



HAIVISION

KB Encoder/Transcoder 5.6
API Integrator's Reference

HVS-ID-API-KB-5.6

Edition Notice

© 2015-2023 Haivision. All rights reserved.

This edition and the products it describes contain proprietary and confidential information. No part of this content may be copied, photocopied, reproduced, translated or reduced to any electronic or machine-readable format without prior written permission of Haivision. If this content is distributed with software that includes an end-user agreement, this content and the software described in it, are furnished under license and may be used or copied only in accordance with the terms of that license. Except as permitted by any such license, no part of this content may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, electronic, mechanical, recording, or otherwise, without the prior written permission of Haivision Systems, Inc. Please note that the content is protected under copyright law even if it is not distributed with software that includes an end-user license agreement.

About Haivision

Founded in 2004, Haivision is now a market leader in enterprise video and video streaming technologies. We help the world's top organizations communicate, collaborate and educate. Recognized as one of the most influential companies in video by Streaming Media and one of the fastest growing companies by Deloitte's Technology Fast 500, organizations big and small rely on Haivision solutions to deliver video. Headquartered in Montreal, Canada, and Chicago, USA, we support our global customers with regional offices located throughout the United States, Europe, Asia and South America.

Trademarks

The Haivision logo, Haivision, and certain other marks are trademarks of Haivision. CoolSign is a registered trademark licensed to Haivision Systems, Inc. All other brand or product names identified in this document are trademarks or registered trademarks of their respective companies or organizations.

Disclaimer

The information contained herein is subject to change without notice. Haivision assumes no responsibility for any damages arising from the use of this content, including but not limited to, lost revenue, lost data, claims by third parties, or other damages.

If you have comments or suggestions, please contact infodev@haivision.com.

While every effort has been made to provide accurate and timely information regarding this product and its use, Haivision Systems Inc. shall not be liable for errors or omissions contained herein or for incidental or consequential damages in connection with the furnishing, performance, or use of this material.

Contents

Edition Notice	2
About Haivision	2
Trademarks	2
Disclaimer	2
Contents	3
About This Document	6
Conventions	6
Typographic Conventions and Elements	6
Action Alerts	6
Obtaining Documentation	7
Getting Service Support	7
Introducing the KB REST API	8
Reasons to Use the KB API	8
Installing the API	8
KB API follows REST Principles	9
Identifying KB Resources by their URIs	9
Resource URI Format	9
Sample URIs	10
A Resource URI is Specified in Two Parts	10
Setting Up a Connection to the Target Transcoder	10
Understanding KB API Requests and Responses	11
Encryption	11
API Requests	11
API Responses	13
API Request and Response Reference	16
Authentication	17
Login	17
Logout	18
Managing Licenses	19
Get License Parameter	19
Reset License	20
Get License State	22
Managing Encoders and Transcoders	23
Get Version	23
Get System Information	24
Get System Time	30
Get ECS State	31
Managing Channels	34
Get Channel List	34
Create Channel	35
Get Channel State	37
Delete Channel	45
Controlling a Channel	47
Input Control	49
Output Control	51
Get Channel Signal List	52
Get Video Preview	55
Inject Metadata	57
Inject AdCue Messages	60

Managing Channel Configurations.....	63
Get Channel Configurations	63
Review a Channel's Configuration.....	67
Load a Channel's Configuration.....	71
Get Configuration Label.....	72
Error Response Messages	74
Create Channel	74
License Violations during Channel Initialization.....	74
Signals Reference	75
ECS Signals	75
Channel Signals	75
Command Line Interface	80
Using the Command Line Interface.....	80
Available Command Line Switches	80
Channel Configuration File Syntax Reference	82
File Structure.....	83
File Structure Example.....	83
Component Objects	84
Processing	85
Input Object.....	85
Input (Common)	86
Input > Format.....	86
Input > Audio_Processing	87
Input > Video_Processing	87
Input > Source (Common)	88
Input > Source (Common for RTMP and MPEG_TS).....	88
Input > Source (RTMP).....	89
Input > Source (MPEG-TS).....	89
Input > Source (DeckLink Baseband).....	90
Input > Source (AJA Baseband).....	92
Input > Source (Default_Source)	92
Preview Object	92
Video_encoder Object	93
Video_encoder > Format	93
Video_encoder > Processing	93
Video_encoder > Encoder	94
Audio_encoder Object	97
Audio_encoder > Format	97
Audio_encoder > Processing.....	98
Audio_encoder > Encoder.....	98
Output Object.....	99
Output > Parameters (Common)	99
Output > Parameters (RTMP).....	100
Output > Parameters (HLS).....	102
Output > Parameters (File).....	105
Output > Parameters (MPEG TS).....	107
Output > Parameters (MPEG DASH)	109
Global Encoder Parameters	110
Warranties	112
1-Year Limited Hardware Warranty.....	112
EXCLUSIONS AND LIMITATIONS	112
OBTAINING WARRANTY SERVICE.....	113
APPLICABLE LAW	113
EULA - End User License Agreement.....	114
READ BEFORE USING	114
SLA - Service Level Agreement.....	114
1. Introduction.....	114
2. Definitions.....	114

3. Service Levels for the Video Content Management System	114
4. Exceptions to Availability for the VCMS	115
5. Credits for Downtime for the VCMS.....	116
6. Support Services for the VCMS	116
7. Service Levels for Haivision Streaming Media Service	117
8. Credits for Outages of Haivision Streaming Media Service	117
9. No Secondary End User Support	117

Getting Help

118

About This Document

Conventions

The following conventions are used to help clarify the content.

Typographic Conventions and Elements

<i>Italics</i>	Used for the introduction of new terminology, for words being used in a different context, and for placeholder or variable text.
bold	Used for strong emphasis and items that you click, such as buttons.
Monospaced	Used for code examples, command names, options, responses, error messages, and to indicate text that you enter.
>	In addition to a math symbol, it is used to indicate a submenu. For instance, File > New where you would select the New option from the File menu.
...	Indicates that text is being omitted for brevity.

Action Alerts

The following alerts are used to advise and counsel that special actions should be taken.



Tip

Indicates highlights, suggestions, or helpful hints.



Note

Indicates a note containing special instructions or information that may apply only in special cases.



Important

Indicates an emphasized note. It provides information that you should be particularly aware of in order to complete a task and that should not be disregarded. This alert is typically used to prevent loss of data.

⚠ Caution

Indicates a potentially hazardous situation which, if not avoided, may result in damage to data or equipment. It may also be used to alert against unsafe practices.

⚠ Warning

Indicates a potentially hazardous situation that may result in physical harm to the user.

Obtaining Documentation

This document was generated from the Haivision InfoCenter. To ensure you are reading the most up-to-date version of this content, access the documentation online at <https://doc.haivision.com>. You may generate a PDF at any time of the current content. See the footer of the page for the date it was generated.

Getting Service Support

For more information regarding service programs, training courses, or for assistance with your support requirements, contact Haivision Technical Support using our Support Portal at: <https://support.haivision.com>.

Introducing the KB REST API

The KB API is the application programming interface that allows you to create custom applications for the KB Encoder. For an introduction to the KB Encoder and related KB technologies, see the *Installation Guide* and *User Guide*. For API reference documentation, see [API Request and Response Reference](#).

Topics Discussed

- [Reasons to Use the KB API](#)
- [Installing the API](#)
- [KB API follows REST Principles](#)
- [Identifying KB Resources by their URIs](#)
 - [Resource URI Format](#)
 - [Sample URIs](#)
 - [A Resource URI is Specified in Two Parts](#)
- [Setting Up a Connection to the Target Transcoder](#)
- [Understanding KB API Requests and Responses](#)
 - [Encryption](#)
 - [API Requests](#)
 - [HTTP Methods for Requests](#)
 - [Required Authentication](#)
 - [HTTP Header for Requests](#)
 - [Sample Request](#)
 - [Request Messages](#)
 - [API Responses](#)
 - [HTTP Header for Responses](#)
 - [Sample Response](#)
 - [Response Messages](#)

Reasons to Use the KB API

Companies that have purchased KB Encoder typically use the KB API to:

- Manage workflows.
- Integrate KB into pre-existing applications that have been specifically designed to manage the entire workflow of live streaming for a particular company.
- Simplify repetitive tasks or access data from KB that is tedious to retrieve or update with the web interface.
- Create a custom application to interface to KB. Some custom applications work in tandem with the web interface or other KB interfaces. Others are designed as the company's only interface to KB.

Installing the API

Usually, the KB API is already installed:

- KB API is part of the KB software and is installed along with every install of the KB transcoding software. This is true whether you are using the software to provide encoding or transcoding

services. All KB Internet Encoders provided by Haivision come with the KB Transcoding software (including the KB API) already installed.

- There are no client-side libraries for the KB API. No KB software needs to be installed on developer workstations. Users of custom software that use KB API do not need any KB software installed on their workstations.

KB API follows REST Principles

The KB API is based on REST, which is an approach to designing APIs that allow client programs to request services from server programs running on remote computers. At a high level, REST concepts are similar to RPC and SOAP. Like most REST-based APIs, the KB API is implemented as a set of HTTPS requests (sometimes referred to as functions or methods).

APIs that conform to REST principles are often called *RESTful* APIs. The KB API, like most RESTful APIs, is not implemented as a set of classes or functions that must be linked to your application. Instead, the KB API is implemented as a set of HTTPS requests and responses that are compliant with HTTP/1.1. Accordingly, the API can be called from almost any programming language.

Identifying KB Resources by their URIs

The KB API manages resources on the KB encoders. Resources include channel configurations, events, licenses, etc. A resource is identified by its Uniform Resource Identifier (URI).

Resource URI Format

The resource URI represents your KB resource with which you want to interact.

A resource URI has the following format:

```
https://[host]:[port]/ecs/[component]/[resource]?[query]
```

where:

- `[host]:[port]` is the hostname or IP address appended with a suffix of a colon and the port. For example, `YourServer.com:1080` or `127.0.0.1:1080`.
- `/ecs/[component]` is `*.xml` or `*.json` to indicate the response content-type or the content root, including one of the following components:
 - version
 - channels
 - events
 - license
 - system
 - time
- `/[resource]` is the hierarchically-organized part of the resource identification.
- `?[query]` is the part of the resource identification consisting of parameters that are passed during the request.

Sample URIs

URI	Description
<code>https://[host]:[port]/ecs/channels.xml</code>	The plural noun indicates that channels is a <i>collection</i> , that is, a resource that contains zero or more <i>elements</i> .
<code>https://[host]:[port]/ecs/channels/Channel_1.xml</code>	Channel_1 is contained by channels, indicating that Channel_1 is an element of channels.
<code>https://[host]:[port]/ecs/version.xml</code>	This resource is neither a collection nor an element of a collection.
<code>https://[host]:[port]/ecs/version.xml</code>	/ecs/ is prefixed to all KB resources.
<code>https://[host]:[port]/ecs/version.xml</code> l -or- <code>https://[host]:[port]/ecs/version.js</code> on	When sending a request that sends a string of data to a resource, postfix either: <ul style="list-style-type: none"> .xml to indicate the string of data is formatted in XML. .json to indicate the string of data is formatted in JSON. When retrieving data from a resource, postfix either: <ul style="list-style-type: none"> .xml to indicate the response be formatted in XML. .json to indicate the response be formatted in JSON.

A Resource URI is Specified in Two Parts

When coding a KB request, the resource URI is specified in two parts:

- The *base URL* is the URL of the computer hosting the KB transcoder (for example `[protocol]://[host]:[port]`). This string is not shown in any of the sample requests.
- The *relative path*, starting from the base URL. This string is shown in the sample requests.

For example, the following explains how a request to delete a resource appears in this document:

```
DELETE /ecs/channels/Channel_1.xml
```

In your own application, insert your base URL right before the relative path. For example,

```
DELETE https://127.0.01:1080/ecs/channels/Channel_1.xml
```

-or-

```
DELETE https://YourServer.com:1080/ecs/channels/Channel_1.xml
```

See [Sample Request](#) for more details about this sample request.

Setting Up a Connection to the Target Transcoder

Before you can send requests to a transcoder or your application, you must define a TCP/IP connection to port 1080 of the target encoder/transcoder. The URL of that transcoder becomes the *base URL* of KB requests.

Tip

By default, the TCP/IP connection is opened and closed for every request sent. If you prefer that the connection be kept open across multiple requests, you can add the header, "Connection: keep-alive" to specify that the connection be kept open.

Understanding KB API Requests and Responses

KB API is not a set of classes or functions that must be linked to your application. Instead, it is a set of *requests* that your application can send to an encoder.

Upon receiving a request from your application, the KB transcoder:

- Attempts to carry out the request.
- Sends a *response* to your application that reports the success or failure of the request.

Encryption

The KB REST API interface only supports TLS/SSL (HTTPS) connections. By default, a self-signed certificate is used. However, it is possible to configure your own certificate in the authentication configuration of the KB UI. See [Supplying KB with a TLS/SSL Security Certificate](#) in the *User's Guide* for details.

API Requests

The following sections explain the KB API requests and provides samples of how they are documented in [API Request and Response Reference](#).

HTTP Methods for Requests

The KB API expresses requests using four common methods of HTTP: GET, POST, PUT, and DELETE.

Method	Description	Examples
GET	Retrieves a resource, that is, data from a specified KB transcoder. Does not modify any data stored on KB.	Get Version , Get Channel List , and Get Configuration Label
POST	Creates a resource on a specified transcoder.	Login and Create Channel
PUT	Updates a resource on a specified transcoder.	Load a Channel's Configuration , Controlling a Channel , Inject Metadata
DELETE	Deletes or disables a resource on a specified transcoder.	Reset License and Delete Channel

Required Authentication

Starting with version 4.3.1, the KB REST API requires user authentication. To implement authentication, two new requests have been added:

- Login request—If successful, returns a Session ID. It is this Session ID that identifies subsequent requests as authenticated. Session IDs expire 30 minutes after the last call was made or when a Logout request is received.
- Logout request—Expires a Session ID so that it is no longer valid.

! Important

Every request, other than the Login request, must include a valid Session ID in its HTTP header as described in [HTTP Header for Requests](#).

HTTP Header for Requests

The header must include the following for all KB REST requests:

```
Content-Type: application/json
```

```
Content-Length: nnnn
```

where, Content-Type is either "application/xml" or "application/json", and Content-Length is the size of the entity-body, in decimal number of OCTETs. These lines are removed from the sections in [API Request and Response Reference](#) for brevity.

Also, the following line must be added to the HTTP header for all KB REST requests, *except* the Login request.

```
Authorization: [Session ID]
```

where, Session ID is the string that was received in response to the Login request. See [Authentication](#).

Sample Request

The following is a sample of the KB API requests listed in [API Request and Response Reference](#). As described in [HTTP Header for Requests](#), the Content-Type and Content-Length headers are omitted from the requests throughout the rest of the book, due to the fact that these headers are familiar to most readers.

As mentioned previously, JSON and XML requests are supported.

[JSON](#) [XML](#)

Requests

```
DELETE /ecs/channels/<channel_id>.json
Authorization: [Session ID]
```

Parameters

Name	Type	Description
channel_id	<i>string</i>	Label, or name, for the channel.
Authorization	<i>[Session ID]</i>	Required on all requests except the Login request. This is the Session ID that was received in response to the Login request. See Authentication .

JSON XML

Requests

```
DELETE /ecs/channels/<channel_id>.xml
Authorization: [Session ID]
```

Parameters

Name	Type	Description
channel_id	<i>string</i>	Label, or name, for the channel.
Authorization	<i>[Session ID]</i>	Required on all requests except the Login request. This is the Session ID that was received in response to the Login request. See Authentication .

In this instance, you would type the command as shown above, replacing the highlighted parameters with values appropriate to your particular implementation. Also, as mentioned in [A Resource URI is Specified in Two Parts](#), the base URL is removed for brevity. For instance, if your base URL is YourServer.com and your channel is named channelA, you would enter either:

```
DELETE https://YourServer.com:1080/ecs/channels/channelA.json
```

-or-

```
DELETE https://YourServer.com:1080/ecs/channels/channelA.xml
```

Request Messages

All POST and PUT requests described in [API Request and Response Reference](#) require additional data to be sent in the body of the request. For example, the [Create Channel](#) request requires the user to send information about the new channel: label, id, and previewOnly. This information is coded in either XML or JSON depending on which extension is used in the request.

API Responses

The following sections explain the KB API responses and provides samples of how they are documented in [API Requests and Responses](#).

HTTP Header for Responses

Each response includes the following HTTP header:

```
HTTP/1.1 200 OK
Content-Type: application/json
Content-Length: nnnn
```

where, the string following HTTP/1.1 is the status code (see [HTTP 1.1 specification](#) for details), Content-Type is either "application/xml" or "application/json", and Content-Length is the size of the entity-body, in decimal number of OCTETs. Since these headers are familiar to most readers, they are only documented here. Throughout the rest of the book, these headers are omitted from the response output.

Sample Response

Each request provided in [API Request and Response Reference](#) includes a sample response. The following illustrates a typical sample response. As described in [HTTP Header for Responses](#), the HTTP Status, Content-Type, and Content-Length headers are omitted from the responses throughout the rest of the book, due to the fact that these headers are familiar to most readers.

The following includes an example response for the [Get Channel List](#) request. As mentioned previously, JSON and XML responses are supported.

[JSON](#) [XML](#)

Response

```
{
  "channels": {
    "channel": [
      {
        "id": "5775626faa602b6d251cf1d6",
        "label": "TestChannel2",
        "outoforder": false,
        "port": 2133,
        "startcounter": 1
      }
    ],
    "count": 1
  }
}
```

[JSON](#) [XML](#)

Response

```
<?xml version="1.0"?>
<channels count="1">
  <channel label="TestChannel2" id="5775626faa602b6d251cf1d6" startcounter="1"
outoforder="0" port="2133"/>
</channels>
```

Response Messages

The KB API responds to requests with either a custom or standard response message:

- A custom response is returned for all GET requests and the Login's POST request. For example, the response to a [Get Version](#) request contains version number information: major, minor, subminor, build, and type. See the individual request's response sections in [API Request and Response Reference](#) for details on these responses.
- Standard information or error messages are returned for all other POST, DELETE, and PUT requests. These messages indicate the success or failure of completing the request. The response sections for these requests in [API Request and Response Reference](#) only list the request successful message. For failure messages, additional KB-specific messages are provided to provide you more details on the error.

