Lightspeed Live Web API Reference

Live Capture V2.1 ComponentPac 7.1.0

Live Stream V2.0

October 2017

233531

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: 233531

Date: October 2017

Contents

Overview

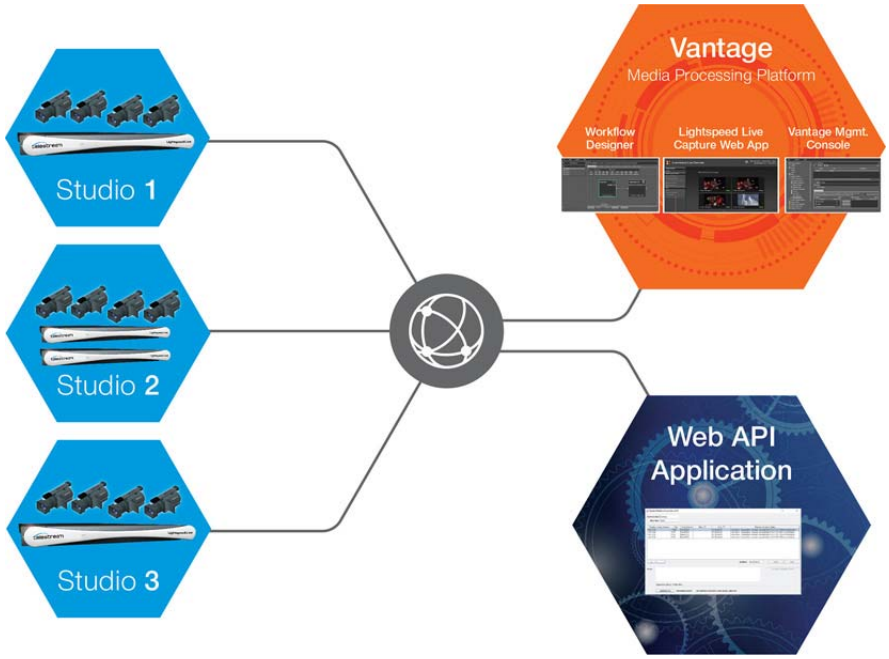
The two Lightspeed Live web APIs—Capture and Stream —enable you to monitor your Lightspeed Live system within a broader, web services-based system or create your own customized monitoring system. You can also create a web services-based system to control streaming and capture beyond the functionality 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
- Operation & Response Formats
- Avoid Using 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.



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 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.)

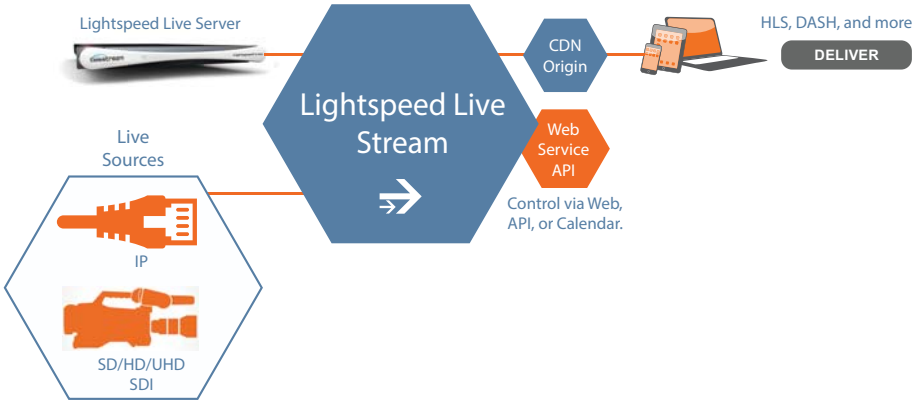
Note: See the Lightspeed Live Guide for details on enabling the web service.

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.



Note: See the Lightspeed Live Guide for details on enabling the web service.

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.

Operation & Response Formats

The Lightspeed LIVE API is 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.

The target Lightspeed Live system responds 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 the parameters passed in.

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

Required and optional response elements are presented in *Live Capture Web Service Operation Responses*; examples are presented in each operation's topic.

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. For example: *localhost* | `LightspeedServer` | `192.168.1.23`.
port	The TCP/IP 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 workflow. (See the Lightspeed Live Guide or the man page for the Capture action.) For Live Stream, the port number (default: 18000) 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

GUIDs (Globally Unique IDentifier)—referred to as UUIDs in Capture operations—are 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, the 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.

Live Capture Web Service Operation Responses

When you send a Capture operation to a Lightspeed Live web service, the web service executes the operation and returns a response appropriate to the operation and the result of the execution.

The response is in XML—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>
```

Topics

Standard Capture Web Service 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`. **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). For example: `01:23:45;00`.
Name	The name of the job. For example: `LiveInOaklandConcert`.
Start	The anticipated starting timecode of a clip. For example: `01:23:45;00`. The timecode type is determined by the configuration of the source input. The timecode can be Free Run, Computer Clock, LTC, or Input Source.
State	Keywords describing the current state of the clip: *Opened*—Currently capturing. *Waiting*—Connected and waiting for jobs to be queued or to start jobs in the queue. *Closed*—Capture is complete. *Cancelled*—The job was canceled before capture. *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 Capture Web Service Response Elements

In addition to the standard response elements, other optional 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: *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. *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.
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. 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

Web Service Error Responses

In the event of an error when an operation attempts to execute, the web service returns an error message. It varies, depending on the operation and parameters you send, and the type of failure.

Capture error responses are in XML format; Stream error responses are strings.

Avoid Using 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 can use the following Lightspeed Live Capture web service GET operations to control capture operations and monitor them:

- 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.

Start

The *Start* operation initiates the recording of a new clip.

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 has the following format:

```
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. For example: `duration=00:30:00:00`. 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. When used in conjunction with end, the end timecode overrides duration when the specified timecode (or one later in time) is detected. If no duration or end is specified, the clip records for 9 hours. **Error**: Returns an error if timecode contains invalid characters.

Parameter	Description
end (optional)	Timecode; specifies the clip's *exclusive* end timecode. For example: `end=01:42:35:00`. The end timecode represents the timecode of the frame after the last frame of video. 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. **Error**: Returns an error if timecode contains invalid characters.
folderPath (optional)	String; specifies the path to a folder where the clip file should be written. The path must end with a \. 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. Requires that the Capture action option for Create Output File(s) on Job Start is enabled. For example: `folderPath=[Drive letter]:\[folder path]\ or \\[Share server]\[Share]\[Folder Path]\`
name (optional)	String; specifies a name for the clip. If a clip name is not specified, then a random name is generated. **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. In addition, the Capture action must also have the Create output file(s) at job start option enabled. For example: `name=LiveInOakland_Cam7_16062016_1342_PQ_234`. **Error**: Returns an error if the name contains invalid characters.

Parameter	Description
rs422delay (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. For example: `rs422delay=-1`. 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. **Note**: The associated Capture action must also have the *Create output file(s) at job start* option enabled. **Error**: Returns an error if the rs422delay contains invalid characters or values out of range.
start (optional)	Timecode; specifies the clip's i*nclusive* starting timecode. For example: `start=01:12:35:00`. The start timecode represents the timecode of the first frame. If a start time is not specified, the clip begins recording immediately. **Error**: Returns an error if timecode contains invalid characters.
tape (optional)	String; specifies a name for the Tape Name or Reel. The Tape Name/Reel is embedded directly into QuickTime and MXF OP1a files. For example: `tape=LiveInOakland_Cam7.` **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 Start Operation Response

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

```
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>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>
</Response>
```

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

```
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.

This has the following format:

```
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. For example: `uuid=21EC2020-3AEA-4069-A2DD-08002B30309D` 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. For example: `timecode=01:12:35:00.` If a timecode is not specified the MarkOut/In point will be issued immediately. MarkOut timecode is INCLUSIVE; the frame will be marked OUT. MarkIn timecode is EXCLUSIVE; the frame will be marked IN. **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 MarkIn/MarkOut Operation 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:

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

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

```
<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 (required)	GUID; the unique identifier of the clip on which to set the EditOut|EditIn point. If not set, all clips are updated. For example: `uuid=21EC2020-3AEA-4069-A2DD-08002B30309D` Returns *404 Not Found* if the specified clip is not in the 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 will be issued immediately. For example: `timecode=01:12:35:00`. EditOut timecode is *inclusive*; the frame will be edited out. EditIn timecode is *exclusive*; the frame will be edited in. **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 EditIn/EditOut Operation 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 the application to control a VTR by sending Sony VTR operations to the VTR 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 has the following format:

```
address:port/record/Message?request=[Sony operation]
```

Parameters

Parameter	Description
request (required)	String, reserved numeric strings identified by Sony for 9-pin device operations. For example: `request=2000` Typical Sony operations (supported by most VTRs) include: • 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 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 Message Operation 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:7500/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 =

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 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 (required)	GUID; the unique identifier of the clip to stop. For example: `uuid=21EC2020-3AEA-4069-A2DD-08002B30309D` 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. **Error**: Returns *404 Not Found* if the clip is not in the list.
end (optional)	Timecode; specifies the clip's *exclusive* end timecode. For example: `end=01:42:35:00`. The end timecode represents the timecode of the frame after the last frame of video. 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. **Error**: Returns *400 Bad Request* if the end time results in a clip with a duration longer than initially specified. **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:7500/record/stop
?uuid=724c593b-8da7-4c3b-b667-78d75e16abe1
```

Typical Stop Operation 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 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 (optional)	GUID; the unique identifier of the clip to obtain status information from. For example: `uuid=21EC2020-3AEA-4069-A2DD-08002B30309D` When this operation is issued with a GUID, the response will contain all status details for the associated clip. 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. **Error**: Returns *404 Not Found* if the clip is not in the list.

Example

```
http://LS-SVR:7500/record/status
?uuid=724c593b-8da7-4c3b-b667-78d75e16abe1
```

