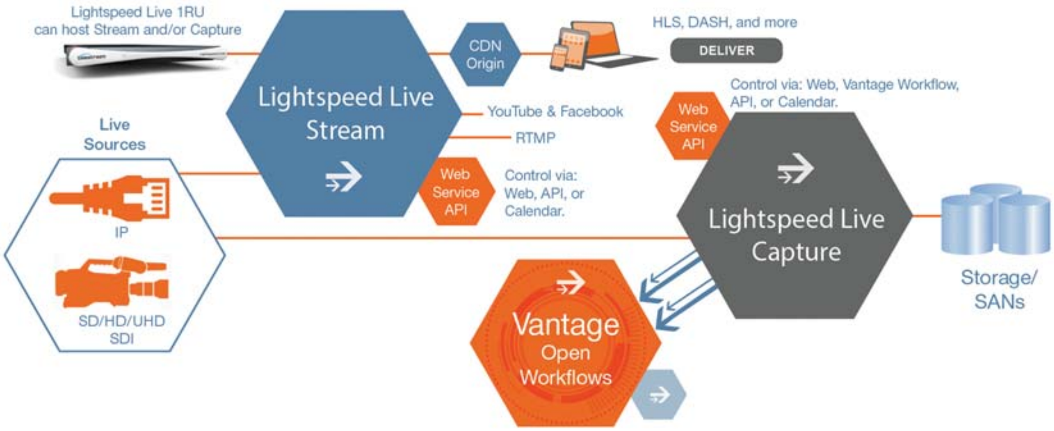
Lightspeed Live Capture workflows can be configured to publish a web service that enables media clips to be recorded using this web services operation set. When a Capture workflow is activated in Vantage via Workflow Designer, the web service begins listening for requests on the specified port. (The user will be warned if the port is in use by another Capture workflow or other service on the host computer.)