An example standard response is shown below. As mentioned previously, JSON and XML responses are supported.

[JSON](#) [XML](#)

Response

```
{
  "messages": {
    "count": 1,
    "message": [
      {
        "text": "Request processed successfully.",
        "type": "info"
      }
    ]
  }
}
```

[JSON](#) [XML](#)

Response

```
<?xml version="1.0"?>
<messages count="1">
  <message type="info">Request processed successfully.</message>
</messages>
```

where:

- count is the number of messages included in this response.
- type is the message type: info, warning, or error.
- text (JSON) or text within the message element (XML) is a detailed text description of the message.

A complete list of possible error messages is [Error Response Messages](#).

API Request and Response Reference

This chapter contains reference information for KB requests and their corresponding responses. For an overview, see [Introducing the KB REST API](#).

! Important

Any reference to an ID is to the GUID, or the 128-bit hex number grouped as 8-4-4-4-12 for a KB resource. For example, a reference to a `channel_id` is actually to the Channel GUID. For example, `{21EC2020-3AEA-1069-A2DD08002B30309D}`.

! Important

Every KB REST API request, other than the Login request, must include a valid Session ID in its HTTP header. When you issue the Login request, the Session ID value is returned in the response.

Topics Discussed

- [Authentication](#)
 - [Login](#)
 - [Logout](#)
- [Managing Licenses](#)
 - [Get License Parameter](#)
 - [Reset License](#)
 - [Get License State](#)
- [Managing Encoders and Transcoders](#)
 - [Get Version](#)
 - [Get System Information](#)
 - [Get System Time](#)
 - [Get ECS State](#)
- [Managing Channels](#)
 - [Get Channel List](#)
 - [Create Channel](#)
 - [Get Channel State](#)
 - [Delete Channel](#)
 - [Controlling a Channel](#)
 - [Input Control](#)
 - [Output Control](#)
 - [Get Channel Signal List](#)
 - [Get Video Preview](#)
 - [Inject Metadata](#)
 - [Inject AdCue Messages](#)
- [Managing Channel Configurations](#)
 - [Get Channel Configurations](#)
 - [Review a Channel's Configuration](#)
 - [Load a Channel's Configuration](#)

- [Get Configuration Label](#)

Authentication

The following authentication functions are available:

- [Login](#)
- [Logout](#)

Login

The login requires a user name and a password. The configured users are the same as configured for the KB UI web application. When you issue the Login request, the Session ID value is returned in the response.

Active for Version: 4.3.1+

Authorizations: Administrator, Operator, Observer

[JSON](#) [XML](#)

Requests

```
POST /ecs/auth.json

{
  "username" : "[user name]",
  "password" : "[password]"
}
```

Parameters

Name	Type	Description
username	<i>string</i>	Valid user.
password	<i>string</i>	Valid password.

Response

The login request returns a session id on success. Otherwise it will return a 403 HTTP error.

```
{
  "sessionid" : "[session id]"
}
```

[JSON](#) [XML](#)

Requests

```
POST /ecs/auth.xml

<?xml version= "1.0" ?>
<user username= "[user name]" password= "[password]" />
```

Parameters

Name	Type	Description
<user>	—	
↳ username	string	Valid user.
password	string	Valid password.

Response

The login request returns a session id on success. Otherwise it will return a 403 HTTP error.

```
<?xml version="1.0"?>
<sessionid value="[session id]" />
```

Logout

The logout is a delete request. The Session ID must be set in the header.

Active for Version: 4.3.1+

Authorizations: Administrator, Operator, Observer

[JSON](#) [XML](#)

Requests

```
DELETE /ecs/auth.json
Authorization: [Session ID]
```

Parameters

Name	Type	Description
Authorization	[Session ID]	The <i>sessionid</i> value returned in the response to a Login request.

Response

N/A

[JSON](#) [XML](#)

Requests

```
DELETE /ecs/auth.xml
Authorization: [Session ID]
```

Parameters

Name	Type	Description
Authorization	<i>string</i>	The <i>sessionid</i> value returned in the response to a Login request.

Response

N/A

Managing Licenses

The following licensing functions are available:

- [Get License Parameter](#)
- [Reset License](#)
- [Get License State](#)

Get License Parameter

Returns the current license information of the KB encoder.

Active for Version: 4.0+

Authorizations: Administrator

[JSON](#) [XML](#)

Requests

```
GET /ecs/license.json
Authorization: [Session ID]
```

Parameters

Name	Type/Value	Description
Authorization	<i>[Session ID]</i>	The sessionid value returned in the response to a Login request.

Response

```
{
  "license": {
    "channels": 4,
    "demo_mode": false,
    "expired": false,
    "no_watermark": true,
    "start_day": 10,
    "start_hour": 22,
    "start_minute": 9,
    "start_month": 7,
    "start_second": 35,
    "start_year": 2016,
    "stop_day": 17,
    "stop_hour": 0,
    "stop_minute": 0,
    "stop_month": 7,
    "stop_second": 0,
    "stop_year": 2016,
    "time_limit": 0,
    "time_zone": "",
    "valid": true
  }
}
```

JSON [XML](#)

Requests

```
GET /ecs/license.xml
Authorization: [Session ID]
```

Parameters

Name	Type/Value	Description
Authorization	[Session ID]	The sessionid value returned in the response to a Login request.

Response

```
<?xml version="1.0" encoding="UTF-8"?>
<license valid="1" expired="0" demo_mode="0" no_watermark="1" time_limit="0" channels="4"
time_zone="" start_year="2016" start_month="7" start_day="10"
start_hour="22" start_minute="9" start_second="35" stop_year="2016" stop_month="7"
stop_day="17" stop_hour="0" stop_minute="0" stop_second="0"/>
```

Reset License

Removes the configured license parameter and data.

Active for Version: 4.0+

Authorizations: Administrator

[JSON](#) [XML](#)

Requests

```
DELETE /ecs/license.json
Authorization: [Session ID]
```

Parameters

Name	Type	Description
Authorization	[Session ID]	The sessionid value returned in the response to a Login request.

Response

```
{
  "messages": {
    "count": 1,
    "message": [
      {
        "text": "Request processed successfully.",
        "type": "info"
      }
    ]
  }
}
```

[JSON](#) [XML](#)

Requests

```
DELETE /ecs/license.xml
Authorization: [Session ID]
```

Parameters

Name	Type	Description
Authorization	[Session ID]	The sessionid value returned in the response to a Login request.

Response

```
<?xml version="1.0"?>
<messages count="1">
  <message type="info">Request processed successfully.</message>
</messages>
```

Get License State

This request delivers information about the current license state of the encoder.

Active for Version: 4.0 - 4.3.1

Authorizations: Administrator

[JSON](#) [XML](#)

Requests

```
GET /ecs/license/state.json
Authorization: [Session ID]
```

Parameters

Name	Type	Description
Authorization	[Session ID]	The sessionid value returned in the response to a Login request.

Response

```
{
  "license_state": {
    "mac": "D4:AE:52:B9:3E:E9",
    "registered_channels": 2,
    "type": "key"
  }
}
```

[JSON](#) [XML](#)

Requests

```
GET /ecs/license/state.xml
Authorization: [Session ID]
```

Parameters

Name	Type	Description
Authorization	[Session ID]	The sessionid value returned in the response to a Login request.

Response

```
<?xml version="1.0" encoding="UTF-8"?>
<license_state type="key" registered_channels="1" mac="00:01:2E:56:43:18"/>
```

Managing Encoders and Transcoders

Haivision sells and supports a range of KB Internet Encoders and KB Live Transcoders. These are "appliances" with hardware and software configurations suited to a variety of encoding and transcoding needs (see haivision.com).

Note

ECS, or Encoder Communication Server, is a program running on the KB encoder system that manages the creation and deletion of encoder channel processes and the communication with them. The ECS is the program that accepts and responds to API REST calls for a given server.

ECS is responsible for launching channels. A channel is a process that:

- Ingests an encoded stream.
- Processes the stream in various ways, like encoding, transcoding, and cropping.
- Outputs the resulting channel process either to an IP network or to a recording on disk.

Get Version

Returns the version, CPU, and memory information regarding the ECS program running on the KB encoder/transcoder.

Active for Version: 4.1+

Authorizations: Administrator, Operator, Observer

[JSON](#) [XML](#)

Requests

```
GET /ecs/version.json
Authorization: [Session ID]
```

Parameters

Name	Type	Description
Authorization	[Session ID]	The sessionid value returned in the response to a Login request.

Response

```
{
  "version" : {
    "build" : "406",
    "major" : "5",
    "minor" : "0",
    "subminor" : "0",
    "type" : "DEV"
  }
}
```

[JSON](#) [XML](#)

Requests

```
GET /ecs/version.xml
Authorization: [Session ID]
```

Parameters

Name	Type	Description
Authorization	[Session ID]	The sessionid value returned in the response to a Login request.

Response

```
<?xml version="1.0"?>
<version major="5" minor="0" subminor="0" build="406" type="DEV"/>
```

Get System Information

Returns information about CPU, memory, installed Decklink cards, and the uptime of the ECS.

Active for Version: 4.1+

Authorizations: Administrator, Operator, Observer

[JSON](#) [XML](#)

Requests

```
GET /ecs/system.json
Authorization: [Session ID]
```

Parameters

Name	Type	Description
Authorization	[Session ID]	The sessionid value returned in the response to a Login request.

Response

Example response with a Blackmagic Decklink card installed:

```
{
  "system": {
    "capturesources": {
      "aja_detected": false,
      "ajacard": [],
      "decklink_detected": true,
      "decklinkcard": [
        {
          "audio_input_connections": "Embedded",
          "audio_output_connections": "Embedded",
          "device_name": "DeckLink SDI",
          "display_name": "DeckLink SDI (1)",

```



```
"max_audio_channels": 8,
"number_of_subdevices": 2,
"subdevice_index": 0,
"supports_input_format_detection": true,
"video_input_connections": "SDI",
"video_output_connections": "SDI",
"videomodes": [
  {
    "format": "480i29.97",
    "fps": 29.97,
    "height": 480,
    "interlaced": true,
    "width": 720
  },
  {
    "format": "480i23.98",
    "fps": 23.98,
    "height": 480,
    "interlaced": true,
    "width": 720
  },
  {
    "format": "576i25",
    "fps": 25,
    "height": 576,
    "interlaced": true,
    "width": 720
  },
  {
    "format": "1080p23.98",
    "fps": 23.98,
    "height": 1080,
    "interlaced": false,
    "width": 1920
  },
  {
    "format": "1080p24",
    "fps": 24,
    "height": 1080,
    "interlaced": false,
    "width": 1920
  },
  {
    "format": "1080p25",
    "fps": 25,
    "height": 1080,
    "interlaced": false,
    "width": 1920
  },
  {
    "format": "1080p29.97",
    "fps": 29.97,
    "height": 1080,
    "interlaced": false,
    "width": 1920
  },
  {
    "format": "1080p30",
    "fps": 30,
    "height": 1080,
    "interlaced": false,
    "width": 1920
  },
  {
    "format": "1080i50",
```

```
    "fps": 50,
    "height": 1080,
    "interlaced": true,
    "width": 1920
  },
  {
    "format": "1080i59.94",
    "fps": 59.94,
    "height": 1080,
    "interlaced": true,
    "width": 1920
  },
  {
    "format": "1080i60",
    "fps": 60,
    "height": 1080,
    "interlaced": true,
    "width": 1920
  },
  {
    "format": "720p50",
    "fps": 50,
    "height": 720,
    "interlaced": false,
    "width": 1280
  },
  {
    "format": "720p59.94",
    "fps": 59.94,
    "height": 720,
    "interlaced": false,
    "width": 1280
  },
  {
    "format": "720p60",
    "fps": 60,
    "height": 720,
    "interlaced": false,
    "width": 1280
  }
]
},
{
  "audio_input_connections": "Embedded",
  "audio_output_connections": "Embedded",
  "device_name": "DeckLink SDI",
  "display_name": "DeckLink SDI (2)",
  "max_audio_channels": 8,
  "number_of_subdevices": 2,
  "subdevice_index": 1,
  "supports_input_format_detection": true,
  "video_input_connections": "SDI",
  "video_output_connections": "SDI",
  "videomodes": [
    {
      "format": "480i29.97",
      "fps": 29.97,
      "height": 480,
      "interlaced": true,
      "width": 720
    },
    {
      "format": "480i23.98",
      "fps": 23.98,
      "height": 480,
```

```
    "interlaced": true,
    "width": 720
  },
  {
    "format": "576i25",
    "fps": 25,
    "height": 576,
    "interlaced": true,
    "width": 720
  },
  {
    "format": "1080p23.98",
    "fps": 23.98,
    "height": 1080,
    "interlaced": false,
    "width": 1920
  },
  {
    "format": "1080p24",
    "fps": 24,
    "height": 1080,
    "interlaced": false,
    "width": 1920
  },
  {
    "format": "1080p25",
    "fps": 25,
    "height": 1080,
    "interlaced": false,
    "width": 1920
  },
  {
    "format": "1080p29.97",
    "fps": 29.97,
    "height": 1080,
    "interlaced": false,
    "width": 1920
  },
  {
    "format": "1080p30",
    "fps": 30,
    "height": 1080,
    "interlaced": false,
    "width": 1920
  },
  {
    "format": "1080i50",
    "fps": 50,
    "height": 1080,
    "interlaced": true,
    "width": 1920
  },
  {
    "format": "1080i59.94",
    "fps": 59.94,
    "height": 1080,
    "interlaced": true,
    "width": 1920
  },
  {
    "format": "1080i60",
    "fps": 60,
    "height": 1080,
    "interlaced": true,
    "width": 1920
  }
}
```

```
    },
    {
      "format": "720p50",
      "fps": 50,
      "height": 720,
      "interlaced": false,
      "width": 1280
    },
    {
      "format": "720p59.94",
      "fps": 59.94,
      "height": 720,
      "interlaced": false,
      "width": 1280
    },
    {
      "format": "720p60",
      "fps": 60,
      "height": 720,
      "interlaced": false,
      "width": 1280
    }
  ]
}
]
},
"cores_count": 8,
"cpu": [
  {
    "cache_size": "8192 KB",
    "cpu_cores": "4",
    "cpu_speed": "3716.562",
    "cpuid_level": "13",
    "model": "58",
    "model_name": "Intel(R) Xeon(R) CPU E3-1270 V2 @ 3.50GHz",
    "physical_id": "0",
    "vendor_id": "GenuineIntel"
  }
],
"cpu_count": 1,
"gpu": {
  "hwencoder": {
    "h264": false,
    "hevc": false
  }
},
"label": "localhost.localdomain",
"memory": {
  "mem_total": "3872260 kB"
},
"network_adapters": {
  "adapter": [
    {
      "address": "10.67.12.128",
      "mac": "d4:ae:52:b9:3e:e9",
      "name": "em1"
    }
  ]
},
"uptime": {
  "hours": 12
}
}
```

JSON [XML](#)

Requests

```
GET /ecs/system.xml
Authorization: [Session ID]
```

Parameters

Name	Type	Description
Authorization	[Session ID]	The sessionid value returned in the response to a Login request.

Response

Example response with a Blackmagic Decklink card installed:

```
<?xml version="1.0"?>
<system label="localhost.localdomain" cpu_count="1" cores_count="8">
  <cpu vendor_id="GenuineIntel" model="58" model_name="Intel(R) Xeon(R) CPU E3-1270 V2 @
3.50GHz"
cpu_speed="3727.910" cache_size="8192 KB" physical_id="0" cpu_cores="4" cpuid_level="13"/>
  <gpu>
    <hwencoder h264="0" hevc="0"/>
  </gpu>
  <memory mem_total="3872260 kB"/>
  <uptime hours="12"/>
  <network_adapters>
    <adapter name="em1" address="10.67.12.128" mac="d4:ae:52:b9:3e:e9"/>
  </network_adapters>
  <capturesources decklink_detected="1">
    <decklinkcard device_name="DeckLink SDI" display_name="DeckLink SDI (1)"
max_audio_channels="8"
number_of_subdevices="2" subdevice_index="0" supports_input_format_detection="1"
video_input_connections="SDI"
video_output_connections="SDI" audio_input_connections="Embedded"
audio_output_connections="Embedded">
      <videomodes>
        <mode format="480i29.97" width="720" height="480" fps="29.97" interlaced="1"/>
      >
        <mode format="480i23.98" width="720" height="480" fps="23.98" interlaced="1"/>
      >
        <mode format="576i25" width="720" height="576" fps="25" interlaced="1"/>
        <mode format="1080p23.98" width="1920" height="1080" fps="23.98"
interlaced="0"/>
        <mode format="1080p24" width="1920" height="1080" fps="24" interlaced="0"/>
        <mode format="1080p25" width="1920" height="1080" fps="25" interlaced="0"/>
        <mode format="1080p29.97" width="1920" height="1080" fps="29.97"
interlaced="0"/>
        <mode format="1080p30" width="1920" height="1080" fps="30" interlaced="0"/>
        <mode format="1080i50" width="1920" height="1080" fps="50" interlaced="1"/>
        <mode format="1080i59.94" width="1920" height="1080" fps="59.94"
interlaced="1"/>
        <mode format="1080i60" width="1920" height="1080" fps="60" interlaced="1"/>
        <mode format="720p50" width="1280" height="720" fps="50" interlaced="0"/>
        <mode format="720p59.94" width="1280" height="720" fps="59.94"
interlaced="0"/>
        <mode format="720p60" width="1280" height="720" fps="60" interlaced="0"/>
      </videomodes>
    </decklinkcard>
  </capturesources>
</system>
```

```

        </decklinkcard>
        <decklinkcard device_name="DeckLink SDI" display_name="DeckLink SDI (2)"
max_audio_channels="8"
number_of_subdevices="2" subdevice_index="1" supports_input_format_detection="1"
video_input_connections="SDI"
video_output_connections="SDI" audio_input_connections="Embedded"
audio_output_connections="Embedded">
        <videomodes>
                <mode format="480i29.97" width="720" height="480" fps="29.97" interlaced="1"/>
        >
                <mode format="480i23.98" width="720" height="480" fps="23.98" interlaced="1"/>
        >
                <mode format="576i25" width="720" height="576" fps="25" interlaced="1"/>
                <mode format="1080p23.98" width="1920" height="1080" fps="23.98"
interlaced="0"/>
                <mode format="1080p24" width="1920" height="1080" fps="24" interlaced="0"/>
                <mode format="1080p25" width="1920" height="1080" fps="25" interlaced="0"/>
                <mode format="1080p29.97" width="1920" height="1080" fps="29.97"
interlaced="0"/>
                <mode format="1080p30" width="1920" height="1080" fps="30" interlaced="0"/>
                <mode format="1080i50" width="1920" height="1080" fps="50" interlaced="1"/>
                <mode format="1080i59.94" width="1920" height="1080" fps="59.94"
interlaced="1"/>
                <mode format="1080i60" width="1920" height="1080" fps="60" interlaced="1"/>
                <mode format="720p50" width="1280" height="720" fps="50" interlaced="0"/>
                <mode format="720p59.94" width="1280" height="720" fps="59.94"
interlaced="0"/>
                <mode format="720p60" width="1280" height="720" fps="60" interlaced="0"/>
        </videomodes>
        </decklinkcard> aja_detected="0"
</capturesources>
</system>
    
```

Get System Time

Returns the time information for the ECS program running on the KB encoder/transcoder. Each encoder/transcoder runs only one instance of ECS.