Typical Status Operation 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>
```

Stream Operations

You can use the Lightspeed Live Stream web service operations in a program to control and monitor streaming on the Lightstream Live Server. Programs written to control and monitor Lightspeed Live streaming can supplement the general-purpose Lightspeed Live Stream web application provided by Telestream.

The web service operations include both GET and POST operations, and they are organized into functional categories to facilitate easier implementation.

Introduction

The Live Stream interface is implemented over HTTP as a RESTful interface, with most results returned in JSON format. A few operations return thumbnails or XML files intended for import/export.

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

Limits of the API

The Live Stream API is generally intended for control and monitoring. You can not create most Live Stream components or configure them using the API. Before you can control and monitor streams on the Lightstream Live Server, you must first create and configure them using the Lightstream Live web application.

Using Live Stream Groups via the API

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

Ports for Live Stream Server Access

Two ports are used to access Live Stream servers, depending on the method:

- *Port 8089*—the default port for accessing a Live Stream server via the web app.
- *Port 18000*—the default port for accessing the Live Stream server via the API.

Both of these ports can be changed in the web app. Under Settings, you can update the REST API Server port or the App Server port. After changing a port, you must restart the services for the new port number to take effect.

Using Shared vs. Dedicated Components

Two types of Stream components—media sources and output channels—are associated with a port directly on a specific server and thus, must be executed directly on that server. For example, when you are controlling a media source, you must execute the operation using the DNS name or the IP address of the Live Stream server where the source was created.

The other components—programs, encoders, and packages, for example—are not tied to a specific server; they are available to all servers in the group, and they can be shared. Thus, you can address the operation to any server in the group and the operation will be executed properly, regardless of where the component was created and configured.

Live Stream Component Hierarchy

When managing and controlling components in Live Stream, it is helpful to understand the organization and hierarchy of components:

- Machine > Source > SourceTrack
- Machine > Source > TextTrack
- Program > Rendition > Segment > Material
- Encoder > Stream
- Package > Variant
- Channel > CalendarEvent

Whenever you are targeting a specific component, you must first identify each of the superior components in the chain by GUID. For example, if you are operating on a specific segment, you must first obtain the GUID of the program, then the rendition, and finally the segment you are targeting. This sequence of operations is documented in each operation's Operation Sequence topic.

Obtaining Help for Stream Operations

You can obtain information about services in a variety of ways:

Displaying a List of Live Stream Services

To obtain a list of all Live Stream services, execute the *GetServiceEndpoints* operation at the root level: `http://<host>:<port>/GetServiceEndpoints`. *GetServiceEndpoints* returns an array with URI records that display each of the Lightspeed Live Stream service operations by category. For details, see *GetServiceEndpoints*.

Displaying the Operations of a Specific Live Stream Service

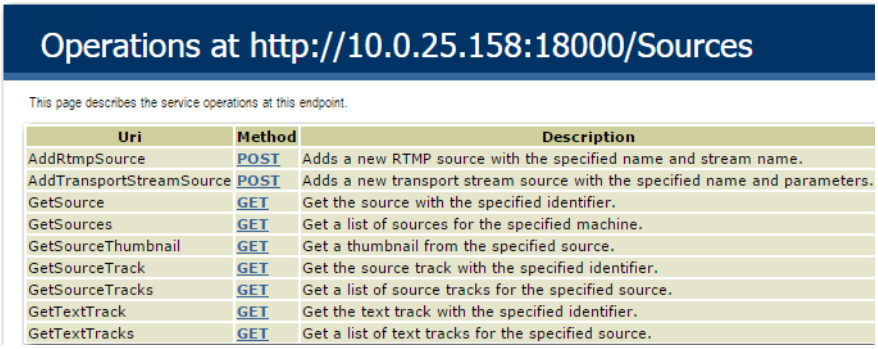
To list all of the operations in a specific service, execute Help in the service you're interested in.

This form of Help has the following format:
`http://<host>:<port>/<service category>/help.`

Keywords: Sources | Programs | Encoders | Packages | Channels.

For example: `http://10.0.25.158:18000/Sources/help`

Help returns a web page listing all operations for the end point category you specified:



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

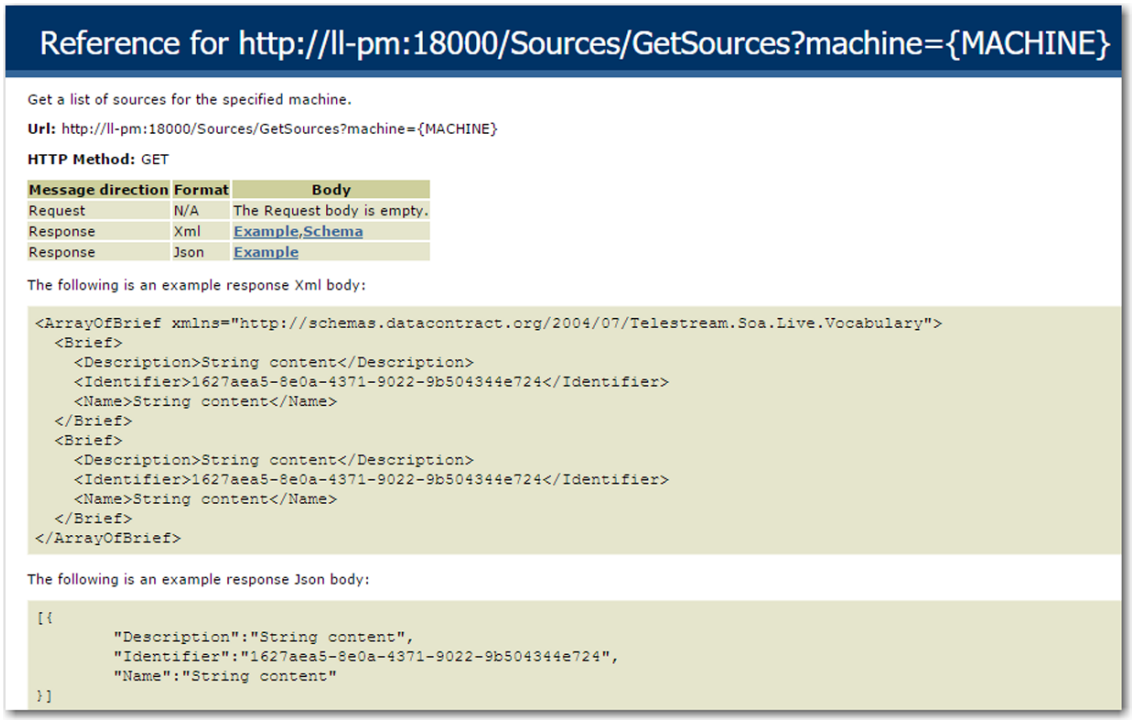
Displaying Operation Details

To display details about a specific operation and view the syntax of the JSON response, execute the operation in question in the <service>/help/operations directory, without providing any parameters.

This form of Help has the following format:
`http://<host>:<port>/<service category>/help/operations/<Operation>`.

For example:
`http://10.0.25.158:18000/Sources/help/operations/GetSources`

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



Reference for http://ll-pm:18000/Sources/GetSources?machine={MACHINE}

Get a list of sources for the specified machine.

Url: http://ll-pm:18000/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 but a few operations, where XML, ZIP, and JPEG data is returned.

System Operations

Use the following operations to obtain system-level information, including a list of servers in the system, Live Stream services, system settings, and logs.

- GetMachines
- GetServiceEndpoints
- GetSystemSettings
- GetSystemLogs.zip

Note: All System operations are located at the root `http://<host>:<port>/`. To display help for system operations, enter `http://<host>:<port>/help`

GetMachines

The purpose of *GetMachines* is to identify the Live Stream server (or in the case of a group, all of the Live Stream servers in the group) by GUID. This is a pre-requisite for executing many of the Sources operations, plus *GetMachineStatistics*.

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 Stream server or any Live Stream server in the group (no parameters).

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 for the Live Stream server (or, in the case of a group, each Live Stream server), with its GUID and other details.

Example

`http://10.9.9.9:18000/GetMachines`

Typical Response

In this response, three server's 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"
  }
]
```

GetServiceEndpoints

The purpose of this GET operation is to obtain a list of all Live Stream services, which is executed at the root level. There are no parameters.

GetServiceEndpoints has the following format:

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

Operation Sequence

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

Results

Upon success, *GetServiceEndpoints* returns an array of records with URIs that display each of the Lightspeed Live Stream service operations by category. The Identifier GUIDs are all zeros, because they are unused.

Example

`http://10.9.9.9:18000/GetServiceEndpoints`

Typical Response

In this response, the Live Stream system responds with a list of endpoints.

```
[
  {
  "Description":"Service for source-related operations",
  "Identifier":"00000000-0000-0000-0000-000000000000",
  "Name":"http:\/\/ll-pm:18000\/Sources\/Help"
  },
  {
  "Description":"Service for program-related operations",
  "Identifier":"00000000-0000-0000-0000-000000000000",
  "Name":"http:\/\/ll-pm:18000\/Programs\/Help"
  },
  {
  "Description":"Service for encoder-related operations",
  "Identifier":"00000000-0000-0000-0000-000000000000",
  "Name":"http:\/\/ll-pm:18000\/Encoders\/Help"
  },
  {
  "Description":"Service for package-related operations",
  "Identifier":"00000000-0000-0000-0000-000000000000",
  "Name":"http:\/\/ll-pm:18000\/Packages\/Help"
  },
  {
    "Description":"Service for channel-related operations",
  "Identifier":"00000000-0000-0000-0000-000000000000",
  "Name":"http:\/\/ll-pm:18000\/Channels\/Help"
  }
]
```

GetSystemSettings

The purpose of this GET operation is to export the entire Live Stream system configuration in XML format. This XML can be used for a variety of purposes, including importing directly into another system, using the Import button in the Live Stream web app. There are no parameters.

GetSystemSettings has the following format:

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

Results

Upon success, *GetSystemSettings* returns the system configuration of the target system in XML format.

Example

`http://10.9.9.9:18000/GetSystemSettings`

Typical Response

In this response, this multi-machine XML is returned (shown here truncated for brevity).

```
<Domain>
  <Configuration>...</Configuration>
  <User>...</User>
  <Machine d1p1:identifier="f1c541f6-d003-41a0-b627-e6fa0add9c95"
d1p1:name="LS-SVR" d1p1:leftaligncheckboxes="false">
    <d1p1:Parameter>...</d1p1:Parameter>
...
    <Source d1p1:identifier="f7338552-a221-4853-a8cb-60de6371ae1b"
d1p1:name="LS-SVR - SDI Input 4" d1p1:description=""
d1p1:leftaligncheckboxes="false">
      <d1p1:Parameter type="boolean" identifier="a1962c94-8a40-
4d6f-a192-dec61e449bcb" name="10-Bit Video" description="Causes
the source to capture 10 bit video rather than 8 bit"
enabled="true" enabledinvariants="false" disableable="false"
browsable="true" optionseditable="false" row="0" column="0"
columnspan="1">...</d1p1:Parameter>
...
      <SourceType>sdi</SourceType>
      <Statistics>...</Statistics>
      <Tracks>...</Tracks>
      <TextTracks/>
    </Source>
  </Machine>
</Domain>
```

GetSystemLogs.zip

The purpose of this GET operation is to obtain the system logs in a zip file. The zip extension to the command identifies the file type returned. It also provides the downloaded file with the correct extension for saving in your default downloads directory (typically, C:\Users\<UserName>\Downloads) when using a browser.

This operation has no parameters.

GetSystemLogs.zip has the following format:

```
http://<host>:<port>/Sources/GetSystemLogs.zip
```

Results

Upon success, *GetSystemLogs.zip* returns a zip file of SNMP logs. The downloaded zip file contains two identical inner zip files. (This arrangement allows the potential for adding other logs or reports in the future.)

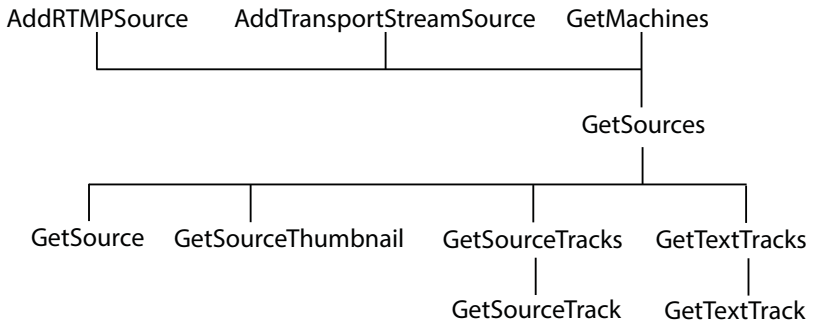
Sources Operations

Use the following operations to obtain a list of servers in a group, add sources, and operate on sources and their components.

- AddRtmpSource
- AddTransportStreamSource
- GetSources
- GetSource
- GetSourceTracks
- GetSourceTrack
- GetTextTracks
- GetTextTrack
- GetSourceThumbnail

Note: All Sources operations start with `http://<host>:<port>/Sources/`.

In this diagram, the operations are organized hierarchically, by machine GUID requirement. The operations to add sources do not require GUIDs. All source operations require a machine GUID.



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

AddRtmpSource

This POST operation adds a new RTMP source to the target Live Stream server.

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

AddRtmpSource has the following format:

```
http://<host>:<port>/Sources/AddRtmpSource?
sourceName={SOURCE NAME}&streamName={STREAM NAME}
```

Operation Sequence

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

Required Parameters

Parameter	Description
sourceName	String; the practical name of the source, which displays in the Sources panel and the Name field of the Configure Source window. For example: `sourceName=Boats Of Port Townsend`
streamName	String; the practical name of the RTMP stream you want associated with this source. This name is specified along with the IP address in the RTMP stream generator program, and the two must match. For example: `streamName=BoatsOfPortTownsendStream`

Results

Upon success, *AddRtmpSource* adds the specified source to the Live Stream server and returns a record with the assigned GUID, indicating the source that was added. If you add an RTMP source twice, the error "Failed to create RTMP source" is returned.