Your custom 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. 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 application 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>/record/<operation>
[?<parameter>=<value>[&<parameter>=<value>]]
```

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

Lightspeed Live systems respond to these 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.

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

## Topics

## 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, as noted in the table following.

**Note:** Operation names and keywords in operations are not case-sensitive, although the keywords are represented in this guide using camel case for readability. 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. Multiple 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:

| Term | Description |
|---|---|
| host | The Windows domain name or the IP address of the Lightspeed Live Capture or Stream server you are targeting.<br>For example: *localhost* \| `LightspeedServer` \| `192.168.1.23`. |
| port | The TCP port number assigned to the web service. For Live Capture, the port number (default: 17000) is user-selectable and displayed in the web service configuration panel of the Capture action inspector in the target Vantage workflow. (See the Lightspeed Live Guide or man page for the Capture action.) The Live REST API server port (default: 18000) and Source REST API server port (default: 15000) is also selectable in the Live Stream web app settings. |
| operation | Reserved word; the web service operation to execute. |
| parameter | A named parameter defined by the web service 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.

In the context of Lightspeed Live, 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.

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. 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 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.

# Response Formats

Response formats vary by API.

## 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

Telestream recommends that you do not use the following reserved characters in any parameter values.

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

Certain characters are omitted; others are changed to a space character. In some circumstances, the following error is displayed: “Request Error - The server encountered an error processing the request. See server logs for more details.”

# Adjusting Chrome Browser Prediction Settings

The Prediction Service settings in the Chrome web browser may cause issues when issuing Lightspeed Live web API operations directly from Chrome. Chrome may attempt to complete the URL address and associated arguments by using its own prediction method.

To prevent possible errors, go to Chrome > Settings > Show advanced settings > Privacy, and un-check the following options:

- Use a prediction service to help complete searches and URLs typed in the address bar or the app launcher search box
- Use a prediction service to load pages more quickly.

# Capture Operations

You use the Lightspeed Live Capture web service GET operations to control capture operations and monitor them.

- Web Service Response Elements
- Get-Statistics
- Start
- Modify
- MarkOut | MarkIn
- EditOut | EditIn
- Message
- Stop
- Status

**Note:** Each operation includes a brief description, including the format of the operation, an example, and all required and optional parameters in a table. Finally, the response is presented with an example.

# Web Service Response Elements

Some XML elements are returned in every response. Other elements are returned only for specific operations, as noted.

## Standard Response Elements

These elements are returned in every response. Some elements are returned only for specific operations, as noted.

| Response Element | Description |
|---|---|
| Access-Control-Allow-Origin | For use only by Telestream. |
| End | The anticipated ending timecode of a clip. For example: `02:23:50;00`. |
| EngineTime | The current Lightspeed Live Capture timecode value. For example: `12:34:56:00`.<br>**Note:** *EngineTime* is only returned when a job is queued and awaiting capture or is actively capturing. |
| MarkIn | The actual starting timecode of a clip (returned from Status and Stop operations with a UUID). |
| MarkOut | The actual ending timecode of a clip (returned from Status and Stop operations with UUID).<br>For example: `01:23:45;00`. |
| Name | The name of the job.<br>For example: `LiveInOaklandConcert`. |
| Start | The anticipated starting timecode of a clip. For example: `01:23:45;00`.<br>The timecode type is determined by the configuration of the source input. The timecode can be Free Run, Computer Clock, LTC, or Input Source. |

| Response Element | Description |
|---|---|
| State | Keywords describing the current state of the clip:<br>*Opened*—Currently capturing.<br>*Waiting*—Connected and waiting for jobs to be queued or to start jobs in the queue.<br>*Closed*—Capture is complete.<br>*Cancelled*—The job was canceled before capture.<br>*Failed*—The job failed during capture. Accompanied by the *TransmitError* response. |
| UUID | The unique identifier (GUID) assigned to a clip. For example: `5b1eb65c-3018-a4cf-8134-6e1c16b378a7`. |
| XMLRevision | The XML response revision. For example: `2`. |

## Optional Response Elements

These elements may also be present in the response, depending on the operation and parameters you send:

| Response Element | Description |
|---|---|
| ActionDuration (for Start \| Stop \| Status operations when UUID specified \| Status when UUID specified) | The estimated duration of the capture action, in seconds. Progress divided by Action Duration multiplied by 100 results in the percentage complete. |
| Channels | Integer; the number of audio channels in the clip. |
| EngineState | Keyword; the state of the Lightspeed Live Capture Engine:<br>*Running*—Currently operating normally. |
| Excluding | Boolean; reports if frames are being excluded via the use of MarkOut\|In or EditOut\|In operations. Mark operations are not used on TIFO files.<br>*Excluding* is *false* upon starting an initial recording and after an In operation. *Excluding* is *true* after an Out operation. |
| FPS (for Start \| Stop \| Status operations when UUID specified \| Status when UUID specified) | The frame rate of the media being captured. |
| FrameRate | Real; the clip's frame rate. |
| HorizontalResolution | Integer; the clip's horizontal resolution. |

| Response Element | Description |
|---|---|
| PercentCompleted | Not utilized in this version. |
| Progress (for Start \| Stop \| Status operations when UUID specified \| Status when UUID specified) | Elapsed progress (in seconds) of the associated capture. |
| Error | String; reports a detailed error during a Failed state.<br>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." |
| VerticalResolution | Integer; the clip's vertical resolution |

# Get-Statistics

The *Get-Statistics* operation obtains timecode and deck status information for the current source, in XML format. This operation has no parameters.

This operation has the following format:

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

## Example

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

```
http://10.5.2.1:17000/record/get-statistics
```

## Typical Response

Issuing this *Get-Statistics* operation resulted in the following response:

```
<Response>
  <UUID>724c593b-8da7-4c3b-b667-78d75e16abe1</UUID>
  <PercentCompleted>0</PercentCompleted>
  <Progress>0</Progress>
  <ActionDuration>660.0594</ActionDuration>
  <FPS>29.97002997003</FPS>
  <Start>18:26:12;11@29.97</Start>
  <End>18:37:12;13@29.97</End>
  <MarkIn/>
  <MarkOut/>
  <Excluding>False</Excluding>
  <Name>Bob-New Workflow_LS-SVR - SDI Input 1.1</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>18:25:53;17</EngineTime>
  <XMLRevision>2</XMLRevision>
</Response>
<Statistics>
<Firmware-Version>2016/12/30</Firmware-Version>
<CPU-Usage>0</CPU-Usage>
<Memory>63</Memory>
<Process>8404</Process>
<SDI>4</SDI>
<Resolution>720x512i</Resolution>
<Frame-Rate>29.97</Frame-Rate>
<Bit-Depth>8-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 initiates the recording of a new clip.

You can optionally start recording immediately or specify start and stop times. You can also explicitly name the file and specify an alternate file storage path. You can also control how recording is handled when discontinuous timecodes are detected.

**Note:** A maximum of 16 jobs per workflow can be queued in a clip list for processing. Jobs are automatically removed from the list when complete. Use the Status operation to determine the number of clips currently queued for processing.

This operation has the following formats:

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

or

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

**Note:** If a start time is not specified, the clip begins recording immediately. If an end time is not specified the clip will record for 9 hours. For best results, specify an end time to avoid running out of disk space or design your application to actively monitor and control the recording time.

## Parameters

Timecodes can be specified in either drop frame (for example: `01:00:10;00`) or non-drop frame format (`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).

| Parameter | Description |
|---|---|
| duration (optional) | Timecode; specifies the duration of the clip based on the number of frames captured. Default: 9 hours.<br>For example: `duration=00:30:00:00`.<br>If duration is used, the capture session continues until the file contains a number of frames equal to the duration parameter. This allows for timecode jumps that occur during capture.<br>When used in conjunction with end, the end timecode overrides duration when the specified timecode (or one later in time) is detected.<br>If no duration or end is specified, the clip records for 9 hours.<br>**Error**: Returns an error if timecode contains invalid characters. |

| Parameter | Description |
|---|---|
| end<br>(optional) | Timecode; specifies the clip's *exclusive* end timecode.<br>For example: `end=01:42:35:00`.<br>The end timecode represents the timecode of the frame after the last frame of video.<br>Capture will stop when the specified timecode (or one later in time) is detected. If neither a duration or end timecode is specified, the clip will record indefinitely.<br>**Error**: Returns an error if timecode contains invalid characters. |
| folderPath<br>(optional) | String; specifies the path to a folder where the clip file should be written. The path must end with a \.<br>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.<br>Requires that the Capture action option for Create Output File(s) on Job Start is enabled.<br>For example: `folderPath=[Drive letter]:\[folder path]\ or \\[Share server]\[Share]\[Folder Path]\` |
| name<br>(optional) | String; specifies a name for the clip. If a clip name is not specified, then a random name is generated.<br>**Note**: For this name to be utilized in a Lightspeed Live Capture workflow, you must include the Base Name token in the Primary or Secondary Output's file name creation pattern in the Filename Patter 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.<br>In addition, the Capture action must also have the Create output file(s) at job start option enabled.<br>For example: `name=LiveInOakland_Cam7_16062016_1342_PQ_234`.<br>**Error**: Returns an error if the name contains invalid characters. |
| discontinuity<br>(optional) | Keyword (Abort \| Ignore); specifies how to control recording when timecode discontinuities are observed in the video stream. When set to "Ignore", any timecode discontinuity will be ignored and the capture will continue. When set to "Abort", the capture will stop and the job is terminated when a timecode discontinuity is detected. |