Active for Version: 1.0+

Authorizations: Administrator, Operator, Observer

[JSON](#) [XML](#)

Requests

```

GET /ecs/time.json
Authorization: [Session ID]
    
```

Parameters

Name	Type	Description
Authorization	[Session ID]	The sessionid value returned in the response to a Login request.

Response

```
{
  "time": {
    "localtime": {
      "day": 11,
      "hour": 22,
      "minute": 5,
      "month": 7,
      "second": 41,
      "year": 2016
    },
    "utctime": {
      "day": 12,
      "hour": 3,
      "minute": 5,
      "month": 7,
      "second": 41,
      "year": 2016
    }
  }
}
```

JSON [XML](#)

Requests

```
GET /ecs/time.xml
Authorization: [Session ID]
```

Parameters

Name	Type	Description
Authorization	[Session ID]	The sessionid value returned in the response to a Login request.

Response

```
<?xml version="1.0"?>
<time>
  <localtime year="2016" month="7" day="11" hour="22" minute="6" second="6"/>
  <utctime year="2016" month="7" day="12" hour="3" minute="6" second="6"/>
</time>
```

Get ECS State

Note

An XML response isn't supported for this request.

Active for Version: 4.5+

Authorizations: Administrator, Operator, Observer

[JSON](#) [XML](#)

Requests

```
GET /ecs.json
Authorization: [Session ID]
```

Parameters

Name	Type	Description
Authorization	[Session ID]	The sessionid value returned in the response to a Login request.

Response

```
{
  "encoder": {
    "CurrentCpuUsage": "32.78",
    "CurrentMemUsage": "23.73",
    "FreeDiskSpaceGB": "403.792",
    "SystemMemUsage": "2452.82",
    "SystemMemUsagePercent": "65",
    "SystemTotalMemory": "3781.50",
    "TotalDiskSpaceGB": "403.827",
    "UsedDiskSpacePercent": "0",
    "app": "kulabyte",
    "build": "406",
    "channels": {
      "channel": [
        {
          "event_id": "",
          "event_label": "",
          "id": "5775626faa602b6d251cf1d6",
          "label": "TestChannel2",
          "outoforder": false,
          "port": 2094,
          "startcounter": 1
        }
      ],
      "count": 1
    },
    "gpu": {
      "hwencoder": {
        "h264": false,
        "hevc": false
      },
      "metrics": {
        "blitter": "0.0",
        "renderer": "0.0",
        "valid": false,
        "video": "0.0",
        "videoenhancement": "0.0"
      },
      "usage": {
        "hevc": "0.0"
      }
    },
    "label": "localhost.localdomain",
    "license": {
      "channels": 4,
```



```
"demo_mode": false,
"expired": false,
"mac": "D4:AE:52:B9:3E:E9",
"no_watermark": true,
"registered_channels": 1,
"start_day": 10,
"start_hour": 21,
"start_minute": 59,
"start_month": 7,
"start_second": 35,
"start_year": 2016,
"stop_day": 17,
"stop_hour": 0,
"stop_minute": 0,
"stop_month": 7,
"stop_second": 0,
"stop_year": 2016,
"time_limit": 0,
"time_zone": "",
"type": "key",
"valid": true
},
"major": "5",
"minor": "0",
"network": {
  "in": "0.032",
  "out": "0.037",
  "total": "0.069"
},
"subminor": "0",
"temperature": "29.8",
"time": {
  "localtime": {
    "day": 11,
    "hour": 22,
    "minute": 4,
    "month": 7,
    "second": 9,
    "year": 2016
  },
  "utctime": {
    "day": 12,
    "hour": 3,
    "minute": 4,
    "month": 7,
    "second": 9,
    "year": 2016
  }
},
"type": "DEV",
"uptime": {
  "hours": 12
}
}
```

[JSON](#) [XML](#)

XML Request Not Supported

Managing Channels

A channel is a process dedicated to ingesting one input signal, processing it in various ways, and outputting the processed signal to a network and/or recording. This section describes creating and controlling channels. [Managing Channel Configurations](#) describes configuring the channel.

Get Channel List

Requests a list of the channels currently running on the encoder or created via the [Create Channel](#) command.

Active for Version: 1.0+

Authorizations: Administrator, Operator, Observer

Note

Use the [Get Channel Configurations](#) command to see all channels available in the web interface, regardless if they are running or stopped or created via the API or web interface.

[JSON](#) [XML](#)

Requests

```
GET /ecs/channels.json
Authorization: [Session ID]
```

Parameters

Name	Type	Description
Authorization	[Session ID]	The sessionid value returned in the response to a Login request.

Response

```
{
  "channels": {
    "channel": [
      {
        "id": "57740998aa602b6d251cf1d5",
        "label": "TestChannel1",
        "outoforder": false,
        "port": 2088,
        "startcounter": 2
      },
      {
        "id": "5775626faa602b6d251cf1d6",
        "label": "TestChannel2",
        "outoforder": false,
        "port": 2174,
        "startcounter": 1
      }
    ]
  },
  "count": 2
}
```

JSON [XML](#)

Requests

```
GET /ecs/channels.xml
Authorization: [Session ID]
```

Parameters

Name	Type	Description
Authorization	[Session ID]	The sessionid value returned in the response to a Login request.

Response

```
<?xml version="1.0"?>
<channels count="2">
  <channel label="TestChannel1" id="57740998aa602b6d251cf1d5" startcounter="2" outoforder="0"
port="2088"/>
  <channel label="TestChannel2" id="5775626faa602b6d251cf1d6" startcounter="1" outoforder="0"
port="2174"/>
</channels>
```

Create Channel

Creates a new channel with a specified channel label on an encoder. If the request can not be fulfilled, the ECS replies with an error message.

Active for Version: 1.0+

Authorizations: Administrator, Operator

[JSON](#) [XML](#)

Requests

```
POST /ecs/channels.json
Authorization: [Session ID]

{
  "channel" : {
    "label" : "Channel_1",
    "id" : "...",
    "previewonly" : "0"
  }
}
```

Parameters

Name	Type	Description
Authorization	[Session ID]	The sessionid value returned in the response to a Login request.
channel	—	
label	string	Unique label, or name, of the new channel. For example, "Channel_1."
id	string	<i>Optional.</i> Contains a unique id for a channel. The id must not contain any not-allowed character for URIs. Usually a GUID should be used here. If this parameter doesn't exist or is empty, an id will be created.
previewonly	string	<i>Optional.</i> If "1", a preview-only channel will be created. A preview-only channel doesn't increase the locked channels of the encoder licenses. But only an input is allowed in the channel configuration, no encoders or outputs.

Response

```
{
  "messages": {
    "count": 1,
    "message": [
      {
        "text": "Resource created successfully.",
        "type": "info"
      }
    ]
  }
}
```

[JSON](#) [XML](#)

Requests

```
POST /ecs/channels.xml
Authorization: [Session ID]

<?xml version="1.0"?>
<channel label="Channel_1" id="..." previewonly="0"/>
```

Parameters

Name	Type	Description
Authorization	[Session ID]	The sessionid value returned in the response to a Login request.
<channel>	—	
label	string	Unique label, or name, of the new channel. For example, "Channel_1."
id	string	Optional. Contains a unique id for a channel. The id must not contain any not-allowed character for URIs. Usually a GUID should be used here. If this parameter doesn't exist or is empty, an id will be created.
previewonly	string	Optional. If "1", a preview-only channel will be created. A preview-only channel doesn't increase the locked channels of the encoder licenses. But only an input is allowed in the channel configuration, no encoders or outputs.

Response

```
<?xml version="1.0"?>
<messages count="1">
  <message type="info">Resource created successfully.</message>
</messages>
```

Get Channel State

Retrieve all state data from the channel specified.

Active for Version: 1.0+

Authorizations: Administrator, Operator, Observer

[JSON](#) [XML](#)

Requests

```
GET /ecs/channels/<channel_id>.json
Authorization: [Session ID]
```

Parameters

Name	Type	Description
channel_id	<i>string</i>	Label, or name, of the channel. For example, "Channel_1."
Authorization	<i>[Session ID]</i>	The sessionid value returned in the response to a Login request.

Response

Click here to expand JSON response...

```
{
  "channel": {
    "App": "kulabyte",
    "CurrentCpuUsage": "14.52",
    "CurrentMemUsage": "829.41",
    "CurrentMemUsagePercent": "22",
    "CurrentVMemUsage": "5088.30",
    "CurrentVMemUsagePercent": "135",
    "ElapsedRunningTime": "00:00:07",
    "MediaSampleUsage": "87",
    "NetworkInTraffic": "1.858",
    "NetworkOutTraffic": "2.051",
    "OutOfOrder": "0",
    "PresetLabel": "TestChannel2",
    "PresetState": "fully_loaded",
    "RecordState": "idle",
    "SystemCpuUsage": "15.72",
    "Threads": "110",
    "encoders": {
      "count": "6",
      "encoder": [
        {
          "Implementation": "Software",
          "InputVideoQueue": "0",
          "VideoCodec": "H264",
          "VideoEncoderBitrate": "1372000",
          "VideoEncoderFps": "29.970",
          "VideoFrameHeight": "540",
          "VideoFrameWidth": "960",
          "comp": "AVC_ENC",
          "id": "AVC_ENC_6",
          "label": "AVC_ENC_6",
          "state": "enabled",
          "type": "videoencoder"
        },
        {
          "Implementation": "Software",
          "InputVideoQueue": "0",
          "VideoCodec": "H264",
          "VideoEncoderBitrate": "1172000",
          "VideoEncoderFps": "29.970",
          "VideoFrameHeight": "480",
          "VideoFrameWidth": "848",
          "comp": "AVC_ENC",
          "id": "AVC_ENC_7",
          "label": "AVC_ENC_7",
          "state": "enabled",
          "type": "videoencoder"
        },
        {
          "Implementation": "Software",
          "InputVideoQueue": "0",
```

```
    "VideoCodec": "H264",
    "VideoEncoderBitrate": "772000",
    "VideoEncoderFps": "29.970",
    "VideoFrameHeight": "360",
    "VideoFrameWidth": "640",
    "comp": "AVC_ENC",
    "id": "AVC_ENC_8",
    "label": "AVC_ENC_8",
    "state": "enabled",
    "type": "videoencoder"
  },
  {
    "Implementation": "Software",
    "InputVideoQueue": "0",
    "VideoCodec": "H264",
    "VideoEncoderBitrate": "372000",
    "VideoEncoderFps": "29.970",
    "VideoFrameHeight": "270",
    "VideoFrameWidth": "480",
    "comp": "AVC_ENC",
    "id": "AVC_ENC_9",
    "label": "AVC_ENC_9",
    "state": "enabled",
    "type": "videoencoder"
  },
  {
    "Implementation": "Software",
    "InputVideoQueue": "0",
    "VideoCodec": "H264",
    "VideoEncoderBitrate": "172000",
    "VideoEncoderFps": "29.970",
    "VideoFrameHeight": "180",
    "VideoFrameWidth": "320",
    "comp": "AVC_ENC",
    "id": "AVC_ENC_10",
    "label": "AVC_ENC_10",
    "state": "enabled",
    "type": "videoencoder"
  },
  {
    "AudioBitDepth": "16",
    "AudioChannelConfig": "1:0:1.00|2:1:1.00",
    "AudioChannels": "2",
    "AudioCodec": "AAC",
    "AudioEncoderBitrate": "128000",
    "AudioLanguageCode": "eng",
    "AudioLanguageComment": "",
    "AudioLanguageName": "English",
    "AudioLanguageType": "512",
    "AudioSampleRate": "48000",
    "AudioStreamNumber": "-1",
    "InputAudioQueue": "0",
    "comp": "AAC_ENC",
    "id": "AAC_ENC_eng_11",
    "label": "AAC_ENC_eng_11",
    "state": "enabled",
    "type": "audioencoder"
  }
]
},
"id": "5775626faa602b6d251cf1d6",
"inputs": {
  "active": "Device_DeckLink_Baseband_2",
  "count": "2",
  "input": [
```

```
{
  "AudioGainValue": "0.000",
  "AudioLevel": "-25.6;-23.3;-100.0;-100.0;-100.0;-100.0;-100.0;-100.0",
  "CC": "",
  "ConfiguredAudio": "1",
  "ConfiguredAudioBitDepth": "16",
  "ConfiguredAudioChannels": "2",
  "ConfiguredAudioSampleRate": "48000",
  "ConfiguredVideo": "1",
  "ConfiguredVideoFrameRate": "29.97",
  "ConfiguredVideoHeight": "1080",
  "ConfiguredVideoWidth": "1920",
  "DetectedAudio": "1",
  "DetectedAudioBitDepth": "16",
  "DetectedAudioChannels": "8",
  "DetectedAudioSampleRate": "48000",
  "DetectedVideo": "1",
  "DetectedVideoFrameRate": "29.97",
  "DetectedVideoHeight": "1080",
  "DetectedVideoWidth": "1920",
  "IBS": "",
  "Implementation": "Software",
  "InputAudioChannels": "8",
  "InputAudioConnector": "Embedded",
  "InputAudioEncoding": "PCM",
  "InputAudioSampleRate": "48000",
  "InputDisplayName": "DeckLink SDI (1)",
  "InputSignal": "good",
  "InputVideoConnector": "SDI",
  "InputVideoEncoding": "YUY2",
  "InputVideoFrameRate": "29.970",
  "InputVideoHeight": "1080",
  "InputVideoWidth": "1920",
  "OutputAudioSampleRate": "48000",
  "OutputVideoFrameRate": "29.97",
  "Position": "topleft",
  "SCTE35State": "disabled",
  "SignalLosses": "0",
  "SupportsAudioGainControl": "1",
  "UsedAudioChannels": "0;1;2;3;4;5;6;7",
  "comp": "Device_DeckLink_Baseband",
  "id": "Device_DeckLink_Baseband_2",
  "label": "Device_DeckLink_Baseband_2"
},
{
  "AudioLevel": "-100.0;-100.0",
  "ConfiguredAudio": "1",
  "ConfiguredAudioBitDepth": "16",
  "ConfiguredAudioChannels": "2",
  "ConfiguredAudioSampleRate": "48000",
  "ConfiguredVideo": "1",
  "ConfiguredVideoFrameRate": "29.97",
  "ConfiguredVideoHeight": "1080",
  "ConfiguredVideoWidth": "1920",
  "DetectedAudio": "1",
  "DetectedAudioBitDepth": "0",
  "DetectedAudioChannels": "0",
  "DetectedAudioSampleRate": "0",
  "DetectedVideo": "1",
  "DetectedVideoFrameRate": "29.97",
  "DetectedVideoHeight": "1080",
  "DetectedVideoWidth": "1920",
  "Implementation": "Software",
  "InputAudioEncoding": "",
  "InputVideoEncoding": "",

```



```
    "OutputAudioSampleRate": "0",
    "OutputVideoFrameRate": "0.00",
    "Position": "topleft",
    "SCTE35State": "disabled",
    "SignalLosses": "0",
    "SupportsAudioGainControl": "0",
    "UsedAudioChannels": "0;1",
    "comp": "Default_Source",
    "id": "Default_Source_3",
    "label": "Default_Source_3"
  }
]
},
"label": "TestChannel2",
"outputs": {
  "count": "2",
  "output": [
    {
      "AudioEncoder": "AAC_ENC_eng_11",
      "AudioLanguageCodes": "eng",
      "AudioLanguageComments": "",
      "AudioLanguageNames": "English",
      "CCLanguageCodes": "eng|||",
      "CCLanguageComments": "|||",
      "CCLanguageNames": "English|||",
      "DefaultAudioLanguage": "",
      "DefaultVideo": "",
      "MasterPlaylistURL": "",
      "Media": "AV",
      "OutputAudioQueues": "65",
      "OutputAudioSampleRates": "48000",
      "OutputVideoFrameRates": "29.97|29.97|29.97|29.97|29.97",
      "OutputVideoQueues": "0|0|1|5|7",
      "PlaybackURL": "",
      "StreamName": "",
      "StreamPlaylistURL": "",
      "StreamURL": "http://hls.server.address",
      "VideoEncoder": "AVC_ENC_6|AVC_ENC_7|AVC_ENC_8|AVC_ENC_9|AVC_ENC_10",
      "WaitingSegmentsCount": "0",
      "comp": "HLS_Sink",
      "id": "HLS_Sink_12",
      "label": "HLS_Sink_12",
      "name": "HLS output",
      "state": "disconnected",
      "type": "http"
    },
    {
      "ArchiveDirectory": "/assets/archive/2016-07-11_21-24-46",
      "ArchiveFilename": "",
      "AudioEncoder": "AAC_ENC_eng_11",
      "AudioLanguageCodes": "eng",
      "AudioLanguageComments": "",
      "AudioLanguageNames": "English",
      "CCLanguageCodes": "eng|||",
      "CCLanguageComments": "|||",
      "CCLanguageNames": "English|||",
      "FreeDiskSpace": "403764",
      "FreeDiskSpacePercent": "100",
      "Media": "AV",
      "OutputAudioQueue": "68",
      "OutputAudioSampleRate": "0",
      "OutputVideoFrameRate": "0.00",
      "OutputVideoQueue": "0",
      "TotalDiskSpace": "403826.723",
      "UsedDiskSpace": "62.887",
    }
  ]
}
```

```

        "VideoEncoder": "AVC_ENC_6",
        "WriteRate": "0.629",
        "comp": "File",
        "id": "File_13",
        "label": "File_13",
        "name": "",
        "state": "idle",
        "type": "archive"
    }
  ],
  "state": "running",
  "type": "multistream"
}
}

```

JSON [XML](#)

Requests

```

GET /ecs/channels/<channel_id>.xml
Authorization: [Session ID]

```

Parameters

Name	Type	Description
channel_id	string	Label, or name, of the channel. For example, "Channel_1."
Authorization	[Session ID]	The sessionid value returned in the response to a Login request.