Example

```
http://10.9.9.9:18000/Sources/AddRtmpSource?
sourceName=Boats Of Port Townsend
&streamName=BoatsOfPortTownsendStream
```

Typical Response

A new RTMP source named *Boats Of Port Townsend* was added to this server. The server generated a GUID for the new source, and also returns the name you supplied.

```
{
  "Description": "",
  "Identifier": "e5b1f7eb-1ca0-429e-b1e8-4d2bacf4900b",
  "Name": "Boats Of Port Townsend"
}
```

AddTransportStreamSource

The purpose of this POST operation is to add a new Transport Stream source to the target Live Stream server, with the specified name and settings.

Note: Sources are defined specifically for a hardware port on the server where they are created. Thus the host that you specify for this operation must be that of the server where the Source was added.

AddTransportStreamSource has the following format:

```
http://<host>:<port>/Sources/AddTransportStreamSource
?sourceName={SOURCE NAME}&multicastIP={MULTICASTIP}
&localIP={LOCALIP}&portNumber={PORTNUMBER}
&programNumber={PROGRAMNUMBER}&sourceFilterIP={SOURCEFILTERIP}
```

AddTransportStreamSource adds the specified source to the Live Stream server and returns a component indicating the source that was added.

Operation Sequence

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

Parameters

Parameter	Description
sourceName	String; practical name of the source, displayed in the Name field of the Configure Source window. For example: sourceName=`Hiking Wind Ridge`
multicastIP (optional)	IP address; specifies the optional multicast group IP address being used by the Transport Stream generator program to broadcast.
localIP	IP address; specifies the IP address of the NIC card in the server where you are listening for the Transport Stream.
portNumber	Port number; specifies the port number that the Transport Stream generator program is broadcasting this Transport Stream on.
programNumber	Integer; specifies the program number in the Transport Stream that you are receiving.
sourceFilterIP (optional)	IP address; specifies the IP address of a Transport Stream when originating from a system that is broadcasting using multicast.

Results

Upon success, *AddTransportStreamSource* adds the specified source to the Live Stream server and returns a component, indicating the source that was added.

Example

```
http://10.0.25.162:18000/Sources/AddTransportStreamSource
?sourceName=Hiking Wind Ridge
&multicastIP=239.1.2.3&localIP=10.0.25.162
&portNumber=1234&programNumber=1
```

Typical Response

In this response, a new Transport Stream source named *Hiking Wind Ridge* was added to this Live Stream server. The Live Stream server generated a GUID for the new source, and also returns the name you supplied.

```
{
  "Description": "",
  "Identifier": "9278c2cd-cc10-4df5-8cb4-018451ef6793",
  "Name": "Hiking Wind Ridge"
}
```

GetSources

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

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)

Parameters

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

Results

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

Example

```
http://10.9.9.9:18000/Sources/GetSources
?machine=1129625b-0d7d-490a-aa3f-315214a0b6f2
```

Typical Response

In this response, all of the Sources that are operational—the four default SDI sources, plus a Transport Stream source—on the target server are listed along with their Identifier and Name elements, 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":"7bb71ac8-f5ba-496f-826d-558b38721ae2",
    "Name":"QA-VL-LIVE-9 - SDI Input 2"
  },
  {
    "Description":null,
    "Identifier":"8a7bc656-6704-41a2-8161-f4d0d5a5f8de",
    "Name":"Test1"
  },
  {
```

```
    "Description":null,
    "Identifier":"b2432c41-5b8f-4fc2-8980-fde99709bef9",
    "Name":"QA-VL-LIVE-9 - SDI Input 1"
  },
  {
    "Description":null,
    "Identifier":"edec5e10-177c-4abb-a274-c9fb82393412",
    "Name":"QA-VL-LIVE-9 - SDI Input 4"
  },
  {
    "Description":null,
    "Identifier":"9278c2cd-cc10-4df5-8cb4-018451ef6793",
    "Name":"Hiking Wind Ridge"
  }
]
```

GetSource

The purpose of this GET operation is to obtain the details of a specific source on the target Live Stream server.

GetSource has the following format:

```
http://<host>:<port>/Sources/GetSource?identifier={SOURCE GUID}
```

Operation Sequence

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

GetMachines (machine GUID) >*GetSources* (source GUID)

Parameters

Parameter	Description
identifier	GUID; string that identifies this source.

Results

On success, *GetSource* returns a record with details about the target source. The record has these key/value pairs:

- Name/Value Pair Details—Process, Resolution, Frame Rate, Bit Depth, Audio, Captions, Deck State, Deck Mode, and Time Code
- Text Tracks—One Brief per text track; Identifier and name
- Tracks—One Brief per track; Identifier and name
- Type—SDI, Slate, FileLoop, etc.

If no source exists with the specified GUID, an error is returned.

Example

```
http://10.9.9.9:18000/Sources/GetSource
?identifier=9278c2cd-cc10-4df5-8cb4-018451ef6793
```

Typical Response

This is a typical response from *GetSource*; a record with key/pair values defining the target source, plus the tracks in the source.

Note: The firmware version and process number are not displayed in the web app unless it is operating in Advanced mode.

```
{
  "Description":null,
  "Identifier":"9278c2cd-cc10-4df5-8cb4-018451ef6793",
  "Name":"Test2",
```

```
  "Details":[
    {
      "Name":"CPU Usage",
      "Value":"1"
    },
    {
      "Name":"Memory",
      "Value":"444"
    },
    {
      "Name":"Process",
      "Value":"38908"
    },
    {
      "Name":"Resolution",
      "Value":"1920x1080i"
    },
    {
      "Name":"Frame Rate",
      "Value":"29.97"
    },
    {
      "Name":"Bit Depth",
      "Value":"10-Bit"
    },
    {
      "Name":"Audio",
      "Value":"2"
    },
    {
      "Name":"Captions",
      "Value":"No"
    },
    {
      "Name":"Time Code (Free Run)",
      "Value":"00:00:53;01"
    }
  ],
  "TextTracks":{
    "Briefs":[
    ]
  },
  "Tracks":{
    "Briefs":[
      {
        "Description":null,
        "Identifier":"ca547d60-3963-4127-8045-640b31aec313",
        "Name":"Stereo 1"
      }
    ]
  },
  "Type":4
}
```

GetSourceTracks

The purpose of this GET operation is to obtain a list of audio tracks for the target source and return them in a list for further use.

GetSourceTracks has the following format:

```
http://<host>:<port>/Sources/GetSourceTracks?source={SOURCE GUID}
```

Operation Sequence

Execute the following operations 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.

Results

On success, *GetSourceTracks* returns a record for each audio track associated with the source.

If no source tracks are present, the array is returned empty ([]).

Example

```
http://10.9.9.9:18000/Sources/GetSourceTracks
?source=dcc037b7-8d32-4159-978a-f9353ec6fa3f
```

If there were more than one audio track associated with this source, there would be a set of records returned; one for each track.

Typical Response

In this response, the target source has one track, named *Stereo 1*:

```
[
  {
    "Description":null,
    "Identifier":"ca547d60-3963-4127-8045-640b31aec313",
    "Name":"Stereo 1"
  }
]
```

GetSourceTrack

The purpose of this GET operation is to obtain details about a specific audio track.

GetSourceTrack has the following format:

```
http://<host>:<port>/Sources/GetSourceTrack
?identifier={TRACK GUID}
```

In addition to Identifier and Name, the track record has these relevant elements:

- ChannelConfiguration—the string identifying the track
- Details—a set of key/value pairs, relevant to the type of audio track.

Operation Sequence

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

GetMachines (machine GUID) > *GetSources* (source GUID) > *GetSourceTracks* (track GUID)

Parameters

Parameter	Description
identifier	GUID; string that identifies a specific audio track.

Results

On success, *GetSourceTrack* returns a record with details of key/value pairs about the target track.

If no SourceTrack exists with the specified GUID, an error is returned.

Example

```
http://10.9.9.9:18000/Sources/GetSourceTrack
?identifier=2da934d3-54f9-4880-8e85-b28851355d61
```

Typical Response

In this response, the target track is Stereo 1. Each track has a set of key/value pairs.

```
{
  "Description":null,
  "Identifier":"2da934d3-54f9-4880-8e85-b28851355d61",
  "Name":"Stereo 1",
  "ChannelConfiguration":2,
  "Details":[
    {
      "Name":"Language",
      "Value":"English"
    },
    {
```

```
      "Name":"Audio Service",
      "Value":"Primary"
    },
    {
      "Name":"Default",
      "Value":"False"
    },
    {
      "Name":"Left Channel",
    "Value":"1"
    },
    {
      "Name":"Right Channel",
      "Value":"2"
    }
  ]
}
```

GetTextTracks

The purpose of this GET operation is to obtain a list of 608 or 708 text (caption) tracks that are present in the target source and return them in a list for further use.

GetTextTracks has the following format:

```
http://<host>:<port>/Sources/GetTextTracks?source={SOURCE GUID}
```

Operation Sequence

Execute the following operations 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.

Results

On success, *GetTextTracks* returns an array; one record for each text track in the source.

If no text tracks are present, the array is returned empty ([]).

Example

```
http://10.9.9.9:18000/Sources/GetTextTracks
?source=2da934d3-54f9-4880-8e85-b28851355d61
```

Typical Response

In this response, the target source has two text tracks. Each has a GUID in the Identifier, plus a Name.

```
[
  {
    "Description":null,
    "Identifier": "ef1752a6-f4fa-49e3-a779-54373f31cb60",
    "Name": "C608English"
  }
  {
    "Description":null,
    "Identifier": "ef1752a6-f4fa-49e3-a779-54373f31cb60",
    "Name": "C708English"
  }
]
```

GetTextTrack

The purpose of this GET operation is to obtain details about a specific text (608 or 708 caption) track.

GetTextTrack has the following format:

```
http://<host>:<port>/Sources/GetTextTrack?identifier={TRACK GUID}
```

Operation Sequence

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

GetMachines (machine GUID) > *GetSources* (source GUID) > *GetTextTracks* (track GUID)

Parameters

Parameter	Description
identifier	GUID; string that identifies a specific text track.

Results

On success, *GetTextTrack* returns a record, with details about the target track.

If no text track exists with the specified GUID, an error is returned.

Example

```
http://10.9.9.9:18000/Sources/GetTextTrack
?identifier=6f79c567-d883-4548-81df-1faeeefe3bf6
```

Typical Response

In this response, the target track is C608English.

```
{
  "Description":null,
  "Identifier":"6f79c567-d883-4548-81df-1faeeefe3bf6",
  "Name":"C608English",
  "Details":[
    {
      "Name":"Language",
      "Value":"English"
    },
    {
      "Name":"CC Index",
      "Value":"1"
    }
  ],
  "TextConfiguration":0
}
```

GetSourceThumbnail

The purpose of this GET operation is to obtain a thumbnail from the target source. The operation returns a JPEG, which you can render or save as a file.

GetSourceThumbnail has the following format:

```
http://<host>:<port>/Sources/GetSourceThumbnail
?source={SOURCE GUID}
```

Operation Sequence

Execute the following operations 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.

Results

On success, *GetSourceThumbnail* returns a JPEG. If the source you are targeting does not have a thumbnail, an HTTP 404 error is returned.

Example

```
http://10.9.9.9:18000/Sources/GetSourceThumbnail
?source=27602854-35a6-4f0c-827c-ebd7ac5d40a9
```

Programs Operations

A program defines what media should be encoded. Programs are comprised of renditions, which are comprised of segments. Segments are comprised of materials.

Note: Segments—in the context of Live Stream—are the collection of material that should be encoded at any given time in a rendition. This is not a segment in a package which refers to an atomic unit of ABR content which players might play as a result of dynamically-changing bandwidth.

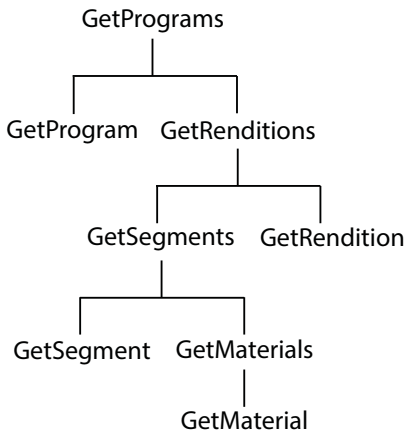
These operations enable you to identify all of the programs in a Live Stream server and target their components: renditions, segments, and materials, successively.

These topics are organized in the order they are often used:

- GetPrograms
- GetProgram
- GetRenditions
- GetRendition
- GetSegments
- GetSegment
- GetMaterials
- GetMaterial

Note: All Programs operations start with `http://<host>:<port>/Programs/`.

In this diagram, the operations are organized hierarchically, by program GUID requirement.



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

GetPrograms

The purpose of this GET operation is to obtain a list of all of the programs that have been added to the target Live Stream system. There are no parameters.

GetPrograms has the following format:

`http://<host>:<port>/Programs/GetPrograms`

Operation Sequence

No operations must be executed before you can execute this operation.

Results

On success, *GetPrograms* returns an array with a record for each program in the system.

If no Programs are present, the array is returned empty ([]).

Example

`http://10.9.9.9:18000/Programs/GetPrograms`

Typical Response

In this response, the array of two records indicates there are two programs on this Live Stream server. Each program has a GUID in the Identifier, and a Name.