| Parameter | Description |
|---|---|
| rs422delay<br>(optional) | Integer; specifies the RS-422 frame delay when capturing from a VTR deck, adjusting the RS-422 time code by the specified number of frames.<br>For example: `rs422delay=-1`.<br>RS-422 time code delay can be used to compensate for VTR decks requiring adjustments to frame timing. 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 this offset adjustment. If you have a VTR deck that requires this adjustment, use a trial-and-error process to determine the correct offset. Default 0; minimum -3 frames and maximum +3 frames.<br>**Note**: The associated Capture action must also have the *Create output file(s) at job start* option enabled.<br>**Error**: Returns an error if the rs422delay contains invalid characters or values out of range. |
| start<br>(optional) | Timecode; specifies the clip's *inclusive* starting timecode.<br>For example: `start=01:12:35:00`.<br>The start timecode represents the timecode of the first frame. If a start time is not specified, the clip begins recording immediately.<br>**Error**: Returns an error if timecode contains invalid characters. |
| tape<br>(optional) | String; specifies a name for the Tape Name or Reel. The Tape Name/Reel is embedded directly into QuickTime and MXF OP1a files.<br>For example: `tape=LiveInOakland_Cam7.`<br>**Error**: Responds with an error if the tape name contains invalid characters. |

## Example

In this example, recording will start immediately, and record for 10 minutes:

```
http://10.5.2.1:8080/record/start?duration=00:10:00:00
```

## Typical Response

Issuing this *Start* operation with a start timecode, duration, name, folderPath and tape parameters:

```
http://ll-pm:17000/record/
start?Start=01:00:00:00&duration=00:05:13:00&name=myClipName
&folderpath=\\ll-pm\d\temp&tape=myTapeName
```

resulted in the following response:

```
<Response>
  <UUID>724c593b-8da7-4c3b-b667-78d75e16abe1</UUID>
  <PercentCompleted>0</PercentCompleted>
  <Progress>0</Progress>
```

```
 <ActionDuration>660.0594</ActionDuration>
 <FPS>29.97002997003</FPS>
 <Start>18:26:12;11@29.97</Start>
 <End>18:37:12;13@29.97</End>
 <MarkIn/>
 <MarkOut/>
 <Excluding>False</Excluding>
 <Name>Bob-New Workflow_LS-SVR - SDI Input 1.1</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>18:25:53;17</EngineTime>
 <XMLRevision>2</XMLRevision>
</Response>
<Response>
  <UUID>f23049e4-9d6f-4a51-bd79-8821bec7d604</UUID>
  <PercentCompleted>0</PercentCompleted>
  <Progress>0</Progress>
  <ActionDuration>312.979333333333</ActionDuration>
  <FPS>29.97002997003</FPS>
  <Start>01:00:00;00@29.97</Start>
  <End>01:05:13;00@29.97</End>
  <MarkIn></MarkIn>
  <MarkOut>01:05:13;00@29.97</MarkOut>
  <Excluding>False</Excluding>
  <Name>MyClipName.tifo</Name>
  <Path>\\ll-pm\D\temp\MyClipName.tifo</Path>
  <Tape>myTapeName</Tape>
  <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>12:26:52;20</EngineTime>
  <XMLRevision>2</XMLRevision>
```

# Modify

The *Modify* operation is used to modify the start time and/or duration of a scheduled event contained in the clip list.

**Note:** Clips you plan to modify must be in the Waiting state. Using a Modify operation on clips in the Opened (capturing) state cause the capture to immediately stop. To modify clips in the Opened state use a Stop operation with an end parameter.

This operation has the following format:

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

## Parameters

The uuid parameter is required. Either the start or duration parameter must be supplied; both are permitted.

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 for non-drop frame source).

| Parameter | Description |
|---|---|
| uuid<br>(required) | GUID; the unique identifier of the clip you wish to modify.<br>For example: `uuid=21EC2020-3AEA-4069-A2DD-08002B30309D`<br>**Error**: Returns *404 Not Found* if the specified clip is not in the list. |
| start<br>(optional) | Timecode; specifies the clip's new i*nclusive* starting timecode.<br>For example: `start=01:12:35:00`.<br>The start timecode represents the timecode of the first frame to be captured.<br>**Error**: Returns an error if timecode contains invalid characters. |
| duration<br>(optional) | Timecode; specifies the new duration of the clip.<br>For example: `duration=01:12:35:00`.<br>**Error**: Returns an error if timecode contains invalid characters. |