Response

Click here to expand XML response...

```

<?xml version="1.0" encoding="UTF-8"?>
<channel type="multistream" id="5775626faa602b6d251cf1d6" state="running"
label="TestChannel2">
  <PresetLabel value="TestChannel2"/>
  <PresetState value="fully_loaded"/>
  <RecordState value="idle"/>
  <OutOfOrder value="0"/>
  <App value="kulabyte"/>
  <CurrentCpuUsage value="30.10"/>
  <SystemCpuUsage value="31.00"/>
  <CurrentMemUsage value="877.89"/>
  <CurrentVMemUsage value="5280.30"/>
  <CurrentMemUsagePercent value="23"/>
  <CurrentVMemUsagePercent value="140"/>
  <Threads value="111"/>
  <MediaSampleUsage value="89"/>
  <NetworkInTraffic value="0.858"/>
  <NetworkOutTraffic value="0.936"/>
  <ElapsedRunningTime value="00:01:31"/>
  <inputs active="Device_DeckLink_Baseband_2" count="2">
    <input id="Device_DeckLink_Baseband_2" label="Device_DeckLink_Baseband_2"
comp="Device_DeckLink_Baseband">
      <OutputVideoFrameRate value="29.97"/>

```

```

<OutputAudioSampleRate value="48000"/>
<SupportsAudioGainControl value="1"/>
<AudioGainValue value="0.000"/>
<AudioLevel value="-31.9;-32.3;-100.0;-100.0;-100.0;-100.0;-100.0;-100.0"/>
<UsedAudioChannels value="0;1;2;3;4;5;6;7"/>
<ConfiguredVideo value="1"/>
<ConfiguredAudio value="1"/>
<ConfiguredVideoFrameRate value="29.97"/>
<ConfiguredVideoWidth value="1920"/>
<ConfiguredVideoHeight value="1080"/>
<ConfiguredAudioSampleRate value="48000"/>
<ConfiguredAudioChannels value="2"/>
<ConfiguredAudioBitDepth value="16"/>
<DetectedVideo value="1"/>
<DetectedAudio value="1"/>
<DetectedVideoWidth value="1920"/>
<DetectedVideoHeight value="1080"/>
<DetectedVideoFrameRate value="29.97"/>
<DetectedAudioSampleRate value="48000"/>
<DetectedAudioChannels value="8"/>
<DetectedAudioBitDepth value="16"/>
<InputVideoEncoding value="YUY2"/>
<Implementation value="Software"/>
<InputAudioEncoding value="PCM"/>
<SignalLosses value="0"/>
<Position value="topleft"/>
<SCTE35State value="disabled"/>
<InputVideoFrameRate value="29.970"/>
<InputVideoWidth value="1920"/>
<InputVideoHeight value="1080"/>
<InputVideoConnector value="SDI"/>
<InputAudioSampleRate value="48000"/>
<InputAudioChannels value="8"/>
<InputAudioConnector value="Embedded"/>
<InputSignal value="good"/>
<IBS value=""/>
<CC value=""/>
<InputDisplayName value="DeckLink SDI (1)"/>
</input>
<input id="Default_Source_3" label="Default_Source_3" comp="Default_Source">
<OutputVideoFrameRate value="0.00"/>
<OutputAudioSampleRate value="0"/>
<SupportsAudioGainControl value="0"/>
<AudioLevel value="-100.0;-100.0"/>
<UsedAudioChannels value="0;1"/>
<ConfiguredVideo value="1"/>
<ConfiguredAudio value="1"/>
<ConfiguredVideoFrameRate value="29.97"/>
<ConfiguredVideoWidth value="1920"/>
<ConfiguredVideoHeight value="1080"/>
<ConfiguredAudioSampleRate value="48000"/>
<ConfiguredAudioChannels value="2"/>
<ConfiguredAudioBitDepth value="16"/>
<DetectedVideo value="1"/>
<DetectedAudio value="1"/>
<DetectedVideoWidth value="1920"/>
<DetectedVideoHeight value="1080"/>
<DetectedVideoFrameRate value="29.97"/>
<DetectedAudioSampleRate value="0"/>
<DetectedAudioChannels value="0"/>
<DetectedAudioBitDepth value="0"/>
<InputVideoEncoding value=""/>
<Implementation value="Software"/>
<InputAudioEncoding value=""/>
<SignalLosses value="0"/>

```

```

        <Position value="topleft"/>
        <SCTE35State value="disabled"/>
    </input>
</inputs>
<encoder count="6">
    <videoencoder id="AVC_ENC_6" state="enabled" label="AVC_ENC_6" comp="AVC_ENC">
        <VideoEncoderFps value="29.970"/>
        <VideoEncoderBitrate value="1372000"/>
        <VideoFrameWidth value="960"/>
        <VideoFrameHeight value="540"/>
        <VideoCodec value="H264"/>
        <InputVideoQueue value="0"/>
        <Implementation value="Software"/>
    </videoencoder>
    <videoencoder id="AVC_ENC_7" state="enabled" label="AVC_ENC_7" comp="AVC_ENC">
        <VideoEncoderFps value="29.970"/>
        <VideoEncoderBitrate value="1172000"/>
        <VideoFrameWidth value="848"/>
        <VideoFrameHeight value="480"/>
        <VideoCodec value="H264"/>
        <InputVideoQueue value="0"/>
        <Implementation value="Software"/>
    </videoencoder>
    <videoencoder id="AVC_ENC_8" state="enabled" label="AVC_ENC_8" comp="AVC_ENC">
        <VideoEncoderFps value="29.970"/>
        <VideoEncoderBitrate value="772000"/>
        <VideoFrameWidth value="640"/>
        <VideoFrameHeight value="360"/>
        <VideoCodec value="H264"/>
        <InputVideoQueue value="0"/>
        <Implementation value="Software"/>
    </videoencoder>
    <videoencoder id="AVC_ENC_9" state="enabled" label="AVC_ENC_9" comp="AVC_ENC">
        <VideoEncoderFps value="29.970"/>
        <VideoEncoderBitrate value="372000"/>
        <VideoFrameWidth value="480"/>
        <VideoFrameHeight value="270"/>
        <VideoCodec value="H264"/>
        <InputVideoQueue value="0"/>
        <Implementation value="Software"/>
    </videoencoder>
    <videoencoder id="AVC_ENC_10" state="enabled" label="AVC_ENC_10" comp="AVC_ENC">
        <VideoEncoderFps value="29.970"/>
        <VideoEncoderBitrate value="172000"/>
        <VideoFrameWidth value="320"/>
        <VideoFrameHeight value="180"/>
        <VideoCodec value="H264"/>
        <InputVideoQueue value="0"/>
        <Implementation value="Software"/>
    </videoencoder>
    <audioencoder id="AAC_ENC_eng_11" state="enabled" label="AAC_ENC_eng_11"
comp="AAC_ENC">
        <AudioEncoderBitrate value="128000"/>
        <AudioSampleRate value="48000"/>
        <AudioChannels value="2"/>
        <AudioBitDepth value="16"/>
        <AudioCodec value="AAC"/>
        <InputAudioQueue value="0"/>
        <AudioChannelConfig value="1:0:1.00|2:1:1.00"/>
        <AudioLanguageName value="English"/>
        <AudioLanguageCode value="eng"/>
        <AudioLanguageComment value=""/>
        <AudioLanguageType value="512"/>
        <AudioStreamNumber value="-1"/>
    </audioencoder>

```

```

</encoder>
<outputs count="2">
  <output type="http" id="HLS_Sink_12" state="disconnected" label="HLS_Sink_12"
comp="HLS_Sink">
    <Media value="AV"/>
    <name value="HLS output"/>
    <AudioEncoder value="AAC_ENC_eng_11"/>
    <AudioLanguageNames value="English"/>
    <AudioLanguageCodes value="eng"/>
    <AudioLanguageComments value=""/>
    <VideoEncoder value="AVC_ENC_6|AVC_ENC_7|AVC_ENC_8|AVC_ENC_9|AVC_ENC_10"/>
    <CCLanguageNames value="English|||"/>
    <CCLanguageCodes value="eng|||"/>
    <CCLanguageComments value="|||"/>
    <OutputVideoQueues value="0|0|1|6|8"/>
    <OutputAudioQueues value="67"/>
    <OutputVideoFrameRates value="29.97|29.97|29.97|29.97|29.97"/>
    <OutputAudioSampleRates value="48000"/>
    <StreamName value=""/>
    <StreamURL value="http://hls.server.address"/>
    <PlaybackURL value=""/>
    <WaitingSegmentsCount value="0"/>
    <StreamPlaylistURL value=""/>
    <MasterPlaylistURL value=""/>
    <DefaultVideo value=""/>
    <DefaultAudioLanguage value=""/>
  </output>
  <output type="archive" id="File_13" state="idle" label="File_13" comp="File">
    <Media value="AV"/>
    <name value=""/>
    <AudioEncoder value="AAC_ENC_eng_11"/>
    <AudioLanguageNames value="English"/>
    <AudioLanguageCodes value="eng"/>
    <AudioLanguageComments value=""/>
    <VideoEncoder value="AVC_ENC_6"/>
    <CCLanguageNames value="English|||"/>
    <CCLanguageCodes value="eng|||"/>
    <CCLanguageComments value="|||"/>
    <ArchiveFilename value=""/>
    <ArchiveDirectory value="/assets/archive/2016-07-11_21-24-46"/>
    <OutputVideoFrameRate value="0.00"/>
    <OutputAudioSampleRate value="0"/>
    <OutputVideoQueue value="0"/>
    <OutputAudioQueue value="68"/>
    <FreeDiskSpace value="403764"/>
    <WriteRate value="0.629"/>
    <FreeDiskSpacePercent value="100"/>
    <TotalDiskSpace value="403826.723"/>
    <UsedDiskSpace value="62.867"/>
  </output>
</outputs>
</channel>

```

Delete Channel

Closes and deletes a channel running on an encoder.

Active for Version: 1.0+

Authorizations: Administrator, Operator

[JSON](#) [XML](#)

Requests

```
DELETE /ecs/channels/<channel_id>.json
Authorization: [Session ID]
```

Parameters

Name	Type	Description
channel_id	string	Label, or name, of the channel. For example, "Channel_1."
Authorization	[Session ID]	The sessionid value returned in the response to a Login request.

Response

```
{
  "messages": {
    "count": 1,
    "message": [
      {
        "text": "Resource deleted successfully.",
        "type": "info"
      }
    ]
  }
}
```

JSON [XML](#)

Requests

```
DELETE /ecs/channels/<channel_id>.xml
Authorization: [Session ID]
```

Parameters

Name	Type	Description
channel_id	string	Label, or name, of the channel. For example, "Channel_1."
Authorization	[Session ID]	The sessionid value returned in the response to a Login request.

Response

```
<?xml version="1.0"?>
<messages count="1">
  <message type="info">Resource deleted successfully.</message>
</messages>
```

Controlling a Channel

Commands can be invoked to control the channels, including:

- Start channel
- Prepare stopping channel
- Stop channel
- Start recording
- Stop recording

Active for Version: 1.0+

Authorizations: Administrator, Operator

[JSON](#) [XML](#)

Requests

```
PUT /ecs/channels/<channel_id>.json
Authorization: [Session ID]

{
  "invoke" :
  {
    "command" : "<command>",
    "param" : "<parameter>"
  }
}
```

Parameters

Name	Type	Description
channel_id	string	Label, or name, of the channel. For example, "Channel_1."
Authorization	[Session ID]	The sessionid value returned in the response to a Login request.
invoke	—	
↳ command	<ul style="list-style-type: none"> • StartChannel • StopChannel • PrepareStop • StartRecord • StopRecord 	<ul style="list-style-type: none"> • StartChannel: Starts the channel. If the parameter contains "startrecord" all file outputs will start recording right from start. The old "StartEncode" command is still supported. • StopChannel: Stops the channel. Use PrepareStop prior to this command. The old "StopEncode" command is still supported. • PrepareStop: This command should be called right before "StopChannel". It stops processing received content in the input, delivers buffered content by the output protocols, and closes the recordings. This improves the stop behavior of a channel. • StartRecord: Starts recording in all file outputs. If the channel is not running it will return KULA_ERROR_WRONG_STATE. • StopRecord: Stops recording in all file outputs. If the channel is not running it will return KULA_ERROR_WRONG_STATE.
	param	string

Response

```
{
  "messages": {
    "count": 1,
    "message": [
      {
        "text": "Request processed successfully.",
        "type": "info"
      }
    ]
  }
}
```

JSON [XML](#)

Requests

```
PUT /ecs/channels/<channel_id>.xml
Authorization: [Session ID]

<?xml version="1.0"?>
<invoke command="<command>" param="<parameter>"/>
```

Parameters

Name	Type	Description
channel_id	string	Label, or name, of the channel. For example, "Channel_1."
Authorization	[Session ID]	The sessionid value returned in the response to a Login request.
invoke	—	
↳ command	<ul style="list-style-type: none"> StartChannel StopChannel PrepareStop StartRecord StopRecord 	<ul style="list-style-type: none"> StartChannel: Starts the channel. If the parameter contains "startrecord" all file outputs will start recording right from start. The old "StartEncode" command is still supported. StopChannel: Stops the channel. Use PrepareStop prior to this command. The old "StopEncode" command is still supported. PrepareStop: This command should be called right before "StopChannel". It stops processing received content in the input, delivers buffered content by the output protocols, and closes the recordings. This improves the stop behavior of a channel. StartRecord: Starts recording in all file outputs. If the channel is not running it will return KULA_ERROR_WRONG_STATE. StopRecord: Stops recording in all file outputs. If the channel is not running it will return KULA_ERROR_WRONG_STATE.
param	string	Optional parameter used by commands. See command description for usage details.

Response

```
<?xml version="1.0"?>
<messages count="1">
  <message type="info">Request processed successfully.</message>
</messages>
```

Input Control

Use to control various channel input attributes. Commands are invoked using the following requests.

Active for Version: 5.0+

Authorizations: Administrator, Operator

[JSON](#) [XML](#)

Requests

```
PUT /ecs/channels/<channel_id>/inputs/<input_id>.json
Authorization: [Session ID]
```

```
{
  "invoke" :
  {
    "command" : "<command>",
    "param" : "<parameter>"
  }
}
```

Parameters

Name	Type	Description
channel_id	<i>string</i>	Label, or name, of the channel. For example, "Channel_1."
input_id	<i>string</i>	Label, or name, of the input. For example, "Device_DeckLink_Baseband_2."
Authorization	<i>[Session ID]</i>	The sessionid value returned in the response to a Login request.
invoke	—	
↳	command	<ul style="list-style-type: none"> audiogaindec: Decreases the audio gain by 5.0 dB. audiogaininc: Increases the audio gain by 5.0 dB. setaudiogain: Directly set the audio gain to the dB value defined by the param string, e.g. - 10.0. Valid range is -20 to +40 dB.
	param	<i>string</i>

Response

```
{
  "messages": {
    "count": 1,
    "message": [
      {
        "text": "Request processed successfully.",
        "type": "info"
      }
    ]
  }
}
```


JSON [XML](#)

Requests

```
PUT /ecs/channels/<channel_id>/inputs/<input_id>.xml
Authorization: [Session ID]

<?xml version="1.0"?>
<invoke command="<command>" param="<parameter>"/>
```

Parameters

Name	Type	Description
channel_id	string	Label, or name, of the channel. For example, "Channel_1."
input_id	string	Label, or name, of the input. For example, "Device_DeckLink_Baseband_2."
Authorization	[Session ID]	The sessionid value returned in the response to a Login request.
invoke	—	
	command	<ul style="list-style-type: none"> audiogaindec: Decreases the input audio gain by 5.0 dB. audiogaininc: Increases the input audio gain by 5.0 dB. setaudiogain: Directly set the input audio gain to the dB value defined by the param string, e.g. -10.0. Valid range is -20 to +40.
	param	string

Response

```
<?xml version="1.0"?>
<messages count="1">
  <message type="info">Request processed successfully.</message>
</messages>
```

Output Control

Use to control various channel output attributes. Commands are invoked using the following requests.

Active for Version: 1.0+

Authorizations: Administrator, Operator


[JSON](#) [XML](#)

Requests

```
PUT /ecs/channels/<channel_id>/outputs/<encode_id>.json
Authorization: [Session ID]

{
  "invoke" :
  {
    "command" : "<command>",
    "param" : "<parameter>"
  }
}
```

Parameters

Name	Type	Description
channel_id	string	Label, or name, of the channel. For example, "Channel_1".
encode_id	string	The label of the output you specified when you created the channel. For example, "Output_1".
Authorization	[Session ID]	The sessionid value returned in the response to a Login request.
invoke	—	
	command	<ul style="list-style-type: none"> EnableOutput DisableOutput
	param	string

Response

```
{
  "messages": {
    "count": 1,
    "message": [
      {
        "text": "Request processed successfully.",
        "type": "info"
      }
    ]
  }
}
```


JSON XML

Requests

```
PUT /ecs/channels/<channel_id>/outputs/<encode_id>.xml
Authorization: [Session ID]

<?xml version="1.0"?>
<invoke command="<command>" param="<parameter>" />
```

Parameters

Name	Type	Description
channel_id	string	Label, or name, of the channel. For example, "Channel_1".
encode_id	string	The label of the output you specified when you created the channel. For example, "Output_1".
Authorization	[Session ID]	The sessionid value returned in the response to a Login request.
invoke	—	
	command	<ul style="list-style-type: none"> EnableOutput DisableOutput
	param	string

Response

```
<?xml version="1.0"?>
<messages count="1">
  <message type="info">Request processed successfully.</message>
</messages>
```

Get Channel Signal List

Requests a listing of all signals on a specified channel. For example, all the events on a specified channel.