```
[
  {
    "Description":null,
    "Identifier":"8bd029bb-c5e3-4c20-af98-48df76f59380",
    "Name":"Trigger Program"
  },
  {
    "Description":null,
    "Identifier":"92aa9eb9-a1fb-4aab-b920-c3259895edb4",
    "Name":"Basic Program"
  }
]
```

GetProgram

The purpose of this GET operation is to obtain detailed information about a specific program.

GetProgram has the following format:

```
http://<host>:<port>/Programs/GetProgram?identifier={program GUID}
```

Operation Sequence

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

GetPrograms (program GUID)

Parameters

Parameter	Description
identifier	GUID; string that identifies a specific program.

Results

On success, *GetProgram* returns a record with details about the program, plus an array of records; one for each rendition in the target program.

If no program exists with the specified GUID, an error is returned.

Example

```
http://10.9.9.9:18000/Programs/GetProgram
?identifier=8bd029bb-c5e3-4c20-af98-48df76f59380
```

Typical Response

In this response, the target program has one Rendition. Each rendition (in the Briefs array) has a GUID Identifier you can use to query it.

```
{
  "Description":null,
  "Identifier":"8bd029bb-c5e3-4c20-af98-48df76f59380",
  "Name":"Trigger Program",
  "Renditions":{
    "Briefs":[
      {
        "Description":null,
        "Identifier":"17b2c3f9-b4ef-401f-bc6d-f01ab6311f84",
        "Name":"Trigger Rendition"
      }
    ]
  },
  "Resolution":"1920 x 1080"
}
```

GetRenditions

The purpose of this GET operation is to obtain a list of Renditions that comprise a specific program.

GetRenditions has the following format:

```
http://<host>:<port>/Programs/GetRenditions?program={PROGRAM GUID}
```

Operation Sequence

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

GetPrograms (program GUID)

Parameters

Parameter	Description
program	GUID; string that identifies a specific program.

Results

On success, *GetRenditions* returns an array, with one record for each rendition of the program.

If no Renditions are present, the array is returned empty ([]).

Example

```
http://10.9.9.9:18000/programs/GetRenditions
?program=7fbb4998-f82a-44da-8d6c-47344b47c10b
```

Typical Response

In this response, there is one rendition, identified by Identifier and Name, which is presented in the web app.

```
[
  {
    "Description":null,
    "Identifier":"17b2c3f9-b4ef-401f-bc6d-f01ab6311f84",
    "Name":"Trigger Rendition"
  }
]
```

GetRendition

The purpose of this GET operation is to obtain details about a specific rendition.

GetRendition has the following format:

```
http://<host>:<port>/Programs/GetRendition
?identifier={RENDITION GUID}
```

Operation Sequence

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

GetPrograms (program GUID) > *GetRenditions* (rendition GUID)

Parameters

Parameter	Description
source	GUID; string that identifies a specific rendition.

Results

On success, *GetRendition* returns a record, with details about the target rendition and each of its segments.

If no rendition exists with the specified GUID, an error is returned.

Example

```
http://10.9.9.9:18000/Programs/GetRendition
?identifier=17b2c3f9-b4ef-401f-bc6d-f01ab6311f84
```

Typical Response

In this response, the target rendition is English Stereo. It has several settings, in the Details, plus a list of Segments and their GUIDs and Name. The rendition Type is also listed.

```
{
  "Description":null,
  "Identifier":"17b2c3f9-b4ef-401f-bc6d-f01ab6311f84",
  "Name":"Trigger Rendition",
  "Details":[
    {
      "Name":"Channel Configuration",
      "Value":"Stereo"
    },
    {
      "Name":"Audio Service",
      "Value":"Primary"
    },
    {
      "Name":"Language",
```

```
        "Value":"English"
      }
    ],
    "Segments":{
      "Briefs":[
        {
          "Description":null,
          "Identifier":"6ef8de15-cd72-4184-b36d-dfb9a4cf87e1",
          "Name":"Trigger Backup Segment"
        },
        {
          "Description":null,
          "Identifier":"82dfa6e9-34b0-40bb-a047-d54ae4f32faa",
          "Name":"Trigger Base Segment"
        }
      ]
    },
    "Tracks":{
      "Briefs":[
        {
          "Description":"This track is from the source audio tracks",
          "Identifier":"749d9558-5606-4295-b88a-e35bb0f25e3a",
          "Name":"Rendition Stereo Track"
        },
        {
          "Description":"This track is from the source audio tracks",
          "Identifier":"4385298a-fe7c-40d8-959b-a798ab305e82",
          "Name":"Rendition audio track 4"
        },
        {
          "Description":"This track is from the source audio tracks",
          "Identifier":"4f889818-2cbc-4403-86d8-741c7b8c1c58",
          "Name":"Rendition audio track 3"
        },
        {
          "Description":"This track is from the source audio tracks",
          "Identifier":"b0e1121d-6275-4acd-bcb5-42326647ef13",
          "Name":"Rendition Stereo Track"
        },
        {
          "Description":"This track is from the source audio tracks",
          "Identifier":"cda1680c-4d58-4938-bd56-9dc88b391ad4",
          "Name":"Rendition audio track 2"
        },
        {
          "Description":"This track is from the source audio tracks",
          "Identifier":"efc212dc-78eb-4a7b-aae8-444067466754",
          "Name":"Rendition Mono Track"
        }
      ]
    },
    "Type":0
}
```

GetSegments

The purpose of this GET operation is to obtain a list of Segments that comprise a specific rendition of a program.

GetSegments has the following format:

```
http://<host>:<port>/Programs/GetSegments?rendition={RENDITION
GUID}
```

Operation Sequence

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

GetPrograms (program GUID) > *GetRenditions* (rendition GUID)

Parameters

Parameter	Description
rendition	GUID; string that identifies a specific rendition.

Results

On success, *GetSegments* returns an array; one record for each segment in the rendition.

If no Segments are present, the array is returned empty ([]).

If the rendition you specify does not exist, the Live Stream server returns an error.

Example

```
http://10.9.9.9:18000/Programs/GetSegments
?rendition=1e32e2f1-3702-4d83-8460-e9d2231db3c5
```

Typical Response

In this response, the target rendition has two segments. Each is identified by Name and by an Identifier GUID.

```
[
  {
    "Description":null,
    "Identifier":"6ef8de15-cd72-4184-b36d-dfb9a4cf87e1",
    "Name":"Trigger Backup Segment"
  },
  {
    "Description":null,
    "Identifier":"82dfa6e9-34b0-40bb-a047-d54ae4f32faa",
    "Name":"Trigger Base Segment"
  }
]
```

GetSegment

The purpose of this GET operation is to obtain details about a specific segment.

GetSegment has the following format:

```
http://<host>:<port>/Programs/GetSegment?identifier={SEGMENT GUID}
```

Operation Sequence

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

GetPrograms (program GUID) > *GetRenditions* (rendition GUID) > *GetSegments* (segment GUID)

Parameters

Parameter	Description
identifier	GUID; string that identifies a specific segment.

Results

On success, *GetSegment* returns a record, with details about the target segment and all of its Material.

If no segment exists with the specified GUID, an error is returned.

Example

```
http://10.9.9.9:18000/Programs/GetSegment
?identifier=6ef8de15-cd72-4184-b36d-dfb9a4cf87e1
```

Typical Response

In this response, the target segment is Trigger Backup Segment. It has a StartTrigger and an EndTrigger, and Material consisting of one asset.


```
{
  "Description":null,
  "Identifier":"6ef8de15-cd72-4184-b36d-dfb9a4cf87e1",
  "Name":"Trigger Backup Segment",
  "EndTrigger":{
    "Description":"DurationIdentifier",
    "Identifier":"31afe9dd-9130-42db-b1af-7e0e7e19a1a7",
    "Name":"Duration",
    "Details":[
      {
        "Name":"Duration",
        "Value":"00:02:00"
      }
    ]
  },
  "Materials":{
```

```
    "Briefs":[
      {
        "Description":"A placeholder for a source.",
        "Identifier":"e816c0fd-d4df-4f41-a253-8d2b4ce829da",
        "Name":"Backup Source Placeholder"
      }
    ]
  },
  "Priority":0,
  "StartTrigger":{
    "Description":"TimeOfDayIdentifier",
    "Identifier":"5cb3f0dc-d50e-495c-9197-19a72d4e981a",
    "Name":"Time of Day",
    "Details":[
      {
        "Name":"Time",
        "Value":"12:00:00"
      },
      {
        "Name":"Repeat Interval",
        "Value":"00:04:00"
      }
    ]
  }
}
```

GetMaterials

The purpose of this GET operation is to obtain a list of Materials that comprise a specific segment.

GetMaterials has the following format:

```
http://<host>:<port>/Programs/GetMaterials?segment={SEGMENT GUID}
```

Operation Sequence

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

GetPrograms (program GUID) > *GetRenditions* (rendition GUID) > *GetSegments* (segment GUID)

Parameters

Parameter	Description
segment	GUID; string that identifies a specific segment.

Results

On success, *GetMaterials* returns an array, with one record per material in the segment. If the segment does not contain any material, the array is returned empty ([]).

Example

```
http://10.9.9.9:18000/Sources/GetMaterials
?segment=1f3a2a14-4ea6-411d-83a7-59694f4eed7e
```

Typical Response

In this response, the target segment has three material items.

```
[
  {
    "Description":"A placeholder for a source.",
    "Identifier":"262f8038-62bd-448d-b0c3-fe98177557fa",
    "Name":"Basic Source Placeholder"
  },
  {
    "Description":"An asset with both a static visual component.",
    "Identifier":"6ee9b669-2e0f-4810-9fff-290a25601551",
    "Name":"Yoshi"
  },
  {
    "Description":"An asset with both a sound and visual
component.",
    "Identifier":"a8b7d795-b309-41c0-83fa-ce5d47ed81aa",
    "Name":"Watchmen"
  }
]
```

GetMaterial

The purpose of this GET operation is to obtain details about a specific item of material.

GetMaterial has the following format:

```
http://<host>:<port>/Programs/GetMaterial?identifier={MATERIAL
GUID}
```

Operation Sequence

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

GetPrograms (program GUID) > *GetRenditions* (rendition GUID) > *GetSegments* (segment GUID) > *GetMaterials* (material GUID)

Parameters

Parameter	Description
identifier	GUID; string that identifies a specific material.

Results

On success, *GetMaterial* returns a record with details about the target material.

If no material exists with the specified GUID, an error is returned.

Example

```
http://10.9.9.9:18000/Programs/GetMaterial
?identifier=0a853173-734f-473e-9b4e-d85d0524e97c
```

Typical Response

In this response, the target material is identified by the Description, Identifier, and a Name. The Details array provides a set of key/value pairs appropriate for this type of material:

```
{
  "Description":"A placeholder for a source.",
  "Identifier":"0a853173-734f-473e-9b4e-d85d0524e97c",
  "Name":"Base Source Placeholder",
  "Details":[
    {
      "Name":"Left",
      "Value":"0"
    },
    {
      "Name":"Width",
      "Value":"1920"
    },
    {
      "Name":"Top",
```

```
        "Value":"0"
      },
      {
        "Name":"Height",
        "Value":"1080"
      },
      {
        "Name":"Lock in Visual Editor",
        "Value":"True"
      }
    ],
    "ZIndex":0
}
```

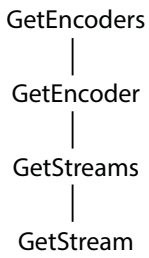
Encoders Operations

The purpose of the Encoders group of operations is to obtain details about specific encoders that have been implemented in the target Live Stream server, and the streams the comprise a given encoder.

- GetEncoders
- GetEncoder
- GetStreams
- GetStream

Note: All Encoders operations start with `http://<host>:<port>/Encoders/`.

In this diagram, the operations are organized hierarchically, by GUID requirement. The GetEncoders operation does not require a GUID.



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

GetEncoders

The purpose of this GET operation is to obtain a list of encoders that have been created on the target Live Stream system. This operation has no parameters.

GetEncoders has the following format:

`http://<host>:<port>/Encoders/GetEncoders`

Operation Sequence

No operations must be executed before you can execute this operation.

Results

On success, *GetEncoders* returns an array, which has one record for each encoder on the Live Stream server.

If no encoders are present, the array is returned empty ([]).

Example

`http://10.9.9.9:18000/Encoders/GetEncoders`

Typical Response

In this response, the target Live Stream server has four encoders; each identified by Name and Description, and GUID in the Identifier which you can use to target the encoder for further utilization.