## Example

```
http://LS-SVR:17000/record/modify?uuid=27638f24-2bb1-4ce6-8816-
5a05a5e05897&start=11:05:06:09
```

## Typical Response

Issuing this *Modify* operation with a UUID and start timecode results in the following response:

```
<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>
```

Issuing this *Modify* operation with a UUID and a duration timecode:

```
http://lsl-pm:17000/record/modify?
uuid=27638f24-2bb1-4ce6-8816-5a05a5e05897&duration=00:45:00:00
```

generated the following response:

```
<Response>
  <UUID>27638f24-2bb1-4ce6-8816-5a05a5e05897</UUID>
  <PercentCompleted>0</PercentCompleted>
  <Progress>0</Progress>
  <ActionDuration>3239.9733066</ActionDuration>
  <FPS>29.97002997003</FPS>
  <Start></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>11:04:41;15</EngineTime>
  <XMLRevision>2</XMLRevision>
</Response>
```

# MarkOut | MarkIn

**Note:** Supported for TIFO format only. 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.

All TIFO frames after a *MarkOut* is executed will be marked with an OUT metadata tag. The TIFO decoder included in a Vantage workflow can ignore frames marked as OUT and prevent them from being passed to a compressor or other downstream processes.

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.

These two operations have the following formats:

```
http://<host>:<port>/record/markout | markin
```

or

```
http://<host>:<port>/record/mark[out | markin]
[?parameter=value[&parameter=value]]
```

## Parameters

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

| Parameter | Description |
|---|---|
| uuid (required) | GUID; the unique identifier of the clip on which to set the MarkOut/In point.<br>For example: `uuid=21EC2020-3AEA-4069-A2DD-08002B30309D`<br>Returns *404 Not Found* if the specified clip is not in the list. |
| timecode (optional) | Timecode; specifies the timecode in the future for a MarkOut/In point in the clip.<br>For example: `timecode=01:12:35:00.`<br>If a timecode is not specified the MarkOut/In point will be issued immediately.<br>MarkOut timecode is INCLUSIVE; the frame will be marked OUT.<br>MarkIn timecode is EXCLUSIVE; the frame will be marked IN.<br>**Error**: Returns an error if timecode contains invalid characters. |

## Example

In this example, all frames in the recording will be tagged as OUT, beginning at timecode 01:00:10:00:

```
http://10.5.2.1:8080/record/markout?timecode=01:00:10:00
```

## Typical Response

Issuing these *MarkIn* and *MarkOut* operations:

http://`lsl-pm:17000/record/markin (or markout)`

resulted in this response:

```
<Response>
  <UUID>27638f24-2bb1-4ce6-8816-5a05a5e05897</UUID>
  <State>Opened</State>
  <Access-Control-Allow-Origin>*</Access-Control-Allow-Origin>
  <EngineState>Running</EngineState>
  <EngineTime>11:06:54;27</EngineTime>
  <XMLRevision>2</XMLRevision>
</Response>
```

# EditOut | EditIn

*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 captured file; issuing an EditIn operation returns to adding source frames to be written to the capture file.

If Filler *is checked* in the Capture action, issuing an *EditOut* enable frames from the Filler file to be written to the captured file. Issuing an *EditIn* returns to adding source frames to be written to the capture 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.

These operations have the following formats:

```
http://<host>:<port>/record/editin | editout
```

When used without a UUID or timecode; all clips in the list are updated.

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

when used with a UUID and/or timecode.

## Parameters

Timecodes can be specified in drop frame (for example: 01:00:10;00) or non-drop frame (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).