- Option 1 - to get all signals acquired after a specified time.
- Option 2 - to get all signals acquired within the last nn seconds.
- Option 3 - to get all signals after specified index.

For a list of possible signals and their descriptions, see [Signals Reference](#).

Active for Version: 1.0+

Authorizations: Administrator, Operator, Observer

Requests

```
GET /ecs/channels/<channel_id>/signals.json?<option>="<option_value>"
Authorization: [Session ID]
```

Parameters

Name	Type	Description
channel_id	<i>string</i>	Label, or name, of the channel. For example, "Channel_1."
option	<ul style="list-style-type: none"> time timespan index 	<ul style="list-style-type: none"> time: Request all signals acquired after a specified time. timespan: Request all signals acquired in the last <i>nn</i> seconds. index: Returns all signals received after the provided index. Use -1 for the first request.
option_value	<ul style="list-style-type: none"> HH:MM:SS:mmm SS <i>int</i> 	<ul style="list-style-type: none"> HH:MM:SS:mmm: hours (in military time), minutes, seconds, and milliseconds respectively for the time option. SS: seconds for the timespan option. Integer value for the index option.
Authorization	<i>[Session ID]</i>	The sessionid value returned in the response to a Login request.

Response

```
{
  "signals": {
    "channel": "TestChannel2",
    "count": 4,
    "signal": [
      {
        "comp": "Device_DeckLink_Baseband",
        "date": "2016:07:12",
        "id": "Device_DeckLink_Baseband_2",
        "index": "3",
        "name": "Input Source",
        "param1": "48000:8:16",
        "param2": "",
        "time": "02:15:57:412",
        "type": "INPUT_AUDIO_FORMAT"
      },
      {
        "comp": "Device_DeckLink_Baseband",
        "date": "2016:07:12",
        "id": "Device_DeckLink_Baseband_2",
        "index": "2",
        "name": "Input Source",
        "param1": "1920:1080:29.97",
        "param2": "",
        "time": "02:15:57:412",
        "type": "INPUT_VIDEO_FORMAT"
      },
      {
        "comp": "Device_DeckLink_Baseband",
        "date": "2016:07:12",
        "id": "Device_DeckLink_Baseband_2",
        "index": "1",
        "name": "Input Source",
        "param1": "",
        "param2": "",
        "time": "02:15:57:015",
        "type": "INPUT_READYTODELIVER"
      },
      {
        "comp": "CONTROLLER",
        "date": "2016:07:12",
        "id": "",
        "index": "0",
        "name": "Channel",
        "param1": "TestChannel2",
        "param2": "",
        "time": "02:15:55:918",
        "type": "CHANNEL_START"
      }
    ]
  }
}
```

[JSON](#) [XML](#)

Requests

```
GET /ecs/channels/<channel_id>/signals.xml?<option>=<option_value>"
Authorization: [Session ID]
```

Parameters

Name	Type	Description
channel_id	<i>string</i>	Label, or name, of the channel. For example, "Channel_1."
option	<ul style="list-style-type: none"> time timespan index 	<ul style="list-style-type: none"> time: Request all signals acquired after a specified time. timespan: Request all signals acquired in the last <i>nn</i> seconds. index: Returns all signals received after the provided index. Use -1 for the first request.
option_value	<ul style="list-style-type: none"> HH:MM:SS:mmm SS <i>int</i> 	<ul style="list-style-type: none"> HH:MM:SS:mmm: hours (in military time), minutes, seconds, and milliseconds respectively for the time option. SS: seconds for the timespan option. Integer value for the index option.
Authorization	[<i>Session ID</i>]	The sessionid value returned in the response to a Login request.

Response

```
<!DOCTYPE channel>
<channel label="TestChannel2">
  <signal param1="48000:8:16" type="INPUT_AUDIO_FORMAT" param2="" index="3"
  id="Device_DeckLink_Baseband_2" name="Input Source" comp="Device_DeckLink_Baseband"
  time="02:15:57:412" date="2016:07:12"/>
  <signal param1="1920:1080:29.97" type="INPUT_VIDEO_FORMAT" param2="" index="2"
  id="Device_DeckLink_Baseband_2" name="Input Source" comp="Device_DeckLink_Baseband"
  time="02:15:57:412" date="2016:07:12"/>
  <signal param1="" type="INPUT_READYTODELIVER" param2="" index="1"
  id="Device_DeckLink_Baseband_2" name="Input Source" comp="Device_DeckLink_Baseband"
  time="02:15:57:015" date="2016:07:12"/>
  <signal param1="TestChannel2" type="CHANNEL_START" param2="" index="0" id="" name="Channel"
  comp="CONTROLLER" time="02:15:55:918" date="2016:07:12"/>
</channel>
```

Get Video Preview

Get a preview of the stream currently being encoded/transcoded on a specified channel. The image is returned as a JPEG image. The image quality is determined by default settings:

- Image size: 356x200 pixels
- JPEG quality: 80 (max 100)
- Update period: 1 frame per second

Active for Version: 1.0+

Authorizations: Administrator, Operator, Observer

Note

The images size and update period can be changed within a channel configuration. See [Preview Object](#) for details.

[JSON](#) [XML](#)**Requests**

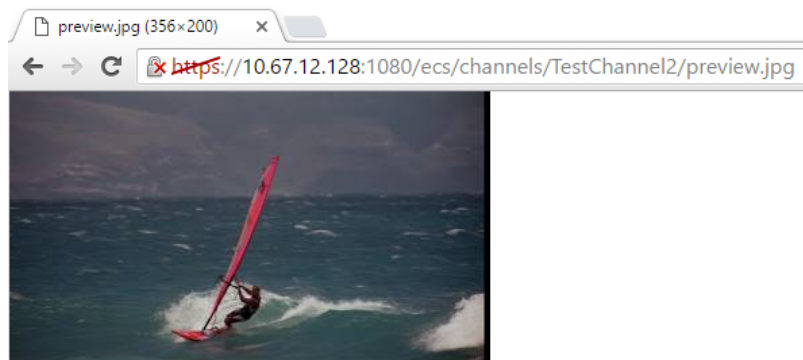
```
GET /ecs/channels/<channel_id>/preview.jpg
Authorization: [Session ID]
```

Parameters

Name	Type	Description
channel_id	<i>string</i>	Label, or name, of the channel. For example, "Channel_1."
Authorization	<i>[Session ID]</i>	The sessionid value returned in the response to a Login request.

Response

The response includes the preview encoded as an application/JPEG image. For example:

[JSON](#) [XML](#)**Requests**

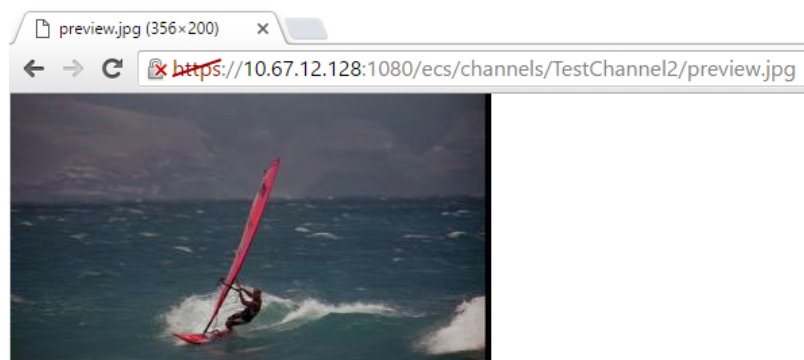
```
GET /ecs/channels/<channel_id>/preview.jpg
Authorization: [Session ID]
```


Parameters

Name	Type	Description
channel_id	<i>string</i>	Label, or name, of the channel. For example, "Channel_1."
Authorization	<i>[Session ID]</i>	The sessionid value returned in the response to a Login request.

Response

The response includes the preview encoded as an application/JPEG image. For example:



Inject Metadata

Create your own metadata, and inject it into any RTMP stream.

Active for Version: 1.0+

Authorizations: Administrator, Operator

[JSON](#) [XML](#)

Requests

```
PUT /ecs/channels/<channel_id>/outputs/<output_id>/streams/<stream_id>.json
Authorization: [Session ID]

{
  "data": [
    {
      label: "MetadataName1",
      type: "int",
      value: "12345",
    },
    {
      label: "MetadataName2",
      type: "double",
      value: "123.45",
    },
    {
      label: "MetadataName3",
      type: "bool",
      value: "true",
    },
    {
      label: "MetadataName4",
      type: "string",
      value: "This is how we inject metadata.",
    }
  ]
}
```

Currently supported types of metadata are "int", "double", "bool", "string".

Parameters

Name	Type	Description
channel_id	string	Label, or name, of the new channel. For example, "Channel_1."
output_id	string	The encode identifier. This label is specified in Step 4 Output Configuration .
stream_id	string	The recording identifier. This is the id of the stream component. It is present in the channel state as "label" under the stream's component section.
Authorization	[Session ID]	The sessionid value returned in the response to a Login request.
data	-	
↳	label	string A label or name for the metadata.
	type	<ul style="list-style-type: none"> int double bool string <ul style="list-style-type: none"> int - a numeric value that does not have a fractional component. double - double-precision floating-point numbers. bool - a data type with only two possible values: "true" or "false". string - a series of characters typically used as a literal constant or as a variable.
	value	See description. Sets the value for the metadata; the format must adhere to the "type" specified above.

Response

```
{
  "messages": {
    "count": 1,
    "message": [
      {
        "text": "Request processed successfully.",
        "type": "info"
      }
    ]
  }
}
```

JSON [XML](#)

Requests

```
PUT /ecs/channels/<channel_id>/outputs/<output_id>/streams/<stream_id>.xml
Authorization: [Session ID]

<?xml version="1.0"?>
<Metadata>
  <data label="MetadataName1" type="int" value="12345" />
  <data label="MetadataName2" type="double" value="123.45" />
  <data label="MetadataName3" type="bool" value="true" />
  <data label="MetadataName4" type="string" value="This is how we inject metadata." />
</Metadata>
```

Parameters

Name	Type	Description
channel_id	string	Label, or name, of the new channel. For example, "Channel_1."
output_id	string	The encode identifier. This label is specified in Step 4 Output Configuration .
stream_id	string	The recording identifier. This is the id of the stream component. It is present in the channel state as "label" under the stream's component section.
Authorization	[Session ID]	The sessionid value returned in the response to a Login request.
data	-	
↳	label	A label or name for the metadata.
	type	<ul style="list-style-type: none"> int - a numeric value that does not have a fractional component. double - double-precision floating-point numbers. bool - a data type with only two possible values: "true" or "false". string - a series of characters typically used as a literal constant or as a variable.

Name	Type	Description
value	See description.	Sets the value for the metadata; the format must adhere to the "type" specified above.

Response

```
<?xml version="1.0"?>
<messages count="1">
  <message type="info">Request processed successfully.</message>
</messages>
```

Inject AdCue Messages

Note

Automatic SCTE-35 messages are generated when the "scte35" doesn't contain a message.

Active for Version: 5.0+

Authorizations: Administrator, Operator

[JSON](#) [XML](#)

Requests

```
PUT /ecs/channels/<channel_id>/inputs/<input_id>/adcue.json
Authorization: [Session ID]

{
  "adcue": {
    "type": "scte35",
    "cue": "<?xml version=\"1.0\" encoding=\"UTF-8\"?>

    <SpliceInfoSection ... </SpliceInfoSection>",
    "duration": "5.0",
    "offset" : "0.0",
    "repeat" : "no"
  }
}
```

Parameters

Name	Type	Description
channel_id	string	Label, or name, of the new channel. For example, "Channel_1."
input_id	string	Label, or name, of the input. For example, "Device_DeckLink_Baseband_2."

Name	Type	Description
Authorization	[Session ID]	The sessionid value returned in the response to a Login request.
adcue	—	
type	<ul style="list-style-type: none"> SpliceOut scte35 	<ul style="list-style-type: none"> "SpliceOut" for a simple mode AdCue message. "scte35" for a complete SCTE-35 message.
cue	string	<p>Optional. When type is set to "scte35", contains a SCTE-35 message in XML representation. Do not use this parameter when type is set to "SpliceOut".</p> <div style="border: 1px solid #ccc; padding: 5px; margin-top: 10px;"> <p>Note</p> <p>When type is set to "scte35" but no cue message is set a <i>SpliceInsert</i> message is created with <code>outOfNetworkIndicator</code> and <code>autoReturn</code> is set to true. But only when a duration value is greater than 0.0. This can be used to create simple SCTE-35 workflows; for example, use together with RTMP onCuePoint ad insertion messages.</p> </div>
duration	string	The duration of the ad insertion in seconds as a string. This value is required when type is set to "SpliceOut". When type is set to "scte35", this value can be set to 0.0 when the duration is not known or not applicable.
offset	string	An offset value to the current stream time in seconds as a string, when this AdCue message should be executed.
repeat	string	Set this parameter to "true", when this message is a repetition of the previous message. Else, set this parameter to "false".

Response

```

{
  "messages": {
    "count": 1,
    "message": [
      {
        "text": "Request processed successfully.",
        "type": "info"
      }
    ]
  }
}

```

JSON [XML](#)

XML

Requests

```
PUT /ecs/channels/<channel_id>/inputs/<input_id>/adcue.xml
Authorization: [Session ID]

<?xml version="1.0"?>
<AdCue type="scte-35" duration="30.0" offset="60.0" repeat="false">
  <cue>
    <!-- SCTE-35 message string in XML representation. The content must be XML encoded! -->
  </cue>
</AdCue>
```

Parameters

Name	Type	Description
channel_id	string	Label, or name, of the new channel. For example, "Channel_1."
input_id	string	Label, or name, of the input. For example, "Device_DeckLink_Baseband_2."
Authorization	[Session ID]	The sessionid value returned in the response to a Login request.
adcue	—	
type	<ul style="list-style-type: none"> SpliceOut scte35 	<ul style="list-style-type: none"> "SpliceOut" for a simple mode AdCue message. "scte35" for a complete SCTE-35 message
cue	string	<p><i>Optional.</i> When type is set to scte35, contains a SCTE-35 message in XML representation. Do not use this parameter when type is set to SpliceOut.</p> <div style="border: 1px solid #f0e68c; padding: 10px; margin-top: 10px;"> <p>Note</p> <p>When type is set to "scte35" but no cue message is set a <i>SpliceInsert</i> message is created with <i>outOfNetworkIndicator</i> and <i>autoReturn</i> set to true. But only when a duration value is greater than 0.0. This can be used to create simple SCTE-35 workflows; for example, use together with RTMP onCuePoint ad insertion messages.</p> </div>
duration	string	The duration of the ad insertion in seconds as a string. This value is required when type is set to SpliceOut. When type is set to scte35, this value can be set to 0.0 when the duration is not known or not applicable.

Name	Type	Description
offset	string	An offset value to the current stream time in seconds as a string, when this AdCue message should be executed.
repeat	string	Set this parameter to "true", when this message is a repetition of the previous message. Else, set this parameter to "false".

Response

```
<?xml version="1.0"?>
<messages count="1">
  <message type="info">Request processed successfully.</message>
</messages>
```

Managing Channel Configurations

A channel configuration defines all the parameters that a channel needs to ingest one input stream, process it in various ways, and output the processed stream to a network or recording. The API provides a means to view and load channel configurations into the encoder.

More details about the channel configuration content is provided in [Channel Configuration File Syntax Reference](#).

Get Channel Configurations

Requests a list of channel configurations currently available in the web interface, including running and idle channels. This command differs from the [Get Channel List](#) command, which returns channels created via API and only running channels created via the web interface.

Active for Version: 5.4+

Authorizations: Administrator, Operator, Observer

Note

An XML response isn't supported for this request.

[JSON](#) [XML](#)

Requests

```
GET /ecs/channelconfigs.json
Authorization: [Session ID]
```

Parameters

Name	Type	Description
Authorization	[Session ID]	The sessionid value returned in the response to a Login request.