```
[
  {
    "Description":"HEVC",
    "Identifier":"0c54e3a9-dd89-4d43-8dc6-a009ab231ff0",
    "Name":"HEVC"
  },
  {
    "Description":"AVC",
    "Identifier":"0f25fcb4-c412-4761-a96e-9eb5480a7013",
    "Name":"AVC-LIVE"
  },
  {
    "Description":"AVC",
    "Identifier":"1d0c1ae0-10d7-47b4-8f67-6656294df67f",
    "Name":"AVC"
  },
  {
    "Description":"AAC",
    "Identifier":"36c5006c-f04b-4e3a-9ca3-8ebbfb1cbc8f",
    "Name":"AAC"
  }
]
```

GetEncoder

The purpose of this GET operation is to obtain details about a specific encoder.

GetEncoder has the following format:

```
http://<host>:<port>/Encoders/GetEncoder?identifier={ENCODER GUID}
```

Operation Sequence

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

GetEncoders (encoder GUID)

Parameters

Parameter	Description
identifier	GUID; string that identifies a specific encoder.

Results

On success, *GetEncoder* returns a record with details about the target encoder and its streams.

If no encoder exists with the specified GUID, an error is returned.

Example

```
http://10.9.9.9:18000/Encoders/GetEncoder
?identifier=0c54e3a9-dd89-4d43-8dc6-a009ab231ff0
```

Typical Response

In this response, the target encoder is identified by its Description, Identifier, and Name. Each stream in the encoder is enumerated—also with a Description, Identifier, and Name.


```
{
  "Description":"HEVC",
  "Identifier":"0c54e3a9-dd89-4d43-8dc6-a009ab231ff0",
  "Name":"HEVC",
  "Details":[
    {
      "Name":"GOP Duration",
      "Value":"3"
    }
  ],
  "Streams":{
    "Briefs":[
      {
        "Description":null,
        "Identifier":"0ab406ba-3c9d-47b1-8e28-ae94fd17db2a",
        "Name":"UHD HEVC"
```

```
      },
      {
        "Description":null,
        "Identifier":"4d75546c-c146-4818-b4f5-f00361190bfa",
        "Name":"1080p-HEVC"
      },
      {
        "Description":null,
        "Identifier":"cbccf0c6-5d12-4f6b-af66-2b53451625cb",
        "Name":"540p-HEVC"
      }
    ]
  }
}
```

GetStreams

The purpose of this GET operation is to obtain a list of streams that have been added to a specific encoder.

GetStreams has the following format:

```
http://<host>:<port>/Encoders/GetStreams?encoder={ENCODER GUID}
```

Operation Sequence

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

GetEncoders (encoder GUID)

Parameters

Parameter	Description
encoder	GUID; string that identifies a specific encoder.

Results

On success, *GetStreams* returns an array with one record for each stream in the encoder.

If no Streams are present, the array is returned empty ([]).

Example

```
http://10.9.9.9:18000/Encoders/GetStreams
?encoder=1bad0882-7cac-4cdd-b2a7-28dd909de467
```

Typical Response

In this response, the target encoder has two streams: one 720p, another for 1080p. Each is identified by Name and by an Identifier GUID. You can use the GUID to target a specific stream for further use.

```
[
  {
    "Description":null,
    "Identifier":"00befa87-3940-4ae0-8652-a66d0e46c0ac",
    "Name":"720p-RTMP"
  },
  {
    "Description":null,
    "Identifier":"e4aa2e66-04c9-4705-aeab-6490c8cec8fc",
    "Name":"1080p-YTL"
  }
]
```

GetStream

The purpose of this GET operation is to obtain details about a specific stream.

GetStream has the following format:

```
http://<host>:<port>/Encoders/GetStream?identifier={STREAM GUID}
```

Operation Sequence

Execute one of the following operations to obtain the required GUID for this operation:

GetStreams or *GetEncoder* (stream GUID)

Parameters

Parameter	Description
identifier	GUID; string that identifies a specific stream.

Results

On success, *GetStream* returns a record, with details about the target stream.

If no stream exists with the specified GUID, an error is returned.

Example

```
http://10.9.9.9:18000/Encoders/GetStream
?identifier=00befa87-3940-4ae0-8652-a66d0e46c0ac
```

Typical Response

In this response, the target stream is identified by the Identifier and Name, plus its Details. The key/value pairs provide a set of parameters appropriate for this type of material.


```
{
  "Description":null,
  "Identifier":"00befa87-3940-4ae0-8652-a66d0e46c0ac",
  "Name":"720p-RTMP",
  "Details":[
    {
      "Name":"Width",
      "Value":"1280"
    },
    {
      "Name":"Height",
      "Value":"720"
    },
    {
      "Name":"Bitrate",
      "Value":"3000"
    },
```

```
    {
      "Name":"Framerate",
      "Value":"30"
    },
    {
      "Name":"Profile",
      "Value":"High"
    },
    {
      "Name":"GOP length",
      "Value":"90"
    },
    {
      "Name":"Display Aspect Ratio",
      "Value":"16:9"
    },
    {
      "Name":"Burn-in Timecode",
      "Value":"True"
    },
    {
      "Name":"CBR mode",
      "Value":"True"
    },
    {
      "Name":"Low latency",
      "Value":"False"
    },
    {
      "Name":"IDR Splices",
      "Value":"False"
    },
    {
      "Name":"Compressor",
      "Value":"x264"
    },
    {
      "Name":"Command Line Options",
      "Value":"--bframes 0"
    },
    {
      "Name":"Preset",
      "Value":"veryfast"
    },
    {
      "Name":"Video Processor",
      "Value":"Default"
    }
  ],
  "Essence":1
}
```

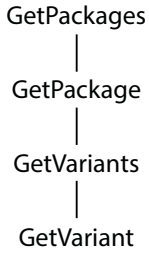
Packages Operations

The Packages operations enable you to query packages and their variants, that have been created on the system (for example, Apple HLS or RTMP).

- GetPackages
- GetPackage
- GetVariants
- GetVariant

Note: All Packages operations start with `http://<host>:<port>/Packages/`.

In this diagram, the operations are organized hierarchically, by GUID requirement. The *GetPackages* operation does not require a GUID.



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

GetPackages

The purpose of this GET operation is to obtain a list of packages that have been created on the target Live Stream system. This operation has no parameters.

GetPackages has the following format:

`http://<host>:<port>/Packages/GetPackages`

Operation Sequence

No operations must be executed before you can execute this operation.

Results

On success, *GetPackages* returns an array, which has one record for each package on the Live Stream server.

If no Packages are present, the array is returned empty ([]).

Example

`http://10.9.9.9:18000/Packages/GetPackages`

Typical Response

In this response, the target Live Stream server has eight packages. Each is identified by a Description, an Identifier GUID, and Name. You can use the GUID to target a specific package for further use.


```
[
  {
    "Description":"DASH",
    "Identifier":"026bb4c3-dd10-4f45-9bdf-66388c20281d",
    "Name":"DASH-SCTE"
  },
  {
    "Description":"RTMP",
    "Identifier":"0b21da42-f881-4e1b-89a4-0413942662f2",
    "Name":"RTMP-Generic"
  },
  {
    "Description":"RTMP",
    "Identifier":"0fff4aed-d89e-44a6-9674-dd046b276c98",
    "Name":"TS-Akamai-RTMP"
  },
  {
    "Description":"YouTube Live",
    "Identifier":"185b4a7b-126e-4c36-bab7-2626244e9a2f",
    "Name":"YTL-Test"
  },
  {
    "Description":"Apple HLS",
    "Identifier":"19df3b43-470a-4451-b243-51997fe81a93",
    "Name":"HLS AVC Test"
```

```
  },
  {
    "Description":"CMAF",
    "Identifier":"2ba22df7-9724-41ff-adc5-00a1dfe4b19f",
    "Name":"CMAF1"
  },
  {
    "Description":"MP4",
    "Identifier":"401310ba-42f2-4ed6-ac8d-2b68b0cbe4ee",
    "Name":"MP4-HEVC"
  },
  {
    "Description":"Transport Stream",
    "Identifier":"e40fcbee-6a2e-4567-9a48-78e32af0d0a8",
    "Name":"TS Package Test"
  }
]
```

GetPackage

The purpose of this GET operation is to obtain details about a specific package.

GetPackage has the following format:

```
http://<host>:<port>/Packages/GetPackage?identifier={package GUID}
```

Operation Sequence

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

GetPackages (package GUID)

Parameters

Parameter	Description
identifier	GUID; string that identifies a specific package.

Results

On success, *GetPackage* returns a record, with details about the target package and its variants.

If no package exists with the specified GUID, an error is returned.

Example

```
http://10.9.9.9:18000/Packages/GetPackage
?identifier=026bb4c3-dd10-4f45-9bdf-66388c20281d
```

Typical Response

In this response, the target package is identified by its Description, Identifier, and Name. The package also lists all of its settings, as appropriate by package type, followed by the program and type. Variants in this package, if any, are also listed with their identification and details.

```
{
  "Description":"DASH",
  "Identifier":"026bb4c3-dd10-4f45-9bdf-66388c20281d",
  "Name":"DASH-SCTE",
  "Details":[
    {
      "Name":"Segment Duration",
      "Value":"9"
    },
    {
      "Name":"Playlist Name",
      "Value":"manifest"
    },
    {
      "Name":"Directory Rollover",
```

```
        "Value":"False"
      },
      {
        "Name":"UTC Timestamps",
        "Value":"False"
      },
      {
        "Name":"SCTE-35 in Manifest",
        "Value":"True"
      },
      {
        "Name":"SCTE-35 in fMP4",
        "Value":"True"
      },
      {
        "Name":"Playlist Type",
        "Value":"Rolling"
      },
      {
        "Name":"Elements",
        "Value":"25"
      },
      {
        "Name":"Encryption",
        "Value":"Unencrypted"
      }
    ],
    "Program":"92aa9eb9-a1fb-4aab-b920-c3259895edb4",
    "Type":2,
    "Variants":{
      "Briefs":[
        {
          "Description":null,
          "Identifier":"5003fae7-0bfa-42f9-b7a4-818ede966c12",
          "Name":"540p-X-AVC"
        },
        {
          "Description":null,
          "Identifier":"f4232a60-d80b-48ab-a5c5-eaeb9619e1a8",
          "Name":"720p-X-AVC"
        }
      ]
    }
  }
```

GetVariants

The purpose of this GET operation is to obtain a list of variants that comprise the target package.

GetVariants has the following format:

```
http://<host>:<port>/Packages/GetVariants?package={package GUID}
```

Operation Sequence

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

GetPackages (package GUID)

Parameters

Parameter	Description
package	GUID; string that identifies a specific package.

Results

On success, *GetVariants* returns an array, which has one record for each variant in the target package.

If no Variants are present, the array is returned empty ([]).

Example

```
http://10.9.9.9:18000/Packages/GetVariants
?package=1bad0882-7cac-4cdd-b2a7-28dd909de467
```

Typical Response

In this response, the target package has two variants; each with a Description, Identifier, and Name, which you can use to target the variant for further utilization.