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

## Example

In this example, Filler has not been checked in the workflow's Capture action. Thus, the media is not being written to disk at this moment. Also, because there is no uuid parameter in this operation, all clips in the list are updated:

```
http://10.5.2.1:8080/record/editin|editout?timecode=01:00:10:00
```

## Typical Response

Issuing these *EditIn* and *EditOut* operations:

```
http://lsl-pm:17000/record/editin (or editout)
```

resulted in this response:

```
<Response>
  <UUID>27638f24-2bb1-4ce6-8816-5a05a5e05897</UUID>
  <State>Opened</State>
  <Access-Control-Allow-Origin>*</Access-Control-Allow-Origin>
  <EngineState>Running</EngineState>
  <EngineTime>11:06:54;27</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) | String, reserved numeric strings identified by Sony for 9-pin device operations.<br>For example: `request=2000`<br>Typical Sony operations (supported by most VTRs) include:<br>• Stop = 2000<br>• Play = 2001<br>• Fast Forward = 2010<br>• Step Forward = 2014<br>• Rewind = 2020<br>• Step Backward = 2024<br>• Cue with DATA (Go To) = 2431[TC DATA]<br>- TC = 09:59:59:00; TC DATA = 00595909<br>Reference your VTR guide for supported operations. |

## Example

In this example, the lsl-pm server is contacted on port 17000, issuing a Sony VTR Stop operation:

```
http://lsl-pm:17000/record/Message?request=2000
```

## Typical Response

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

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

```
http://LS-SVR:17000/record/message?request=2001
```

resulted in the following response:

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

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

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

```
http://lsl-pm:17000/record/message?request=612002
```

The response is:

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

The response string 7220000193. It 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 recording and/or removes a clip from the active list.

This operation has the following format:

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

## Parameters

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).

| Parameter | Description |
|---|---|
| uuid<br>(required) | GUID; the unique identifier of the clip to stop.<br>For example: `uuid=21EC2020-3AEA-4069-A2DD-08002B30309D`<br>When this operation is issued the response contains all information for the associated clip. If a clip isn't specified in the request, the response contains the UUID of each clip in the list.<br>**Error**: Returns *404 Not Found* if the clip is not in the list. |
| end<br>(optional) | Timecode; specifies the clip's *exclusive* end timecode.<br>For example: `end=01:42:35:00`.<br>The end timecode represents the timecode of the frame after the last frame of video.<br>If an end time is not specified the clip will stop recording immediately. This parameter has no affect if the duration parameter was used within the job's start operation.<br>**Error**: Returns *400 Bad Request* if the end time results in a clip with a duration longer than initially specified.<br>**Note:** Only change the end timecode on files captured without an initial duration or when capturing QuickTime Closed or MXF OP1a Closed container formats. |

## Example

```
http://LS-SVR:17000/record/stop
?uuid=724c593b-8da7-4c3b-b667-78d75e16abe1
```

## Typical Response

Issuing this *Stop* operation resulted in the following response:

```
<Response>
  <UUID>724c593b-8da7-4c3b-b667-78d75e16abe1</UUID>
  <PercentCompleted>0</PercentCompleted>
  <Progress>111.244458333333</Progress>
  <ActionDuration>660.0594</ActionDuration>
  <FPS>29.97002997003</FPS>
  <Start>18:26:12;11@29.97</Start>
  <End>18:37:12;13@29.97</End>
  <MarkIn>18:26:12;11@29.97</MarkIn>
  <MarkOut/>
  <Excluding>False</Excluding>
  <Name>Bob-New Workflow_LS-SVR - SDI Input 1.1</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>18:28:03;29</EngineTime>
  <XMLRevision>2</XMLRevision>
</Response>
```

# Status

The *Status* operation requests status information for a clip. When you do not specify a GUID, the web service returns the list of recordings currently in the list. You can process the list to extract a GUID and obtain status on any recording you want.

You can also obtain the GUID of a recording from the start operation.

This operation has the following formats:

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

or

```
http://<host>:<port>/record/status
?uuid=21EC2020-3AEA-4069-A2DD-08002B30309D
```

## Parameters

| Parameter | Description |
|---|---|
| uuid<br>(optional) | GUID; the unique identifier of the clip to obtain status information from.<br>For example: `uuid=21EC2020-3AEA-4069-A2DD-08002B30309D`<br>When this operation is issued with a GUID, the response will contain all status details for the associated clip.<br>If a clip GUID is not specified, ALL GUIDs of all clips in the clip list are returned and the response will include general status information.<br>**Error**: Returns *404 Not Found* if the clip is not in the list. |

## Example