Response

Click here to expand JSON response...

```
[
  {
    "preset": {
      "version": "5.0",
      "type": "multistream",
      "label": "RTMP",
      "description": "",
      "kulabytepreset": {
        "input": {
          "format": {
            "videoin_enable": "1",
            "videoin_width": "1920",
            "videoin_height": "1080",
            "videoin_framerate": "29.97",
            "audioin_enable": "1",
            "audioin_samplerate": "48000",
            "audioin_bitspersample": "16",
            "audioin_channels": "2"
          },
          "video_processing": {},
          "audio_processing": {},
          "source": [
            {
              "parameters": {
                "serverurl": "rtmp://1.2.3.4",
                "streamname": "abc",
                "buffersize": "5000",
                "capture_closed_captions": "true"
              },
              "id": "RTMP_Source_2",
              "type": "RTMP_Source",
              "audiolanguages": [
                {
                  "channels": [
                    0,
                    1
                  ],
                  "language": {
                    "id": "eng",
                    "name": "English",
                    "comment": ""
                  }
                }
              ],
              "close_captioning": [
                {
                  "id": "",
                  "name": "",
                  "comment": ""
                },
                {
                  "id": "",
                  "name": "",
                  "comment": ""
                }
              ]
            }
          ]
        }
      }
    }
  }
]
```



```

        },
        {
            "id": "",
            "name": "",
            "comment": ""
        },
        {
            "id": "",
            "name": "",
            "comment": ""
        }
    ]
},
{
    "type": "Default_Source",
    "id": "Default_Source_1",
    "parameters": {
        "mode": "1",
        "src": "/assets/images/Secondary_breaks.png"
    }
}
]
},
"preview": {
    "frame_width": "356",
    "frame_height": "200",
    "framerate": "1.0"
},
"video_encoder": [
    {
        "format": {
            "videoout_enable": "1",
            "videoout_width": "1920",
            "videoout_height": "1080",
            "videoout_framerate": "25-30"
        },
        "processing": {},
        "encoder": {
            "type": "AVC_ENC",
            "id": "AVC_ENC_4",
            "parameters": {
                "name": "AVC_ENC_4",
                "bitrate_avg": "3808",
                "gop_bcount": "2",
                "preset": "h264_high",
                "performance_preset": "7",
                "level": "auto",
                "entropy_coding_mode": "cabac",
                "gop_duration": "2000",
                "gop_fixed": "true"
            }
        }
    }
],
"audio_encoder": [
    {
        "format": {
            "audioout_enable": "1",
            "audioout_samplerate": "48000",
            "audioout_bitspersample": "16",
            "audioout_channels": "2"
        },
        "processing": {
            "audio_channel_config": "0:0|1:1"
        }
    }
],

```

```
        "encoder": {
          "type": "AAC_ENC",
          "id": "AAC_ENC_eng_5",
          "parameters": {
            "name": "AAC_ENC_eng_5",
            "langcode": "eng",
            "langcomment": "",
            "langname": "English",
            "bitrate_avg": "192"
          }
        }
      ],
      "output": [
        {
          "type": "RTMP_Sink",
          "id": "RTMP_Sink_6",
          "parameters": {
            "serverurl": "rtmp://2.3.4.5",
            "streamname": "abcd",
            "cdn": {
              "cdn": "none"
            },
            "username": "",
            "password": "",
            "name": "rtmp",
            "videoencoder": "AVC_ENC_4",
            "audioencoder": "AAC_ENC_eng_5",
            "source_serverurl": "rtmp://2.3.4.5",
            "source_playbackurl": "",
            "playbackurl": "",
            "source_streamname": "abcd",
            "langnames": "English",
            "langcodes": "eng",
            "langcomments": "",
            "agent_id": "KulaByte/5.0",
            "authmode": "auto",
            "ccincaptioninfo": "false",
            "ccincuepoint": "false",
            "ccintextdata": "false",
            "ccinmetadata": "false",
            "scte35_adcue": "false",
            "scte35_cuepoint": "false",
            "scte35_cuemessage": "false",
            "cc_langnames": "|||",
            "cc_langcodes": "|||",
            "cc_langcomments": "|||"
          }
        },
        {
          "type": "File",
          "id": "File_7",
          "parameters": {
            "videoencoder": "AVC_ENC_4",
            "audioencoder": "AAC_ENC_eng_5",
            "langnames": "English",
            "langcodes": "eng",
            "langcomments": "",
            "filename": "RTMP-%year%-%month%-%day%-%hour%-%minute%-%second%-
%index%.mp4",
            "splitonduration": "false",
            "fileduration": "14400",
            "splitonsize": "true",
            "filesize": "4194304",
            "cc_langnames": "|||",
```

```
        "cc_langcodes": "|||",
        "cc_langcomments": "|||"
      }
    ]
  },
  "_id": "5c6c3e74c08586c8070219d7"
},
"_id": "5c6c3e74c08586c8070219d7",
"state": "idle",
"actions": {
  "recording": "idle"
},
"name": "RTMP",
"description": "",
"metrics": {
  "memory": "0",
  "v_memory": "0",
  "cpu": "0",
  "network": {
    "incoming": "0",
    "outgoing": "0"
  }
}
}
]
```

[JSON](#) [XML](#)

XML Request Not Supported

Review a Channel's Configuration

Allows you to review the channel configuration of a specified channel. See [Channel Configuration File Syntax Reference](#) for details regarding the file format.

Active for Version: 1.0+

Authorizations: Administrator, Operator, Observer

Note

An XML response isn't supported for this request.

[JSON](#) [XML](#)

Requests

```
GET /ecs/channels/<channel_id>/preset.json
Authorization: [Session ID]
```

Parameters

Name	Type	Description
channel_id	—	Label, or name, of the channel. For example, "Channel_1."
Authorization	[Session ID]	The sessionid value returned in the response to a Login request.

Response

Click here to expand JSON response...

```
{
  "_id": "5746fd09b87e350328a88b07",
  "description": "test%20description",
  "kulabytepreset": {
    "audio_encoder": [
      {
        "encoder": {
          "id": "AAC_ENC_eng_7",
          "parameters": {
            "bitrate_avg": "128",
            "langcode": "eng",
            "langcomment": "",
            "langname": "English",
            "name": "AAC_ENC_eng_7"
          },
          "type": "AAC_ENC"
        },
        "format": {
          "audioout_bitspersample": "16",
          "audioout_channels": "2",
          "audioout_enable": 1,
          "audioout_samplerate": "48000"
        },
        "processing": {
          "audio_channel_config": "0:0|1:1"
        }
      }
    ],
    "input": {
      "audio_processing": {},
      "format": {
        "audioin_bitspersample": "16",
        "audioin_channels": "2",
        "audioin_enable": "1",
        "audioin_samplerate": "48000",
        "videoin_enable": "1",
        "videoin_framerate": "29.97",
        "videoin_height": "1080",
        "videoin_width": "1920"
      },
      "source": [
        {
          "audiolanguages": [
            {
              "channels": [
                0,
                1
              ],
              "language": {
                "comment": "",
                "id": "eng",

```

```
        "name": "English"
      }
    ]
  },
  "id": "RTMP_Source_2",
  "parameters": {
    "bufferSize": "5000",
    "serverurl": "rtmp://a",
    "streamname": "a"
  },
  "type": "RTMP_Source"
},
{
  "audiolanguages": [],
  "id": "Default_Source_3",
  "parameters": {},
  "type": "Default_Source"
}
],
"video_processing": {}
},
"output": [
  {
    "id": "HLS_Sink_4",
    "parameters": {
      "audioencoder": "AAC_ENC_eng_7",
      "cdn": {
        "cdn": "none",
        "customer_hostname": null,
        "event_name": null,
        "file_name": null,
        "stream_id": null
      },
      "directory_rollover": false,
      "langcodes": "eng",
      "langcomments": "",
      "langnames": "English",
      "name": "a",
      "password": "",
      "playbackurl": "",
      "playlistname": "playlist.m3u8",
      "segment_count": "10",
      "segment_duration": "10000",
      "serverurl": "http://a",
      "username": "",
      "videoencoder": "AVC_ENC_4|AVC_ENC_5|AVC_ENC_6"
    },
    "type": "HLS_Sink"
  },
  {
    "id": "File_5",
    "parameters": {
      "audioencoder": "AAC_ENC_eng_7",
      "fileduration": "14400",
      "filename": "Test-AVC_ENC_4-AAC_ENC_eng_7.mp4",
      "filesize": "2097152.00",
      "langcodes": "eng",
      "langcomments": "",
      "langnames": "English",
      "splitonduration": "false",
      "splitonsize": "true",
      "videoencoder": "AVC_ENC_4"
    },
    "type": "File"
  }
]
```

```
],
"preview": {
  "frame_height": 200,
  "frame_width": 356,
  "framerate": 1
},
"video_encoder": [
  {
    "encoder": {
      "id": "AVC_ENC_4",
      "parameters": {
        "bitrate_avg": "772",
        "entropy_coding_mode": "cabac",
        "gop_bcount": "2",
        "gop_duration": "3000",
        "gop_fixed": "true",
        "level": "auto",
        "name": "AVC_ENC_4",
        "performance_preset": "7",
        "preset": "h264_main"
      },
      "type": "AVC_ENC"
    },
    "format": {
      "videoout_enable": 1,
      "videoout_framerate": "25-30",
      "videoout_height": "360",
      "videoout_width": "640"
    },
    "processing": {}
  },
  {
    "encoder": {
      "id": "AVC_ENC_5",
      "parameters": {
        "bitrate_avg": "372",
        "entropy_coding_mode": "cabac",
        "gop_bcount": "2",
        "gop_duration": "3000",
        "gop_fixed": "true",
        "level": "auto",
        "name": "AVC_ENC_5",
        "performance_preset": "7",
        "preset": "h264_main"
      },
      "type": "AVC_ENC"
    },
    "format": {
      "videoout_enable": 1,
      "videoout_framerate": "25-30",
      "videoout_height": "270",
      "videoout_width": "480"
    },
    "processing": {}
  },
  {
    "encoder": {
      "id": "AVC_ENC_6",
      "parameters": {
        "bitrate_avg": "172",
        "entropy_coding_mode": "cabac",
        "gop_bcount": "2",
        "gop_duration": "3000",
        "gop_fixed": "true",
        "level": "auto",
```

```

        "name": "AVC_ENC_6",
        "performance_preset": "7",
        "preset": "h264_main"
    },
    "type": "AVC_ENC"
},
"format": {
    "videoout_enable": 1,
    "videoout_framerate": "25-30",
    "videoout_height": "180",
    "videoout_width": "320"
},
"processing": {}
}
]
},
"label": "Test",
"type": "multistream",
"version": "5.0"
}
    
```

JSON [XML](#)

XML Request Not Supported

Load a Channel's Configuration

Updates a channel with the supplied channel configuration. See [Channel Configuration File Syntax Reference](#) for details regarding the file format.

Active for Version: 1.0+

Authorizations: Administrator, Operator

[JSON](#) [XML](#)

Requests

```

PUT /ecs/channels/<channel_id>/preset.json
Authorization: [Session ID]

{
    ... (Channel configuration content) ...
}
    
```

Parameters

Name	Type/Value	Description
channel_id	—	Label, or name, of the channel. For example, "Channel_1."
Authorization	[Session ID]	The sessionid value returned in the response to a Login request.

Name	Type/Value	Description
...Channel configuration content...	—	See Channel Configuration File Syntax Reference .

Response

```
{
  "messages": {
    "count": 1,
    "message": [
      {
        "text": "Request processed successfully.",
        "type": "info"
      }
    ]
  }
}
```

JSON [XML](#)

Requests

```
PUT /ecs/channels/<channel_id>/preset.xml
Authorization: [Session ID]

... (Channel configuration content) ...
```

Parameters

Name	Type/Value	Description
channel_id	—	Label, or name, of the channel. For example, "Channel_1."
Authorization	[Session ID]	The sessionid value returned in the response to a Login request.
...Channel configuration content...	—	See Channel Configuration File Syntax Reference .

Response

```
<?xml version="1.0"?>
<messages count="1">
  <message type="info">Request processed successfully.</message>
</messages>
```

Get Configuration Label

Retrieves the label of a channel's configuration.

Active for Version: 1.0+

Authorizations: Administrator, Operator, Observer

[JSON](#) [XML](#)

Requests

```
GET /ecs/channels/<channel_id>/preset/label.json
Authorization: [Session ID]
```

Parameters

Name	Type	Description
channel_id	<i>string</i>	Label, or name, of the channel. For example, "Channel_1."
Authorization	<i>[Session ID]</i>	The sessionid value returned in the response to a Login request.

Response

```
{
  "preset": "This is the configuration Label"
}
```

[JSON](#) [XML](#)

Requests

```
GET /ecs/channels/<channel_id>/preset/label.xml
Authorization: [Session ID]
```

Parameters

Name	Type	Description
channel_id	<i>string</i>	Label, or name, of the channel. For example, "Channel_1."
Authorization	<i>[Session ID]</i>	The sessionid value returned in the response to a Login request.

Response

```
<preset label="This is the configuration Label"/>
```

Error Response Messages

The following lists the possible error messages that may be returned in response to a request.

Create Channel

If the response to a [Create Channel](#) request provides a 500 HTTP error code, the response also includes one of the following API error messages:

Code	Description
KULA_ERROR_INIT_FAILED	The initialization of the channel resource failed. This shouldn't occur in real life, maybe if the encoder process reaches his memory limit-
KULA_ERROR_ALREADY_EXISTS	A channel with the same name already exists.
KULA_ERROR_LICENSE_ERROR_TIME	The license time expired.
KULA_ERROR_LICENSE_ERROR_OS	The used operating system is not licensed.
KULA_ERROR_LICENSE_NOT_ALLOWED	The operation is not allowed by the license. The channel couldn't register at the license server but the error is not a reached channel limit. For example if the license server is down.
KULA_ERROR_INVALID_PARAMETER	Some parameter are not incorrect. This shouldn't occur in real life.
KULA_ERROR_LICENSE_NO_FREE_CHANNEL	There is no free channel available. The license limit was passed.
KULA_ERROR_LICENSE_INFO_NOT_AVAILABLE	The license server is not available or didn't provide a license.
KULA_ERROR_LICENSE_ACCESS_DENIED	The user is not allowed to lock a channel on the license server.

License Violations during Channel Initialization

While loading a channel configuration, license violations are reported as warnings because they do not prevent a successful load. The response contains messages with the occurred violations.

Message Format:

"LicenseViolation: TYPE: <TYPE>, COMP: <Component ID>"

where <TYPE> describes what exactly occurred and the <Component ID> contains the id of the component that was effected.

Type	Description
LICENSE_PASSTHROUGH_NOT_ALLOWED	A pass-through output was not created

Signals Reference

Note

In this context, an "event" is something that happens in a program, e.g., a key is clicked, a buffer has reached its limit, and so forth).

To request a list of signals that have fired by an event in a specified channel, see [Get Channel Signal List](#). These signals are also reported:

- On the channel's Notification Log screen. See [Channel Details: Notification Log Screen](#) in the User's Guide.
- In the application log files. See [Accessing Reports \(Logs\)](#) in the User's Guide.

ECS Signals

Signal name	Param1	Param2	When does the signal occur	Default handling	Version
ENCODER_REMOVES_CHANNEL	Channel id	Channel GUID	The ECS removes a channel because it has passed this restart limit.	—	4.0+

Channel Signals

Signal name	Param1	Param2	Module	When does the signal occur	Default Handling	Version
ERROR	Error string like KULA_ERROR_OVERFLOWMEMORY	optional, error description		Could be sent on an error that is not handled by other signals.		4.0+
RTMP_INPUT_STARTBUFFERING	—	—		Input receives media data and starts buffering		4.0+
RTMP_INPUT_READYTODELIVER	—	—	RTMP Input components	The input component throws this event before delivering the first audio or video sample.	The CKulaController class switch from the default Input component to the Input Component.	4.0+
RTMP_INPUT_CONNECTING	—	—		RTMP input is connecting		4.0+
RTMP_INPUT_REQUIRESAUTHENTICATION	—	—		RTMP input connection requires authentication.		4.0+

Signal name	Param1	Param2	Module	When does the signal occur	Default Handling	Version
RTMP_INPUT_CONNECTED	—	—		RTMP source component connected to the server		4.0+
RTMP_INPUT_DISCONNECTED	—	—		RTMP source component disconnected from the server		4.0+
CAPTURE_SIGNAL_LOST	—	—		Capture card or input stream detects no input signal.		4.0+
LOST_CAPTURE_SIGNAL_BACK	—	—		Capture card or input stream detects an input signal after a signal lost.		4.0+
LOW_INPUT_FRAMERATE	Expected video frame rate in fps.	Current video frame rate in fps.		The input video frame rate is lower than expected.		4.0+
LOW_INPUT_SAMPLERATE	Expected audio sample rate in samples per second.	Current audio sample rate in samples per second.		The input audio sample rate is lower than expected.		4.0+
VIDEO_BUFFER_OVERFLOW	Limit count of buffer items.	—		A video buffer in a component reaches its limit.		4.0+
AUDIO_BUFFER_OVERFLOW	Limit count of buffer items.	—		An audio buffer in a component reaches its limit.		4.0+
VIDEO_BUFFER_WARNING	Limit count of buffer items.	Current count of item buffers.		A video buffer increases over 50% of its limit.		4.0+
AUDIO_BUFFER_WARNING	Limit count of buffer items.	Current count of item buffers.		An audio buffer increases over 50% of its limit.		4.0+
SEGMENT_BUFFER_OVERFLOW	Limit count of buffer items.	—		A segment buffer in a component reaches its limit.	This signal is sent by HLS output.	4.0+
SEGMENT_BUFFER_WARNING	Limit count of buffer items.	Current count of item buffers.		A segment buffer increases over 50% of its limit.	This signal is sent by HLS output.	4.0+
RTMP_OUTPUT_CONNECTING	—	—		RTMP output is connecting.		4.0+
RTMP_OUTPUT_CONNECTED	—	—		RTMP output connected.	When enabled (encoder cmd. line option -R), archive files are split in response to this event.	4.0+

Signal name	Param1	Param2	Module	When does the signal occur	Default Handling	Version
RTMP_OUTPUT_DISCONNECTED	—	—		RTMP output disconnected.		4.0+
RTMP_OUTPUT_REQUIRESAUTHENTICATION	—	—		RTMP output connection requires authentication.		4.0+
RTMP_STREAM_STARTED	—	—		RTMP stream started.		4.0+
RTMP_STREAM_PUBLISHED	—	—		RTMP stream was published.		4.0+
RTMP_STREAM_STOPPED	—	—		RTMP Stream stopped.		4.0+
RTMP_STREAM_UNPUBLISHED	—	—		RTMP stream was unpublished.		4.0+
RTMP_BAD_NAME	Stream name.	—		Signals a bad name for the RTMP stream. Usually this means that a stream with the same name is already published on the server.		4.3+
RTMP_STREAM_NOT_FOUND	—	—	RTMP Input	A RTMP stream is not found on a RTMP server.		4.3+
RTMP_INSUFFICIENTBW	—	—	RTMP Input	RTMP input data rate is too low.		4.3+
RTMP_NO_AUDIO_PASS_THROUGH	Audio format name, e.g. "MP3"	—	RTMP Input	Pass-through is not possible for input audio format.		4.3+
OUTPUT_BUFFER_WARNING	Output buffer level in percent.	—	RTMP Output	The level of the output buffer becomes critical.		4.3+
OUTPUT_BUFFER_OVERFLOW	Output buffer level in percent.	—	RTMP Output	The level of the output buffer exceeds the limit and is critical.		4.3+
AV_UNBALANCED	Current video frame rate in frames per second.	Current audio sample rate in samples per second.	RTMP_INPUT	This signal is sent if both audio and video data is expected but the input of one of these streams is close to zero.		4.3+
NEW_OUTPUT_FILE	New file name.	Output folder.		A new output file was created.		4.0+