```
[
  {
    "Description":null,
    "Identifier":"5003fae7-0bfa-42f9-b7a4-818ede966c12",
    "Name":"540p-X-AVC"
  },
  {
    "Description":null,
    "Identifier":"f4232a60-d80b-48ab-a5c5-eaeb9619e1a8",
    "Name":"720p-X-AVC"
  }
]
```

GetVariant

The purpose of this GET operation is to obtain details about a specific variant.

GetVariant has the following format:

```
http://<host>:<port>/Packages/GetVariant?identifier={VARIANT GUID}
```

Operation Sequence

Execute one of the following operations to obtain the required GUID for this operation:

GetVariants (variant GUID)
or
GetPackages (package GUID) > *GetPackage* (variant GUID)

Parameters

Parameter	Description
identifier	GUID; string that identifies a specific variant.

Results

On success, *GetVariant* returns a record, with details about the target variant.

If no variant exists with the specified GUID, an error is returned.

Example

```
http://10.9.9.9:18000/Packages/GetVariant
?identifier=f4232a60-d80b-48ab-a5c5-eaeb9619e1a8
```

Typical Response

In this response, the target package is identified by its Description, Identifier, and a Name. Specific settings are presented in key/value pairs, plus a rendition and stream.

```
{
  "Description":null,
  "Identifier":"f4232a60-d80b-48ab-a5c5-eaeb9619e1a8",
  "Name":"720p-X-AVC",
  "Details":[
    {
      "Name":"Default",
      "Value":"False"
    }
  ],
  "Rendition":"1b5b2916-0948-4c3d-b726-d92df1ee6ad1",
  "Streams":[
    "b50a47dc-601f-43b7-a11a-0e2864112f75",
    "a0de7639-687c-4ad8-b226-dd2572cb1f7d"
  ]
}
```

Channels Operations

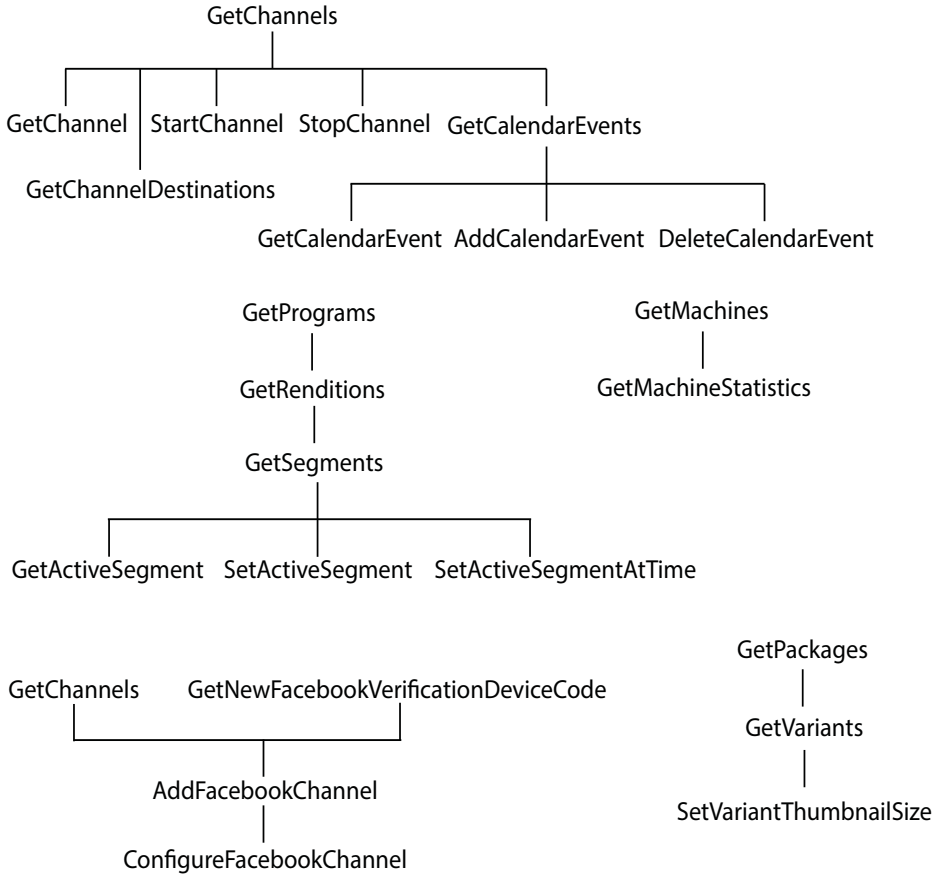
The Channels category of operations enable you to identify all of the channels in your Live Stream server. You can use these operations to query and control each channel and work with channel calendar events, and you can set the active segment on a channel.

- GetChannels
- GetChannel
- GetChannelOutputLocations
- SetVariantThumbnailSize
- GetChannelThumbnail
- StartChannel
- StopChannel
- GetCalendarEvents
- GetCalendarEvent
- AddCalendarEvent
- DeleteCalendarEvent
- GetActiveSegment
- SetActiveSegment
- SetActiveSegmentAtTime
- GetMachineStatistics
- GetChannelStatistics
- GetNewFacebookVerificationDeviceCode
- AddFacebookChannel
- ConfigureFacebookChannel

Note: All Channels operations start with `http://<host>:<port>/Channels/`.

Channels are accessible to all the servers in a system and you can access or configure any channel from any server in the system by targeting a single machine.

In this diagram, the Channels operations are organized hierarchically in four different trees, based on their GUID requirements. Each operation in this category requires a channel GUID to execute correctly.



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

GetChannels

The purpose of this GET operation is to obtain a list of channels on the target Live Stream system. No parameters are required.

GetChannels has the following format:

`http://<host>:<port>/Channels/GetChannels`

Operation Sequence

No operations must be executed before you can execute this operation.

Results

On success, *GetChannels* returns an array, with one record for each channel in the Live Stream system.

If there are no channels, the array is returned empty ([]).

Example

`http://10.9.9.9:18000/Channels/GetChannels`

Typical Response

In this response, the Live Stream server has several channels; each with a Description, Identifier, and Name, which you can use to target the channel for further utilization.


```
[
  {
    "Description":null,
    "Identifier":"12133c5b-59dd-48cf-8443-ec7a2501b252",
    "Name":"HLS1"
  },
  {
    "Description":null,
    "Identifier":"40de4c1b-5523-4ed3-a89a-e319a3fc88d2",
    "Name":"RTMPAkamai"
  },
  {
    "Description":null,
    "Identifier":"486caeef-de83-4862-89c7-e79dab1a0564",
    "Name":"YTL"
  },
  {
    "Description":null,
    "Identifier":"826891c7-d81e-428d-b868-8f0ed58c77fe",
    "Name":"Dash_SCTE"
  }
]
```

GetChannel

The purpose of this GET operation is to obtain details about a specific channel on the target Live Stream server.

GetChannel has the following format:

```
http://<host>:<port>/Channels/GetChannel?identifier={CHANNEL GUID}
```

Operation Sequence

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

GetChannels (channel GUID)

Parameters

Parameter	Description
identifier	GUID; string that identifies a specific channel.

Results

On success, *GetChannel* returns a record with details, including events and packages in the specified channel.

If no channel exists with the specified GUID, an error is returned.

Example

```
http://10.9.9.9:18000/Channels/GetChannel
?identifier=9378dd79-a374-46f8-9f23-dc68d8d52d07
```

Typical Response

In this response, the target channel is *Northwest Passage*, with all its details.


```
{
  "Description":"Northwest Passage",
  "Identifier":"12133c5b-59dd-48cf-8443-ec7a2501b252",
  "Name":"HLS1",
  "Active":false,
  "ActiveSegments":[

  ],
  "Assignments":[
    {
      "Live":"2a74714a-8368-4741-8707-7547e2155d73",
      "Source":"21f65bae-ed24-4025-9254-a873bd6f247e"
    }
  ],
  "CalendarEvents":{
    "Briefs":[
```

```
  ]
},
"Machine":"da716ac0-d5e2-4d18-a718-533b42f6457a",
"PrimaryPackage":{
  "Description":null,
  "Identifier":"af06a78f-d467-4112-bd28-671a0eca5bcb",
  "Name":"General outputs locations",
  "Details":[
    {
      "Name":"Output Location",
      "Value":"Push to CDN"
    },
    {
      "Name":"Publishing Point",
      "Value":"http:\/\/post.nwstudio.akamaihd.net\/554433"
    },
    {
      "Name":"HTTP Method",
      "Value":"POST"
    },
    {
      "Name":"Remove Local Copy",
      "Value":"True"
    },
    {
      "Name":"Output Package Name",
      "Value":"Default"
    }
  ],
  "Package":"19df3b43-470a-4451-b243-51997fe81a93"
},
"SecondaryPackage":[
  {
    "Description":null,
    "Identifier":"af06a78f-d467-4112-bd28-671a0eca5bcb",
    "Name":"General outputs locations",
    "Details":[
      {
        "Name":"Output Location",
        "Value":"Push to CDN"
      },
      {
        "Name":"Publishing Point",
        "Value":"http:\/\/post.nwstudio.akamaihd.net\/554433"
      },
      {
        "Name":"HTTP Method",
        "Value":"POST"
      },
      {
        "Name":"Remove Local Copy",
        "Value":"True"
      },
      {
        "Name":"Output Package Name",
        "Value":"Default"
      }
```

```
        ],
        "Package":"19df3b43-470a-4451-b243-51997fe81a93"
      }
    ]
}
```

GetChannelOutputLocations

The purpose of this GET operation is to obtain the destinations for the target channel.

GetChannelOutputLocations has the following format:

```
http://<host>:<port>/Channels/
GetChannelOutputLocations?channel={CHANNEL GUID}
```

Operation Sequence

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

GetChannels (channel GUID)

Parameters

Parameter	Description
channel	GUID; string that identifies a specific channel.

Results

On success, *GetChannelOutputLocations* returns an array, with a record for each destination in the target channel (primary, and potentially multiple secondary).

Example

```
http://10.9.9.9:18000/Channels/GetChannelOutputLocations
?channel=1bad0882-7cac-4cdd-b2a7-28dd909de467
```

Typical Response

In this response, the channel has a primary and secondary package; details include Description, Identifier, and Name, the output location, and the package GUID.