```
http://LS-SVR:17000/record/status
?uuid=724c593b-8da7-4c3b-b667-78d75e16abe1
```

## Typical Response

Issuing this *Status* operation resulted in the following response:

```
<Response>
  <UUID>724c593b-8da7-4c3b-b667-78d75e16abe1</UUID>
  <PercentCompleted>0</PercentCompleted>
  <Progress>62.2288333333333</Progress>
  <ActionDuration>660.0594</ActionDuration>
  <FPS>29.97002997003</FPS>
  <Start>18:26:12;11@29.97</Start>
  <End>18:37:12;13@29.97</End>
  <MarkIn>18:26:12;11@29.97</MarkIn>
  <MarkOut/>
  <Excluding>False</Excluding>
  <Name>Bob-New Workflow_LS-SVR - SDI Input 1.1</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>18:27:14;24</EngineTime>
  <XMLRevision>2</XMLRevision>
</Response>
```

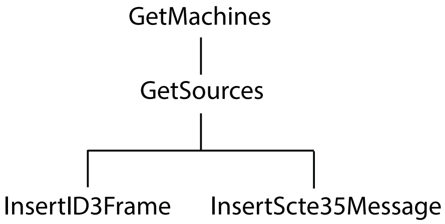
# Source Operations

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 at 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:

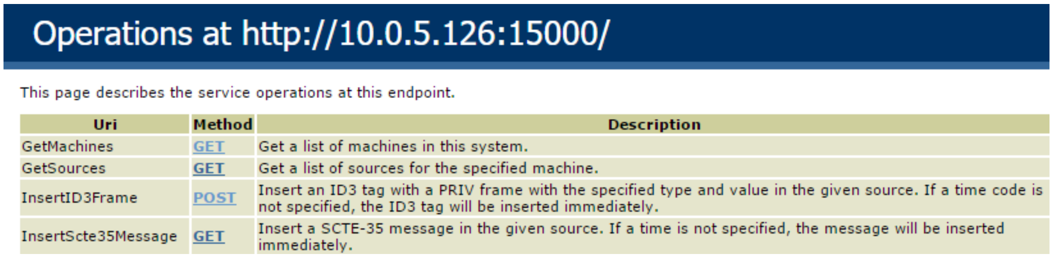
## Displaying the Operations of a Live Source Service

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

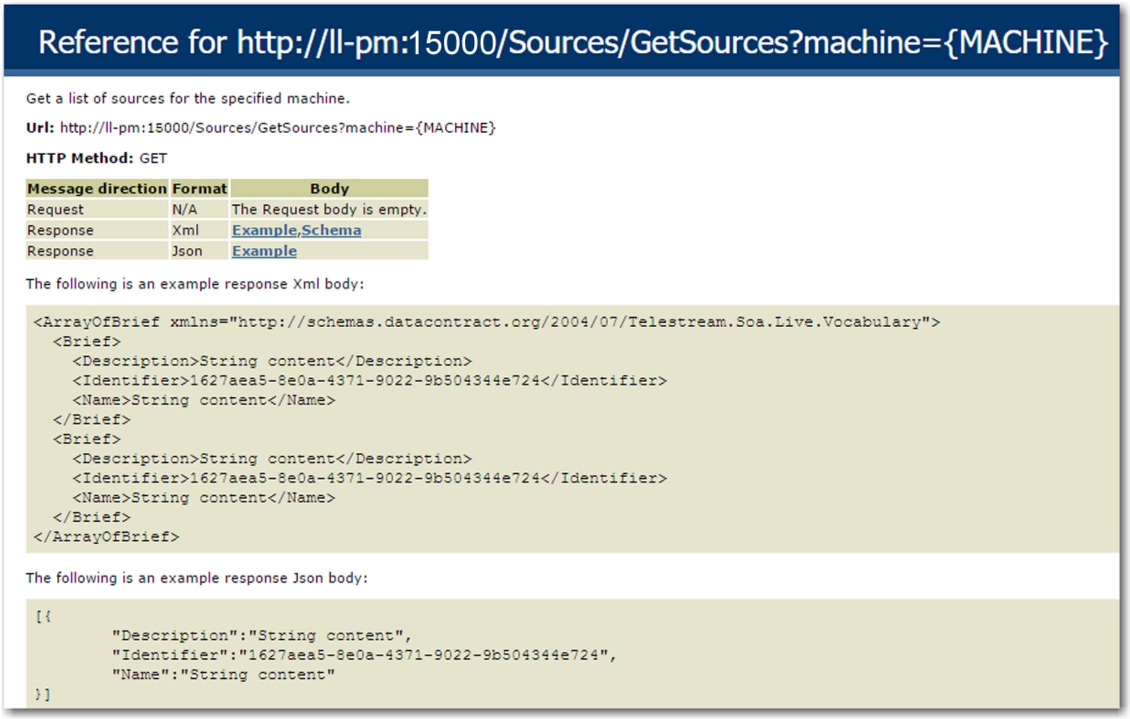
  or

- 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.<br>For example: `source=442082bf-bde8-44b5-9bce-832b6d0fd885` |
| type | String; the type of PRIV frame.<br>For example: `type=com.cisco.streaming.SplicePoint.0` |
| 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.<br>For example: `source=01:03:15:00@29.97` |

## Required Post Body

| | |
|---|---|