Signal name	Param1	Param2	Module	When does the signal occur	Default Handling	Version
NEW_OUTPUT_FILE_ERROR	Name of the failed new output file.	Error		An error occurred during creation of a new output file.		4.0+
OUTPUT_FILE_CLOSED	Name of the closed output file.	—		An output file was closed.		4.0+
LOW_INPUT_BITRATE	—	—		The input bit rate is too low.		4.0+
CHANNEL_RESTART	—	—		The channel restarts.		4.0+
CHANNEL_RESTART_FAILED	—	—		The channel can't restore the state of a previous session.		4.0+
NO_LICENSE_AVAILABLE	License type	—		No license information is available.		4.0+
LICENSE_NO_FREE_CHANNEL	—	—		No free channel is available.		4.0+
LICENSE_VIOLATION	—	—		License violation.		4.0+
TS_INPUT_STARTBUFFERING	—	—		Input receives media data and starts buffering.		4.0+
TS_INPUT_READYTODELIVER	—	—		The input component throws this event before delivering the first audio or video sample.	The CKulaController class switch from the default Input component to the Input Component.	4.0+
TS_INPUT_OPENING_PORT	—	—		Trying to open UDP/RTP port.		4.0+
TS_INPUT_OPENED_PORT	—	—		Opened UDP/RTP port.		4.0+
TS_INPUT_CLOSED_PORT	—	—		Closed UDP/RTP port.		4.0+
TS_NO_PASSTHROUGH	—	—		No Pass-Through possible for MPEG-2 Video Ingest.		4.0+
DECKLINK_CAPTURE_READYTODELIVER	—	—	Decklink input component	The input component throws this event before delivering the first audio or video sample.	The CKulaController class switch from the default Input component to the Input Component.	4.1+

Signal name	Param1	Param2	Module	When does the signal occur	Default Handling	Version
TS_OUTPUT_OPENING_PORT	URL	—	TS_Sink			4.4+
TS_OUTPUT_OPENED_PORT	URL	—	TS_Sink			4.4+
TS_OUTPUT_CLOSED_PORT	URL	—	TS_Sink			4.4+
TS_OUTPUT_SENDING	URL	—	TS_Sink			4.4+
VIDEO_BITRATE_CHANGED	Actual bitrate	Configured bitrate	Video encoder	Video encoder sends this after bitrate correction to match available output bandwidth.		4.3+
SRT_DECRYPTION_FAILED	—	—	TS_Source	The input stream can't be decrypted.		4.4+
INPUT_VIDEO_FORMAT	Contains video width, height, and framerate separated by ":". Format: <width>:<height>:<framerate> E.g. : "1920:1080:29.97"	—	Decklink, RTMP, and TS inputs	Video format of the received stream. Sent by Decklink, RTMP, and TS input components before delivering the first video sample.		5.0+
INPUT_AUDIO_FORMAT	Contains audio sample rate, number of channels, and bits per sample separated by ":". Format: <samplerate>:<channels>:<bitsper sample> E.g. : "48000:2:16"	—	Decklink, RTMP, and TS inputs	Audio format of the received stream. Sent by Decklink, RTMP, and TS input components before delivering the first audio sample.		5.0+

Command Line Interface

Using the Command Line Interface

The KB software has a Command Line Interface (CLI) that can be used to schedule encodes. When using the Command Line Interface, you can invoke CLI switches to help customize encodes.

Available Command Line Switches

Note

Switches require a leading "-". When using multiple switches ensure they are used in the order listed in this section.

`KulabyteChannel [option(s)]`

Where:

`Option` - Can be one or more of the options listed in the following table.

Command Line Options	
Option	Description
-p <port>	Port number for the REST interface. Accepts every value greater than 0. Take care that the port is not already in use. Default is port 8080.
-l	Enable component based log file. Every component creates his own log file.
-L	Enable global log file. All components uses the same log file.
-f <log folder>	Output folder for log files.
-v <level>	Level for log entries.
	>= 80: all log messages (info, entry, exit, log, warning, error and fatal messages)
	< 80: no info, entry and exit messages
	< 60: no info, entry, exit and log messages
	< 40: no info, entry, exit, log and warning messages
	< 20: no info, entry, exit, log, warning and error messages
	< 0: no messages
-t	Enable timestamps logs

Command Line Options	
Option	Description
-s	Log file size in bytes.
-g <guid>	Encoder guid. Format: {xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx}
-h	Help output
-n <encoder name>	Encoder name.
-r <preset path>	Load a preset file and start the encoder.
-P <performance log path>	Enables the performance log file function.
-a <recordings folder>	Default output folder for recordings.
-R	Enables recording file splitting on RTMP output stream connect event (only on first RTMP output stream in each channel).

Channel Configuration File Syntax Reference

As discussed in [Review a Channel's Configuration](#) and in [Importing/Exporting a Channel](#) in the *User's Guide*, the channel configuration can be exported. This plain text file is in JSON format and its syntax is explained in this section. This file can then be loaded into other KB Encoders/Transcoders via the API as explained in [Load a Channel's Configuration](#).

Caution

Do not edit this file directly. Use the UI to apply any necessary channel configuration changes, and then export the configuration.

Important

Mandatory – All red-colored parameters must be part of a channel configuration file. Otherwise it is incomplete and the encoder/transcoder will not function!

Note

Version information is supplied in the elements and parameters tables to indicate the first version of the software an element or parameter was supported.

Topics Discussed

- [File Structure](#)
 - [File Structure Example](#)
 - [Component Objects](#)
 - [Processing](#)
- [Input Object](#)
 - [Input \(Common\)](#)
 - [Input > Format](#)
 - [Input > Audio_Processing](#)
 - [Input > Video_Processing](#)
 - [Input > Source \(Common\)](#)
 - [Input > Source \(Common for RTMP and MPEG_TS\)](#)
 - [Input > Source \(RTMP\)](#)
 - [Input > Source \(MPEG-TS\)](#)
 - [Input > Source \(DeckLink Baseband\)](#)
 - [Input > Source \(AJA Baseband\)](#)
 - [Input > Source \(Default_Source\)](#)
- [Preview Object](#)
- [Video_encoder Object](#)
 - [Video_encoder > Format](#)
 - [Video_encoder > Processing](#)
 - [Video_encoder > Encoder](#)

- [Audio_encoder Object](#)
 - [Audio_encoder > Format](#)
 - [Audio_encoder > Processing](#)
 - [Audio_encoder > Encoder](#)
- [Output Object](#)
 - [Output > Parameters \(Common\)](#)
 - [Output > Parameters \(RTMP\)](#)
 - [Output > Parameters \(HLS\)](#)
 - [Output > Parameters \(File\)](#)
 - [Output > Parameters \(MPEG TS\)](#)
 - [Output > Parameters \(MPEG DASH\)](#)

File Structure

There are five elements to a channel configuration file: input, preview, video_encoder, audio_encoder, and output. While only one input element is allowed, the channel configuration file can contain as many encoder and output elements as required (and/or dictated by the number the machine can handle).

Element	Description
input	The element that defines the input of the encoding or transcoding process. At least one of the sub-elements is required. Normally, input types should not be used together. Only the Default_Source element can be included with other input types.
preview	This element is only shown for completeness. Usually it is not necessary to add this element, unless there are some special requirements regarding the preview image.
video_encoder	This element defines the video encoding.
audio_encoder	This element defines the audio encoding.
output	The element that defines output destinations of the streams.

File Structure Example

The following is a sample file structure. The <Component Objects> are defined in [Component Objects](#).

```

Channel Configuration File Structure
1  {
2    "version": "5.0",
3    "type": "multistream",
4    "label": "...",
5    "description": "...",
6    "kmlabytetest": {
7      "input": {
8        "format": {
9          ...
10       },
11       "video_processing": {
12         ...
13       },
14       "audio_processing": {
15         ...
16       },
17       "source": [
18         <Component Objects>
19       ]

```

```

20     },
21     "preview": {
22         ...
23     },
24     "video_encoder": [
25         {
26             "format": {
27                 ...
28             },
29             "processing": {
30                 ...
31             },
32             "encoder": {
33                 <Component Object>
34             }
35         }
36     ],
37 ],
38 "audio_encoder": [
39     {
40         "format": {
41             ...
42         },
43         "processing": {
44             ...
45         },
46         "encoder": {
47             <Component Object>
48         }
49     }
50 ],
51     ...
52 ],
53 "output": [
54     <Component Objects>
55 ]
56 }
57 }

```

Component Objects

A main configuration object is a component object. It contains all information for an engine component. Component objects are used to define the input source, video encoders, audio encoders, and outputs, and are structured as follows:

```

1  {
2  "type" : "...",
3  "id" : "...",
4  "parameters" : {
5  "<parameter_name_1>" : "<parameter value_3>",
6  "<parameter_name_2>" : "<parameter value_2>",
7  ...
8  "<parameter_name_n>" : "<parameter value_n>",
9  }
10 }

```

The following table describes the elements used by all component objects. See the individual component object sections for possible values of the type field and descriptions of the corresponding parameters object.

Element	Description
type	Component type id, e.g. RTMP_Source, H264_ENC
id	A unique string that identifies the component.
parameters	Object that contains the parameters of the component.

Processing

The processing elements can be found within the input element and also within the audio_encoder and video_encoder elements. The processing sub-elements contain parameters for some processing functions.

Input processing is applied if the audio or video settings in the input > format section differ from the input signal received from a capture device, an RTMP stream, or a TS stream. For example, if the capture card is configured to capture 1080i59.94 and the input format section states 1280x720@29.97, then the video is resized and the frame rate is reduced (every second frame is dropped). If video processing is configured to deinterlace, the video is first deinterlaced from 1080i59.94 to 1080p29.97 and then resized to 720p29.97.

Note

Audio Sample Rate Conversion is not yet supported.

Input Object

The Input module provides video and audio data and maybe some additional data, like closed captioning or metadata. Only one input module can be used as input.

Element	Description	Version
format	Includes details about the input source format (video framerate, video resolution, audio sample rate, etc.). See Input > Format for descriptions of the format elements.	5.0+
video_processing	Object for video processing. See Input > Video_Processing for descriptions of the video_processing elements.	5.0+
audio_processing	Object for audio processing. See Input > Audio_Processing for descriptions of the audio_processing elements.	5.0+
source	The source array in the input object can contain one or two input component objects. If the array contains two objects then one of them must be of the type Default_Source. The second object should define an input object, like RTMP_Source or Device_DeckLink_Baseband. See the following sections for descriptions of the source elements.	5.0+

Input (Common)

Element	Description	Range of Values	Default	Version
aspect_ratiox	Explicitly set an aspect ratio for the captured signal. By default the Sample Aspect Ratio is assumed to be 1:1 (square pixels) and therefore the frame aspect ratio is, e.g., 16:9 for all HD content and 4:3 for PAL (720x576). However, PAL may be sent anamorphic in 16:9, which would require AFD (Active Format Description) in the SDI VANC, which Decklink Capture Cards don't read and therefore is lost. These Frame Aspect Ratio parameters (aspect_ratiox, aspect_ratioy) can overwrite the default by setting it to, e.g., 16:9 or 4:3. The appropriate Sample Aspect Ratio will be calculated by the component. The default value is "0", meaning that the parameter is ignored.	[0-]	0	5.0+
aspect_ratioy	See above.	[0-]	0	5.0+
audio_offset	Offset for audio timestamps in ms (helps with sync shift).	[-1000-1000]	0	5.3+

Input > Format

Element	Description	Range of Values	Default	Version
videoin_enable	Enable/Disable input video.	[0-1]	1	4.0+
videoin_width	Input video width. If the source is Device_DeckLink_Baseband, this value can be used to resize the video image by the Decklink hardware. In this case, the width value must not be the width of the captured input signal. This feature can be used to reduce CPU load if the highest output resolution is smaller than the input signal resolution.	—	—	4.0+
videoin_height	Input video height. If the source is Device_DeckLink_Baseband, this value can be used to resize the video image by the Decklink hardware. In this case, the height value must not be the height of the captured input signal. This feature can be used to reduce CPU load if the highest output resolution is smaller than the input signal resolution.	—	—	4.0+
videoin_framerate	Input video frame rate.	—	—	4.0+
audioin_enable	Enable/Disable input audio.	[0-1]	1	4.0+
audioin_samplerate	Input audio sample rate.	—	—	4.0+
audioin_bitspersample	Input audio bit-rate.	—	—	4.0+
audioin_channels	Input audio channels number.	—	—	4.0+

Input > Audio_Processing

Element	Description	Range of Values	Default	Version
audio_channel_config	The audio output channels can be selected from any input channel. This option can be used to select a dedicated input channel from a captured multichannel audio signal, e.g. to provide streams in different languages. Therefore, the processing section of the audio stream contains one or more <audio_channel_config value="[input channel number]:[output channel number]"/> elements. All entries with channel numbers above the max channels defined in the input or output format definition are ignored. The channel number range is from 0 to max_channels-1. Audio channel configuration specified in the Input section serves as default configuration for outputs where channel configuration is not specified.	[input channel number]:[output channel number] example: "2:0" input channel 2 to output channel 0.	In the default configuration the first two channels are used for audio output.	4.1+
audio_gain	This value contains the gain value for the input audio signal in dB.	[-20.0-40.0]	0.0	4.1+

Input > Video_Processing

Element	Description	Range of Values	Default	Version
interpolation	Specifies the interpolation mode used for resizing.	[NN, Linear, Cubic, Cubic2p_BSpline, Cubic2p_Catmullrom, Cubic2p_b05c03, Super, Lanczos]	Linear	4.1+
deinterlace_mode	Specifies the algorithm used for deinterlacing interlaced content. "Progressive", "Disable" or "Disabled" all disable deinterlacing. So, in these cases the input should be progressive.	<ul style="list-style-type: none"> Disabled: [Progressive, Disable, Disabled] Enabled: [Duplicate, Blend, Median, Edge_Detect, Median_Threshold, CAVT] 	Progressive	4.1+
crop_left	Cropping of the source video in DeckLink Ingest Mode.	CropLeft + CropRight < Image Width Value must be a factor of 4 and must be ≥ 0.	0	4.3+
crop_right	Cropping of the source video in DeckLink Ingest Mode.	CropLeft + CropRight < Image Width Value must be a factor of 4 and must be ≥ 0.	0	4.3+
crop_top	Cropping of the source video in DeckLink Ingest Mode.	CropTop + CropBottom < Image Height Value must be a factor of 4 and must be ≥ 0.	0	4.3+

Element	Description	Range of Values	Default	Version
crop_bottom	Cropping of the source video in DeckLink Ingest Mode.	CropTop + CropBottom < Image Height Value must be a factor of 4 and must be ≥ 0.	0	4.3+
hquality_deinterlace	Enables or disables the high quality deinterlace mode.	[true, false]	true	5.0+

Input > Source (Common)

The input > source object is a component object. These input sources are supported for the type element in this object:

type Value	Description	Version
RTMP_Source	RTMP stream ingest.	4.0+
TS_Source	MPEG2 transport stream (TS) ingest.	4.0+
Device_DeckLink_Baseband	Baseband capture from Blackmagic Design DeckLink devices.	4.1+
Default_Source	Source module for static image source with silent audio. This module is a little special because it can be used together with one of the other ingests. The video/audio feed of this module is used if the other input module, the main module, cannot provide a stream.	4.0+
Device_AJA_Baseband	Baseband capture from AJA Corvid HEVC device	5.0+

The input > source object also includes an input audio language object:

Element	Description	Range of Values	Default	Version
audiolanguages	Container element defining input audio language channels.	—	—	4.0+

Input > Source (Common for RTMP and MPEG_TS)

Element	Description	Range of Values	Default	Version
UseInputBuffer	<div style="border: 1px solid yellow; padding: 5px; margin-bottom: 5px;"> <p>Note Should be "false" for SRT ingest.</p> </div>	[true, false]	true	4.0+
BufferSize	Buffer size for media data in milliseconds. Play out starts after receiving min <bufferize> amount of media data.	[1000-]	5000	4.0+
AudioLevelUpdateInterval	Time interval for audio level checks in ms.	[100-5000]	250	4.0+
AudioLevelUsePeak	Use the peak (or the last) audio level value for the update interval.	[true, false]	true	4.0+

Element	Description	Range of Values	Default	Version
enable_sw_decode	Forces decoding by the CPU. For supported platforms, the default decoder is a Intel QuickSync-based implementation.	[true, false]	false	5.3+

Input > Source (RTMP)

Element	Description	Range of Values	Default	Version
ServerURL	Video Stream URL	—	—	4.0+
StreamName	Stream name	—	—	4.0+
Agent_Id	Name of the RTMP agent	—	KulaByte/<Version Major>.<Version Minor>	4.0+
AutoBufferSizeAdjustment	Automatic Input Buffer Size Adjustment	[off, low, medium, high]	off	4.1+

Input > Source (MPEG-TS)

Element	Description	Range of Values	Default	Version
url	Video Stream URL, for unicast & multicast and different protocol UDP, RTP, and SRT.	<ul style="list-style-type: none"> "udp://..." "rtp://..." "srt://..." 	—	<ul style="list-style-type: none"> 4.0+ 4.4+ (SRT support)
port	Network protocol port number.	—	—	4.0+
audio_config	Contains the audio configuration for all audio streams that should be used. If this value is empty, the first found audio stream is used. The audio channel configurations are separated with a ' ' and contains the type, a value, and the output stream number separated with a colon ':'. There are two types available "index" and "pid": "index" means the index of audio channel in the program table, and "pid" is the actual uses PID. The output stream number is an increasing unique number starting with 0. For example: "index:1:0 index:3:1" or "pid:101:0 pid:102:1".	—	""	5.0+
audio_pid	Audio Stream PID	[0-8192]	First PID with audio content.	4.0+
video_pid	Video Stream PID	[0-8192]	First PID with video content.	4.0+
srt_mode	Used SRT mode.	[listener, caller, rendezvous]	listener	4.4+
srt_passphrase	The password phrase for encryption.	Max. 80 characters.	""	4.4+
srt_latency	Acceptable latency in milliseconds.	—	125	4.4+

Element	Description	Range of Values	Default	Version
network_adapter	Defines the used network adapter. When this value is empty or set to "auto", the NIC is automatically selected. To use a specific NIC, use the NIC name or the configured IP address.	—	auto	5.0.1+
use_fieldrate_as_framerate	For interlaced HEVC decoding, if set to true, forces to use the detected rate as the frame rate. Default behavior uses the detected rate as the field rate.	[true, false]	false	5.4+
source_address	Defines the IGMP v3 source address for source-specific multicast connections. This can be used for UDP and RTP ingest.	—	""	5.5+

Input > Source (DeckLink Baseband)

```
<Device_DeckLink_Baseband id='Device_Decklink_0' stream='Video_Audio'>
  <cardindex value='1' />
  <capture_closed_captions value='true' />
  <VideoSource value='sdi' />
  <VideoFormat value='1080p60' />
  <AudioSource value='embedded' />
</Device_DeckLink_Baseband>
```

Note

Audio output channels can be selected from any input channel.