```
[
  {
    "Description":null,
    "Identifier":"af06a78f-d467-4112-bd28-671a0eca5bcb",
    "Name":"General outputs locations",
    "Details":[
      {
        "Name":"Output Location",
        "Value":"Push to CDN"
      },
      {
        "Name":"Publishing Point",
        "Value":"http:\/\/post.nwstudio.akamaihd.net\/554433"
      },
      {
        "Name":"HTTP Method",
        "Value":"POST"
      },
```

```
      {
        "Name":"Remove Local Copy",
        "Value":"True"
      },
      {
        "Name":"Output Package Name",
        "Value":"Default"
      }
    ],
    "Package":"19df3b43-470a-4451-b243-51997fe81a93"
  },
  {
    "Description":null,
    "Identifier":"af06a78f-d467-4112-bd28-671a0eca5bcb",
    "Name":"General outputs locations",
    "Details":[
      {
        "Name":"Output Location",
        "Value":"Push to CDN"
      },
      {
        "Name":"Publishing Point",
        "Value":"http:\/\/post.nwstudio.akamaihd.net\/554433"
      },
      {
        "Name":"HTTP Method",
        "Value":"POST"
      },
      {
        "Name":"Remove Local Copy",
        "Value":"True"
      },
      {
        "Name":"Output Package Name",
        "Value":"Default"
      }
    ],
    "Package":"19df3b43-470a-4451-b243-51997fe81a93"
  }
]
```

SetVariantThumbnailSize

This POST operation sets the size of thumbnails obtained from the target variant.

Note: Variants can be shared across channels, so *SetVariantThumbnailSize* affects all instances of this variant in every channel where it is used.
The variant must be inactive before you execute this operation. After execution, you must restart the channel of the variant, then execute *GetChannelThumbnail*.

Use this operation to specify the size before calling *GetChannelThumbnail*. This operation has no effect on the size of thumbnails presented in the Channels panel of the web app.

Output sizes up to 4096 x 2160 are supported. Use 0 for height/width to revert the size to their default values (160 x 90).

SetVariantThumbnailSize has the following format:

```
http://<host>:<port>/Channels/SetVariantThumbnailSize
?variant={VARIANT GUID}&width={INT}&height={INT}
```

Operation Sequence

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

GetPackages (package GUID) > *GetVariants* (variant GUID)

Parameters

Parameter	Description
variant	GUID; string that identifies a specific variant.
width	INT; string that specifies the width of the thumbnail in pixels. Max: 4096.
height	INT; string that specifies the height of the thumbnail in pixels. Max: 2160.

Results

On success, *SetVariantThumbnailSize* returns a string indicating that the thumbnail size has been set.

Example

```
http://10.0.25.162:18000/Channels/SetVariantThumbnailSize
?variant=b0e7a6c0-a920-4702-b4f7-5fafe599a26c
&width=1280&height=720
```

Typical Response

This operation was successful: "Set thumbnail size for variant 720p-X-AVC".

GetChannelThumbnail

The purpose of this GET operation is to obtain a thumbnail of a specific variant of the target active channel (inactive channels do not have thumbnails). The operation returns a JPEG, which you can display (render) or save as a file.

The *GetChannelThumbnail* command requires both a channel and variant GUID. So to get the thumbnail of variant A, you provide the variant A GUID; to get the thumbnail of variant B, you provide the variant B GUID.

Variants are source-agnostic, while channels are not. Variant A may be used in Channel A and Channel B at the same time, which may be using different sources, in which case the thumbnails will be the same size but different content.

Note: If you have modified the thumbnail size (using *SetVariantThumbnailSize),* you must restart the target channel for the size modification to take effect, then execute *GetChannelThumbnail*.

GetChannelThumbnail has the following format:

```
http://<host>:<port>/Channels/GetChannelThumbnail
?channel={CHANNEL GUID}&variant={VARIANT GUID}
```

Operation Sequence

Execute the following operations to obtain the required GUIDs for this operation:

GetChannels (channel GUID)

and

GetPackages (package GUID) > *GetVariants* (variant GUID)

Parameters

Parameter	Description
channel	GUID; string that identifies a specific channel.
variant	GUID; string that identifies a specific variant.

Results

On success, *GetChannelThumbnail* returns a JPEG.

If the channel is inactive, an HTTP 404 error is returned—there is no thumbnail.

Example

```
http://10.9.9.9:18000/Channels/GetChannelThumbnail
?channel=27602854-35a6-4f0c-827c-ebd7ac5d40a9
&variant=b0e7a6c0-a920-4702-b4f7-5fafe599a26c
```

StartChannel

The purpose of this POST operation is to begin streaming a specified channel. When you start the channel, the active segment is encoded according to the package variant definitions and delivered to the locations you have specified.

Note: Channels are defined specifically for a hardware port on the server where they are created. Thus the host that you specify for this operation must be that of the server where the Channel was added.

StartChannel has the following format:

```
http://<host>:<port>/Channels/StartChannel?channel={CHANNEL GUID}
```

Operation Sequence

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

GetChannels (channel GUID)

Parameters

Parameter	Description
channel	GUID; string that identifies a specific channel.

Results

On success, *StartChannel* returns a status message indicating success: "Successfully started channel".

Example

```
http://10.9.9.9:18000/Channels/StartChannel
?channel=d5d2a977-68c9-4fa3-a687-5d60535d4958
```

Typical Response

In this response, the operation was successful: "Successfully started channel".

StopChannel

The purpose of this POST operation is to terminate the streaming of a specified channel.

Note: Channels are defined specifically for a hardware port on the server where they are created. Thus the host that you specify for this operation must be that of the server where the Channel was added.

StopChannel has the following format:

```
http://<host>:<port>/Channels/StopChannel?channel={CHANNEL GUID}
```

Operation Sequence

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

GetChannels (channel GUID)

Parameters

Parameter	Description
channel	GUID; string that identifies a specific channel.

Results

On success, *StopChannel* returns a status message indicating success: "Successfully stopped channel".

Example

```
http://10.9.9.9:18000/Channels/StopChannel
?channel=d5d2a977-68c9-4fa3-a687-5d60535d4958
```

Typical Response

In this response, the operation was successful: "Successfully stopped channel".

GetCalendarEvents

The purpose of this GET operation is to obtain a list of CalendarEvents from the target channel.

GetCalendarEvents has the following format:

```
http://<host>:<port>/Channels/GetCalendarEvents?channel={CHANNEL GUID}
```

Operation Sequence

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

GetChannels (channel GUID)

Parameters

Parameter	Description
channel	GUID; string that identifies a specific channel.

Results

On success, *GetCalendarEvents* returns an array, with one record for each calendar event in the channel.

If no calendar events are present, the array is returned empty ([]).

Example

```
http://10.9.9.9:18000/Channels/GetCalendarEvents
?channel=9378dd79-a374-46f8-9f23-dc68d8d52d07
```

Typical Response

In this response, the target channel has two calendar events; with a Description, Identifier, and Name, which you can use to target the calendar event for further utilization.

```
[
  {
    "Description": null,
    "Identifier": "4ac69260-b10e-4da9-955e-831ea02a814b",
    "Name": "Event1"
  },
  {
    "Description": null,
    "Identifier": "290c6ad3-3b2a-42cd-9e31-aac9fd069d93",
    "Name": "Event2"
  }
]
```

GetCalendarEvent

The purpose of this GET operation is to obtain details about a specific calendar event in the target channel.

GetCalendarEvent has the following format:

```
http://<host>:<port>/Channels/
GetCalendarEvent?identifier={CALENDAREVENT GUID}
```

Operation Sequence

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

GetChannels (channel GUID) > *GetCalendarEvents* (calendar-event GUID)

Parameters

Parameter	Description
identifier	GUID; string that identifies a specific calendar event.

Results

On success, *GetCalendarEvent* returns a record, with details about the event, including frequency, and start and end times.

If no calendar event exists with the specified GUID, an error is returned. For example, "Could not find calendar event with identifier 4ac69260-b10e-5da9-955e-831ea02a814b".

Example

```
http://10.9.9.9:18000/Channels/GetCalendarEvent
?identifier=2f5df131-deaf-4789-87dd-7ff850bcf1dc
```

Typical Response

In this response, the target CalendarEvent has a Description, Identifier, and Name, plus details about the event, including the start and end timestamps.

```
{
    "Description": null,
    "Identifier": "4ac69260-b10e-4da9-955e-831ea02a814b",
    "Name": "Event1",
    "Color": {
        "primary": "green",
        "secondary": null
    },
    "Details": [
        {
            "Name": "Frequency",
            "Value": "Does Not Repeat"
        },
```

```
            {
                "Name": "By day",
                "Value": ""
            }
        ],
        "End": "2017-11-15T10:00:00.0000000-08:00",
        "Start": "2017-11-15T09:00:00.0000000-08:00",
        "Title": null
    }
```

AddCalendarEvent

The purpose of this POST operation is to add a calendar event to the specified channel. A calendar event is the date and time you want to start and stop broadcasting a channel. You can add as many calendar events as you require.

Note: If you add multiple events with the same name, the name is appended with (<999>) where 999 is an incremental integer starting at 1.

AddCalendarEvent has the following format:

```
http://<host>:<port>/Channels/AddCalendarEvent
?name={NAME}&channel={CHANNEL GUID}&start={START}&end={END}
```

Operation Sequence

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

GetChannels (channel GUID)

Parameters

Parameter	Description
name (optional)	String; practical name that identifies this calendar event. If not supplied, the default string “Scheduled Stream Activation” is placed in the Name field.
channel	GUID; string that identifies this channel.
start	String; identifies the date and time to start the channel. If GMT is not specified, the time is local to the server. Support is provided for date/time strings that follow a recognized pattern, as described in the Microsoft .net DateTime.Parse method (for details, see https://msdn.microsoft.com/en-us/library/system.datetime.parse.aspx#StringToParse). For example, MM-DD-YYYY HH:MM:SS or Thu, 01 May 2017 07:34:42 GMT.
end	String; identifies the date and time to stop the channel, as described above in start.

Results

On success, *AddCalendarEvent* returns a record with the GUID of the new event.

Example

```
http://10.0.2.258:18000/Channels/AddCalendarEvent
?channel=9378dd79-a374-46f8-9f23-dc68d8d52d07&name=FirstEvent
&start=03-15-2017 09:00:00&end=03-15-2017 10:00:00
```

Typical Response

In this response, the new event was successfully added to the channel; its Identifier GUID is returned, which you can use to target the event for further utilization.


```
{
    "Description": null,
    "Identifier": "4ac69260-b10e-4da9-955e-831ea02a814b",
    "Name": "Event1"
}
```

DeleteCalendarEvent

The purpose of this POST operation is to delete a calendar event from a channel.

DeleteCalendarEvent has the following format:

```
http://<host>:<port>/Channels/DeleteCalendarEvent
?channel={CHANNEL GUID}&calendar-event={CALENDAR-EVENT GUID}
```

Operation Sequence

Execute one of the following operation sequences to obtain the required GUID for this operation:

GetChannels (channel GUID)

and

GetCalendarEvents(calendar-event GUID)

or

AddCalendarEvent (calendar-event GUID)

Parameters

Parameter	Description
channel	GUID; string that identifies this channel.
calendar-event	GUID; string that identifies this calendar event.

Results

On success, *DeleteCalendarEvent* returns a status message: "Successfully deleted calendar event".

Example

```
http://10.0.2.258:18000/Channels/DeleteCalendarEvent
?channel=9378dd79-a374-46f8-9f23-dc68d8d52d07
&calendar-event=a82a7f8e-d3c1-4bab-8644-67a4f5917a52
```

Typical Response

In this response, the new event was successfully deleted. A successful Status was returned: "Successfully deleted calendar event".

GetActiveSegment

The purpose of this GET operation is to obtain details about the active segment on the target channel, which must be active.

GetActiveSegment has the following format:

```
http://<host>:<port>/Channels/GetActiveSegment?
channel={CHANNEL GUID}
```

Operation Sequence

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

GetChannels (channel GUID)

Parameters

Parameter	Description
channel	GUID; string that identifies this channel.

Results

On success, *GetActiveSegment* returns the active segment's record.

If the channel isn't active, it returns an error: "Channel is not active".

If no channel exists with the specified GUID, an error is returned: "Could not find channel with identifier <channel GUID>".

Example

```
http://10.9.9.9:18000/Channels/GetActiveSegment?
channel=68db056e-6ab7-4898-a6cd-5c888422606d
```

Typical Response

In this response, the active segment is returned.