| Type Value | String; appropriately-encoded string that is the value for the type key-pair value inserted into the video.<br>This operation supports four MIME types:<br>- text/plain (base 64-encoded binary data)<br>- application/xml or text/xml (un-encoded XML)<br>- 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+MHgxNWQxZTg8L01hcmtlcklEPg0KPFByZXJvbGxITlM+MDwvUHJl
cm9sbEhOUz4NCjxTcGxpY2U+DQo8RXZlbnRJRD4xNDI5OTkyPC9FdmVudElEPg0KPE
91dE9mTmV0d29yaz4wPC9PdXRPZk5ldHdvcms+DQo8UHJvZ3JhbVNwbGljZT4xPC9Q
cm9ncmFtU3BsaWNlPg0KPER1cmF0aW9uPjA8L0R1cmF0aW9uPg0KPFNwbGljZUltbW
VkaWF0ZT4wPC9TcGxpY2VJbW1lZGlhdGU+DQo8VGltZVNwZWNpZmllZD4xPC9UaW1l
U3BlY2lmaWVkPg0KPEF1dG9SZXR1cm4+MDwvQXV0b1JldHVybj4NCjwvU3BsaWNlPg
0KPFByb2dyYW0+DQo8UHJvZ3JhbUlEPjEyNDwvUHJvZ3JhbUlEPg0KPEF2YWlsTnVt
PjA8L0F2YWlsTnVtPg0KPEF2YWlsRXhwZWN0ZWQ+MDwvQXZhaWxFeHBlY3RlZD4NCj
wvUHJvZ3JhbT4NCjxUaW1lPg0KPFNwbGljZVRpbWVITlM+MTAxOTE4MjEzMDAwMDA8
L1NwbGljZVRpbWVITlM+DQo8U3BsaWNlRGF0ZVRpbWU+MjAxNy0xMC0yN1QwMDoxNz
oyM1o8L1NwbGljZURhdGVUaW1lPg0KPER1cmF0aW9uSE5TPjA8L0R1cmF0aW9uSE5T
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}
```

## 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.<br>For example: `source=442082bf-bde8-44b5-9bce-832b6d0fd885` |
| 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.<br>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).<br>For example: scte35Message=`"Program Start"`<br>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.<br>For example: `timeCode=01:03:15:00@29.97` |
| eventID (optional) | Integer value; String of a 32-bit unsigned integer value which specifies the ID of the message. Default when not specified: *4294967295.* |

## 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 | 032 | 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"
```