```
{
  "Description":null,
  "Identifier":"f269a2b3-c8e5-41a6-8855-118de2c3b94e",
  "Name":"SCTE Ad Break Segment"
}
```

SetActiveSegment

The purpose of this POST operation is to immediately start streaming the specified segment, stopping streaming of the current active segment. The channel must be running—that is, you must have started the channel. The specified segment may be the current segment being streamed—no error is returned.

Note: The segment must be activated directly on the target server.

Typical use cases for manually changing the active segment include correcting a state that is determined to be incorrect, and manually (or with server-side logic) changing the segment to the correct segment.

Another use case: If you or your system determines that the next media segment is one you don't want to broadcast, you can switch from the segment you want to avoid to a different segment (even a “We'll be right back” segment, with color bars).

SetActiveSegment has the following format:

```
http://<host>:<port>/Channels/SetActiveSegment
?channel={CHANNEL GUID}&segment={SEGMENT GUID}
```

Operation Sequence

Execute the following operations to obtain the required GUIDs for this operation:

GetChannels (channel GUID)

and

GetPrograms (program GUID) > *GetRenditions* (rendition GUID) > *GetSegments* (segment GUID)

Parameters

Parameter	Description
channel	GUID; string that identifies this channel.
segment	GUID; string that identifies this segment.

Results

On success, *SetActiveSegment* returns the success message: "Segment successfully triggered".

If the channel is not active, an error is returned: "Channel is not active".

Example

```
http://10.0.25.158:18000/Channels/SetActiveSegment
?channel=ad1c45b7-67fb-419d-8c5b-8ba474bd6dfd
&segment=1d20e392-8876-411a-9681-70e08e7baca9
```

Typical Response

In this response, the Live Stream system reported the successful operation: "Segment successfully triggered".

SetActiveSegmentAtTime

This POST operation schedules a segment to start broadcasting at a specified time for the current date. The channel must be running—you must have started the channel. This operation does not remove configured triggers for any segments. If the channel is stopped and started again, it will start at the base segment of the source again.

The specified source may be the current source being streamed—no error is returned.

Note: The segment must be activated directly on the target server.

SetActiveSegmentAtTime has the following format:

```
http://<host>:<port>/Channels/SetActiveSegment
?channel={CHANNEL GUID}&segment={SEGMENT GUID}&time={HH:MM:SS}
```

Operation Sequence

Execute the following operations to obtain the required GUIDs for this operation:

GetChannels (channel GUID)
and
GetPrograms (program GUID) > *GetRenditions* (rendition GUID) > *GetSegments* (segment GUID)

Parameters

Parameter	Description
channel	GUID; string that identifies this channel.
segment	GUID; string that identifies this segment.
time	Time code (in format HH:MM:SS 24-hour time) at which the segment should begin.

Results

On success, *SetActiveSegmentAtTime* returns a status message: "Successfully added trigger". If the channel is not active, an error is returned: "Channel is not active".

Example

```
http://10.0.25.158:18000/Channels/SetActiveSegmentAtTime
?channel=ad1c45b7-67fb-419d-8c5b-8ba474bd6dfd
&segment=1d20e392-8876-411a-9681-70e08e7baca9
&time=10:40:00
```

Typical Response

In this response, the returned message reports the successful operation: "Successfully added trigger".

GetMachineStatistics

The purpose of this GET operation is to obtain statistical information about the target server. Statistics are only available when at least one channel on the target server is streaming.

GetMachineStatistics has the following format:

```
http://<host>:<port>/Channels/GetMachineStatistics?
machine={MACHINE GUID}
```

Operation Sequence

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

GetMachines (machine GUID)

Parameters

Parameter	Description
machine	GUID; string that identifies the target Live Stream server.

Results

On success, *GetMachineStatistics* returns a record, with a variety of statistical values.

If the machine GUID does not exist or no channels are active, an error is returned: "Failed to generate statistics. No channels belonging to machine b6347bb5-c8c8-4bb1-8cec-53342ca6225d are currently active", for example.

Example

```
http://10.0.25.158:18000/Channels/Getmachinestatistics?
machine=af0f694b-5f17-4b8f-af13-34486d488012
```

Typical Response


```
{
  "Description":null,
  "Identifier":"da716ac0-d5e2-4d18-a718-533b42f6457a",
  "Name":"QA-VL-LIVE-9",
  "CpuUsage":15,
  "DiskSpaceUsed":"88",
  "GpuComputeUtilization":34,
  "GpuEncoderUtilization":0,
  "GpuMemory":13,
  "Memory":15
}
```

GetChannelStatistics

The purpose of this GET operation is to obtain statistical information about the target active channel.

GetChannelStatistics has the following format:

```
http://<host>:<port>/Channels/GetChannelStatistics
?channel={CHANNEL GUID}
```

Operation Sequence

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

GetChannels (channel GUID)

Parameters

Parameter	Description
channel	GUID; string that identifies this channel.

Results

On success, *GetChannelStatistics* returns a record with a variety of statistical values.

If the channel GUID does not exist, an error is returned: "Could not find channel with identifier 68db056e-6ab7-4898-a6cd-5c888422606d", for example.

If the channel is not active, an error is returned: "Channel is not active".

Example

```
http://10.0.25.158:18000/Channels/GetChannelStatistics?
channel=68db056e-6ab7-4898-a6cd-5c888422606d
```

Typical Response


```
{
  "Description": null,
  "Identifier": "808f9b52-5ad4-4520-82ce-c3df5175c48f",
  "Name": "HLS_Test",
  "CpuUsage": 2,
  "GpuMemory": 0,
  "Memory": 1092,
  "OutputLocation": null,
  "Uptime": "0:00:24"
}
```

GetNewFacebookVerificationDeviceCode

The purpose of this GET operation is to obtain a new device code which you can use to authenticate a new Facebook user.

Note: After executing this operation, you must enter the URL returned to display the authentication page and use the device code returned to authenticate this use.

GetNewFacebookVerificationDeviceCode has the following format:

`http://<host>:<port>/Channels/GetNewFacebookVerificationDeviceCode`

There are no parameters required for this operation.

Results

On success, *GetNewFacebookVerificationDeviceCode* returns a record with a new device code.

Example

```
http://LSS:18000/Channels/GetNewFacebookVerificationDeviceCode
```

Typical Response

In this response, the record includes the new device code you can use to authenticate streaming.

```
{
  "Description": null,
  "Identifier": "00000000-0000-0000-0000-000000000000",
  "Name": "Facebook Authentication",
  "DeviceCode": "8TLMXR4D",
  "VerificationURI": "https://www.facebook.com/device"
}
```

AddFacebookChannel

The purpose of this POST operation is to add a new Facebook channel to the Live Stream system.

AddFacebookChannel has the following format:

```
http://<host>:<port>/Channels/AddFacebookChannel?
name={NAME}&package={package GUID}&userName={USERNAME}
```

or

```
http://<host>:<port>/Channels/AddFacebookChannel?
name={NAME}&package={package GUID}&deviceCode={DEVICECODE}
```

Note: After authenticating your account manually in Facebook, you must supply the device code for this operation one time, to authenticate your account using Live Stream as well. After executing this operation one time, you should always use the userName parameter, supplying the user name of the account.

Operation Sequence

Execute the following operations to obtain the required GUIDs for this operation:

GetPackages (package GUID)

and

GetNewFacebookVerificationDeviceCode (device code). This device code must be used directly in Facebook before being used in this operation.

Parameters

Parameter	Description
name	String that identifies this channel; displayed in the web app.
package	String; GUID that identifies the primary package; obtained from *GetPackages*.
userName	String that identifies the user of an authenticated Facebook account. When using userName, do not use deviceCode.
deviceCode	Code returned to Live Stream as a result of a *GetNewFacebookVerificationDeviceCode* operation; required to authenticate a new Facebook account one time. When authenticating, do not include userName.

Results

On success, *AddFacebookChannel* returns a channel record.

If the device code you use is not valid, this error is returned: “An error occurred while creating channel: Invalid Facebook verification device code”.

If the user you supply is not authenticated, this error is returned: “Penelope_N45PLJ is not authenticated for Facebook Live”, for example.

Example

```
http://LS-SVR:18000/Channels/AddFacebookChannel?
name=FBChannel007&package=1af0ba42-de0e-48d0-8fa4-db14401e5bda&
userName=LiveStreamer&deviceCode=J24EMZRY
```

Typical Response

In this response, the Channel record reports the GUID details of the new *FBChannel007* channel added to the Live Stream system.


```
{
  "Description": null,
   "Identifier": "b01c8be8-6c0e-492e-ba1c-12d78d77687f",
   "Name": "FBChannel007",
   "Active": "false",
   "Assignments": "",
   "CalendarEvents":
  {
      "Briefs": ""
  },
   "Machine": "4b6f71b5-65dd-4df1-a4fb-209139737c21",
   "PrimaryPackage":
    {
      "Description": null,
      "Identifier": "90ad1d8f-d10b-425e-b39d-a7e447cc766b",
      "Name": "FacebookLive Outputs",
      "Details": {
         "ParameterBrief": {
            "Name": "User Name",
            "Value": "LiveStreamer"
        }
      },
      "Package": "1af0ba42-de0e-48d0-8fa4-db14401e5bda"
  },
   "SecondaryPackage": ""
}
```

ConfigureFacebookChannel

The purpose of this POST operation is to update the settings of an existing Facebook channel.

ConfigureFacebookChannel has the following format:

```
http://<host>:<port>/Channels/ConfigureFacebookChannel?
channel={CHANNEL GUID}&startDate={STARTDATE}&title={TITLE}&
description={DESCRIPTION}&postToPage={POSTTOPAGE}&
postToGroup={POSTTOGROUP}&postToEvent={POSTTOEVENT}&
deleteVideo={DELETEVIDEO}&privacy={PRIVACY}
```

Operation Sequence

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

GetChannels (channel GUID)

Parameters

Parameter	Description
channel	GUID; string that identifies this Facebook channel.
startDate (optional)	String; identifies the date and time to start the channel. If GMT is not specified, the time is local to the server. Support is provided for date/time strings that follow a recognized pattern, as described in the Microsoft .net DateTime.Parse method (for details, see https://msdn.microsoft.com/en-us/library/system.datetime.parse.aspx#StringToParse). For example, MM-DD-YYYY HH:MM:SS or Thu, 01 May 2017 07:34:42 GMT. If a startDate is not included, the schedule will be set to *Go Live Now*. If the startDate is included, the schedule is set to *Schedule for Later*.
title (optional)	String; displayed as the title of the Facebook Live event.
description (optional)	String; description displayed as the description of the event.
postToPage postToGroup postToEvent (optional)	The postToPage, postToGroup, and postToEvent parameters are mutually exclusive. Use only one (or none) to select where to post the video. The value of the parameter is a string of the target page, group, or event name. If none of these parameters are present, Post To is set to User.
deleteVideo (optional)	Keywords: True | False to specify whether the video should be deleted after broadcast.
privacy (optional)	Keywords: Public | Friends | Only Me to specify the privacy level for this broadcast. This is only relevant if the Post To parameter is set to *User*. When posting to Pages, Groups, and Events, privacy is *Public*.

Results

On success, *ConfigureFacebookChannel* returns a record with the channel details.

If the Channel is not a Facebook channel, an error is returned: “Channel's primary package is not a Facebook package”.

Example

```
http://10.0.25.158:18000/Channels/Configurefacebookchannel?
channel=90e90139-cace-4d10-8024-6533b1b74edd&deleteVideo=true
```

Typical Response


```
{
  "Description":null,
  "Identifier":"90e90139-cace-4d10-8024-6533b1b74edd",
  "Name":"FBChannel",
  "Active":"false",
  "Assignments":"",
  "CalendarEvents":{
    "Briefs":""
  },
  "Machine":"4b6f71b5-65dd-4df1-a4fb-209139737c21",
  "PrimaryPackage":{
    "Description":null,
    "Identifier":"90ad1d8f-d10b-425e-b39d-a7e447cc766b",
    "Name":"FacebookLive Outputs",
    "Details":{
      "ParameterBrief":[
        {
          "Name":"User Name",
          "Value":"EJBobick"
        },
        {
          "Name":"Schedule",
          "Value":"Go Live Now"
        },
        {
          "Name":"Event Title",
          "Value":""
        },
        {
          "Name":"Broadcast Description (Optional)",
          "Value":""
        },
        {
          "Name":"Delete Video from Facebook After Broadcast",
          "Value":"True"
        },
        {
          "Name":"StreamId",
          "Value":""
        },
        {
          "Name":"Status",
```

```
            "Value":""
          },
          {
            "Name":"Stream URL",
            "Value":""
          },
          {
            "Name":"Stream Name",
            "Value":""
          },
          {
            "Name":"Secure Stream URL",
            "Value":""
          },
          {
            "Name":"Secure Stream Name",
            "Value":""
          },
          {
            "Name":"Post To",
            "Value":"User"
          },
          {
            "Name":"Privacy",
            "Value":""
          }
        ]
      },
      "Package":"1af0ba42-de0e-48d0-8fa4-db14401e5bda"
    },
    "SecondaryPackage":""
}
```