Tip

Use this audio processor option to select dedicated input channels from a captured multichannel signal to provide streams in different languages.

Element	Description	Range of Values	Default	Version
videosource	String of the input connector used for video.	[sdi, hdmi, opticalsdi, component, component_betacam, composite, composite_0_ire, svideo]	sdi	4.1+

Element	Description	Range of Values	Default	Version
videofORMAT	Input video format string starting with the vertical resolution, followed by a character for interlaced [i] or progressive [p], and the frame rate.	[480i29.97, 480i59.94, 480p29.97, 480i23.98, 576i25, 576i50, 576p25, 720p50, 720p59.94, 720p60, 1080i25, 1080i50, 1080i29.97, 1080i59.94, 1080i30, 1080i60, 1080p23.98, 1080p24, 1080p25, 1080p29.97, 1080p30, 1080p50, 1080p59.94, 1080p60, 2k23.98, 2k24, 2k25, 4k23.98, 4k24, 4k25, 4k29.97, 4k30, 4k50, 4k59.94, 4k60]	720p59.94	<ul style="list-style-type: none"> 4.1+ 4.5+ (2K and 4K formats)
audiosource	String of the input connector used for audio.	[embedded, aesebu, analog, analogxlr, analogrca]	embedded	<ul style="list-style-type: none"> 4.1+ (analogxlr) 4.5+ (analogrca)
cardindex	DeckLink card index number starting from 0.		0	4.1+
enableformatdetection	Enables the input video format detection if the capture card supports this feature.	[true, false]	true	4.3+
capture_closed_captions	Enables the close capture detection.	[true, false]	true	4.2+
closed_captions_vbi_line_1	First VBI line for analog video input.	21-23	21	4.2+
closed_captions_vbi_line_2	Second VBI line for analog video input.	284-286	284	4.2+
closed_captions_vanc_line	VANC line for digital input.	1-13, 15-19, 264-276, 278-282	9	4.2+
enable_ibs	Enables the in-band signal or cue tone detection.		false	4.2+
VancToVideoLine1	Copy VAN Line X to Video Line 1.	0 [disabled] - 30	20	4.3+
VancToVideoLine2	Copy VAN Line X to Video Line 2.	0 [disabled] - 30	0	4.3+
VancToVideoLine3	Copy VAN Line X to Video Line 3.	0 [disabled] - 30	0	4.3+
UseDeckLink4KExtremeCCTweak	Enable a workaround for capturing Analog CEA-608 Closed Captioning Information using a DeckLink 4K Extreme Card with drivers earlier than 10.0 (e.g. 9.8). It will not work with the new 10.0/10.1 drivers. The Card is identified by its name "DeckLink 4K Extreme". This workaround is enabled by default, so this entry can be used to disable it.	[true, false]	true	4.3+

Input > Source (AJA Baseband)

Element	Description	Range of Values	Default	Version
cardindex	AJA card index number starting from 0.		0	5.0+
capturemode	Configures the card to capture 4K or HD video signal. AJA Corvid HEVC can capture one 4K channel (4x 3G-SDI feeds), or up to four HD channels simultaneously.	[4k, hd]	hd	5.0+
videosource	String of the input connector used for video. For 4K capture this value should be an empty string.	[sdi, sdi1, sdi2, sdi3, sdi4] "sdi" and "sdi1" both select the first SDI connector.	sdi	5.0+

Input > Source (Default_Source)

Element	Description	Range of Values	Default	Version
mode	Mode used for video image. <ul style="list-style-type: none"> 0: Use internal image 1: Use image file from the src parameter. 	[0,1]	0	4.0+
src	Path to a JPEG or PNG image file. The "mode" value must be set to 1.	string	" "	4.0+

Preview Object

Adjustments to the following elements impact the preview image function.

! Important

Increasing these values results in a higher CPU usage.

Element	Description	Range of Values	Default	Version
frame_width	Preview image width in pixel.	number	356	4.0+
frame_height	Preview image height in pixel.	number	200	4.0+
framerate	Defines the update rate for the preview image generation in frames/sec. It makes no sense to set this value to a higher rate than the preview images are requested. This only results in a higher system load, especially if lots of channels are running.	number	1.0	4.0+
save_frames	Enable/Disable saving of preview JPEGs to disk.	[true, false]	false	4.2+
num_saved_frames	The number of preview JPEGs to keep on disk. If set to 1, the file name will not contain frame index.	number	20	4.2+

Element	Description	Range of Values	Default	Version
directory	Directory where preview JPEGs are saved.	min one character	"/opt/haivision/var/kulabyte/preview"	4.2+

Video_encoder Object

The video_encoder module defines the video encoding parameters.

Element	Description	Version
format	Object that includes details about the output video format (video framerate, video resolution, etc.). See Video_encoder > Format for descriptions of the format elements.	5.0+
processing	Object that includes interlacing options. See Video_encoder > Processing for description of the processing elements.	5.0+
encoder	Object that includes encoder options. See Video_encoder > Encoder for descriptions of the encoder elements.	5.0+

Video_encoder > Format

Element	Description	Range of Values	Default	Version
videoout_enable	Enable/Disable output video.	[0-1]	Input	4.0+
videoout_width	Output video width.	Divisible by 2	Input	4.0+
videoout_height	Output video height.	Divisible by 2	Input	4.0+
videoout_framerate	Output video frame rate: "Input Video Frame Rate / X", with X ≥ 1. The video encoder supports framerate groups. The encoder selects the best matching framerate output of this group depending on the input framerate. This improves the framerate configuration when the input framerate is not fixed at configuration time.	<ul style="list-style-type: none"> NTSC: [60, 59.94, 30, 29.97, 15, 14.985, 7.5, ...] PAL: [50, 25, 12.5, 6.25, ...] Framerate groups: ["50-60", "25-30", "12.5-15", "6.25-7.5"] 	Input	<ul style="list-style-type: none"> 4.0+ 5.0+ (framerate groups)

Video_encoder > Processing

Element	Description	Range of Values	Default	Version
Interpolation	Interpolation algorithm for resizing.	[NN, Linear, Cubic, Cubic2p_BSpline, Cubic2p_Catmullrom, Cubic2p_b05c03, Super, Lanczos]	Linear	4.0+

Element	Description	Range of Values	Default	Version
InterlaceMode	Specifies the algorithm used for deinterlacing interlaced content. "Progressive" disables deinterlacing; so in this case, the input should be progressive, as it will be treated as if it was progressive.	<ul style="list-style-type: none"> Disable: [Progressive] Enable: [Duplicate, Blend, Median, Edge_Detect, Median_Threshold, CAVT] 	Progressive	4.0+
DeinterlaceMode (Similar to "InterlaceMode")	"DeinterlaceMode" was added as an alternative name for "InterlaceMode", since it actually is a deinterlace mode. "Disable" and "Disabled" (in addition to "Progressive") were added as optional values in 4.1, to make the value more logical.	<ul style="list-style-type: none"> Disable: [Progressive, Disable, Disabled] Enable: [Duplicate, Blend, Median, Edge_Detect, Median_Threshold, CAVT] 	Disable	4.1+
hquality_deinterlace	Enables or disables the high quality deinterlace mode.	[true, false]	true	5.0+

Video_encoder > Encoder

The video_encoder > encoder object is a component object, as defined in [Component Objects](#). These encoders are supported for the type element in this object:

type Value	Description	Version
MC_AVC	H264/AVC encoder CPU (software)	4.0+
AVC_ENC	H264/AVC encoder CPU (software)	4.0+
HW_AVC_ENC	H264/AVC encoder GPU (hardware)	4.5+
HEVC_ENC	H265/HEVC encoder CPU (software)	4.5+
HW_HEVC_ENC	H265/HEVC encoder GPU (hardware)	5.0+

The following table lists the elements supported in the video_encoder > encoder > parameters object:

Element	Description	Range of Values	Default	Version
preset	Switching between channel configurations	MC_AVC, AVC_ENC: [H264_BASELINE, H264_MAIN, H264_HIGH, H264_HIGH422] HW_AVC_ENC: [H264_BASELINE, H264_MAIN, H264_HIGH] HEVC_ENC, HW_HEVC_ENC: [H264_MAIN]	H264_MAIN	4.3+

Element	Description	Range of Values	Default	Version
performance_pre set	Specifies the speed/quality encoding channel configuration.	[0-15] KB UI channel configuration options: <ul style="list-style-type: none"> Performance — 4 (x264 "Superfast", x265 "Ultrafast") Balanced — 7 (x264 "Faster", x265 "Faster") Quality — 12 (x264 "Slow", x265 "Medium") 	7 (Balanced)	4.3+
bitrate_mode	Switching between rate control modes.	[CBR, CQT, VBR, TQM]	CBR	4.3+
bitrate_avg	Video average output bit-rate (Kbps)	[1-288000]	Varies depending on resolution.	4.3+
BITRATE_MAX	Hypothetical stream scheduler rate (Kbps).	[bitrate_avg - 288000]	Varies depending on resolution.	4.3+
PROFILE	Select profile.	[Profile_Baseline, Profile_Main, Profile_High, Profile_High10, Profile_High422, Profile_High444]	Profile_Main	4.3+
LEVEL	Select level.	MC_AVC, AVC_ENC, HW_AVC_ENC: [Level_Auto, Level_1, Level_2, Level_3, Level_4, Level_5, Level_11, Level_12, Level_13, Level_21, Level_22, Level_31, Level_32, Level_41, Level_42, Level_51, Level_52] HEVC_ENC, HW_HEVC_ENC: [Level_Auto, Level_1, Level_2, Level_3, Level_4, Level_5, Level_6, Level_21, Level_31, Level_41, Level_51, Level_52, Level_61, Level_62]	Level_Auto	4.3+
gop_length	Max GOP size in frames.	[1-300]	90 (NTSC), 75 (PAL)	4.3+
gop_duration	Max GOP duration in milliseconds. If this value is greater than 0, it will be used instead of gop_length.	[0-10000]	0	5.0+
gop_min_length	Min GOP length.	[0-300]	<ul style="list-style-type: none"> 10 (4.3-4.5) 1 (5.0+) 	4.3-4.5 5.0+
gop_min_duration	Min GOP duration in milliseconds. If this value is greater than 0, it is used instead of gop_min_length. This value must be equal (which means a fix gop size) or smaller than gop_duration.	[0-10000]	0	5.0+

Element	Description	Range of Values	Default	Version
gop_fixed	Enable/Disable encoding with fixed GOP size. If set to true, GOP min length/duration is set to match GOP length/duration. If set to false, GOP min length/duration is set to 1 (if not set to other value in the channel configuration).	[true, false]	false	5.0+
gop_bcount	Max number of b-frames.	[0-3]	0	4.3+
slice_arg	Slice count	[0-number of MB rows]	1	4.3+
b_slice_pyramid	Enable/Disable pyramid coding	[true, false]	false	4.3+
adapt_b	Enable/Disable adaptive B-frames placement.	[true, false]	false	4.3+
vcsd_mode	Enable/Disable scene change detector.	[VSCDMode_IDR, VSCDMode_OFF]	VSCDMode_OFF	4.3+
bit_rate_buffer_size	Bit-rate buffer size (Kbps).	[1024-288000]	1150	4.3+
num_reference_frames	Number of reference frames.	[0-16]	2	4.3+
use_deblocking_filter	Enable deblocking filter.	[true, false]	true	4.3+
deblocking_alpha_CO_offset	Deblocking alpha c0 offset.	[-6-6]	-1	4.3+
deblocking_beta_offset	Deblocking beta offset.	[-6-6]	-1	4.3+
entropy_coding_mode	Switching between entropy coding modes.	[CABAC, CAVLC]	CABAC	4.3+
video_full_range	Enable/Disable full-range colors.	[true, false]	false	4.3+
write_aud_delimiters	Enable/Disable writing access unit delimiters.	[true, false]	true	4.3+
num_threads	Specifies maximum number of threads to be used.	[0-16]	0	4.3+
CALC_PSNR	Calculate overall PSNR or not.	[true, false]	false	4.3+
use_hrd	Specifies whether HRD conformance should be maintained or not. The option is only enabled for specific channel configurations, such as Blu-ray, Sony PSP, Apple iPod, etc. For general or custom channel configurations, it is always disabled.	[true, false]	true	4.3+
intra_refresh_mode	Specifies intra refresh mode (it may be useful for improving error robustness of encoded video).	[IntraRefreshMode_Off, IntraRefreshMode_Slow, IntraRefreshMode_Medium, IntraRefreshMode_Fast]	IntraRefreshMode_Off	4.3+

Element	Description	Range of Values	Default	Version
picture_struct	Enables or disables interlaced encoding. You do not need to disable deinterlacing with other parameters.	[interlaced, progressive]	progressive	5.5+

Audio_encoder Object

The audio_encoder module defines the audio encoding parameters.

Element	Description	Version
format	Object that includes details about the output audio format (audio bitrate, sample rate, number of channels, etc.). See Audio_encoder > Format for descriptions of the format elements.	5.0+
processing	Object that includes audio channel assignment options. See Audio_encoder > Processing for description of the processing elements.	5.0+
encoder	Object that includes encoder options. See Audio_encoder > Encoder for descriptions of the encoder elements.	5.0+

Audio_encoder > Format

Element	Description	Range of Values	Default	Version
audioout_enable	Enable/Disable output audio.	[0-1]	Input	4.0+
audioout_samplerate	Output audio sample rate. Must be the same value as the input sample rate.	[44100, 48000]	Input	4.0+
audioout_bitspersample	Output audio bitrate. Must be the same value as the input bits per sample value.	[16]	Input	4.0+
audioout_channels	Output audio channels number.	[1-2]	Input	4.0+

Audio_encoder > Processing

Element	Description	Range of Values	Default	Version
audio_channel_config	The audio output channels can be selected from any input channel. This option can be used to select a dedicated input channel from a captured multichannel audio signal, e.g., to provide streams in different languages. Therefore, the processing section of the audio stream contains one or more <audio_channel_config value="[input channel number]: [output channel number]"/> elements when the XML configuration file format is used. In JSON only one audio channel config value is used, but the values are separated by a vertical line, " ". All entries with channel numbers above the max channels defined in the input or output format definition are ignored. The channel number range is from 0 to max_channels-1.	[input channel number]:[output channel number] example: "2:0" input channel 2 to output channel 0.	In the default configuration, the first two channels are used for audio output or stream number 0 if the stream number is defined (≠-1)	4.0+
audio_stream_number	The number of the audio stream that the encoder should use. When the input sources uses stream numbers this parameter must be added. The value -1 means that this stream index should be ignored.	[-1-31]	-1	5.0+

Audio_encoder > Encoder

The audio_encoder > encoder object is a component object, as defined in [Component Objects](#). These encoders are supported for the type element in this object:

type Value	Description	Version
AAC_ENC	Advanced Audio Coding (AAC)	4.0+

The following table lists the elements supported in the audio_encoder > encoder > parameters object:

Element	Description	Range of Values	Default	Version
bitrate_avg	Audio average output bitrate (Kbps).	[6, 7, 8, 10, 12, 14, 16, 20, 24, 28, 32, 40, 48, 56, 64, 80, 96, 112, 128, 160, 192, 224, 256, 320, 384, 448, 512, 1024]	160	4.3+
object_type	Audio object type (AAC profile).	[LC, LTP, SBR, LD, PS, ELD]	LC	4.3+
ns_Mode	Noise shaping model.	[Simple, Advanced]	Simple	4.3+

Element	Description	Range of Values	Default	Version
langname	Contains the language name assigned to the audio encoder.	—	—	5.0+
langcode	Contains the language code according to ISO-639-2 assigned to audio encoder.	3 characters	—	5.0+
langcomment	Contains the language comment assigned to the audio encoder.	—	—	5.0+

Output Object

The output object is a component object. These input sources are supported for the type element in this object:

type Value	Description	Version
RTMP_Sink	Streams to a RTMP streaming server.	4.0+
HTTP_Sink	Streams to a HLS server.	4.0+
File	Writes the streams into MP4 files.	4.0+
TS_Sink	Sends MPEG TS streams to SRT destinations.	4.4+
DASH_MP4_Sink	Streams to a DASH server.	5.0+

Output > Parameters (Common)

Element	Description	Range of Values	Default	Version
audioencoder	Contains the IDs of audio encoders separated by ' '. This option is used for encode models required to support multiple streams per output, e.g. for DASH.	—	—	5.0+
videoencoder	Contains the IDs of video encoders separated by ' '. This options is used for new encode models required to support multiple streams per output, e.g. for DASH.	—	—	5.0+
langnames	Contains the language name for each audio encoder separated by ' '. The values can be empty, but at least the separator ' ' must be there to ensure that the names are assigned to the correct audio encoder.	—	—	5.0+
langcodes	Contains the language code according to ISO-639-2 for each audio encoder separated by ' '. The values can be empty, but at least the separator ' ' must be there to ensure that the names are assigned to the correct audio encoder.	3 characters	—	5.0+

Element	Description	Range of Values	Default	Version
langcomments	Contains the language comment for each audio encoder separated by ' '. The values can be empty, but at least the separator ' ' must be there to ensure that the names are assigned to the correct audio encoder.	—	—	5.0+
cc_langnames	Contains the language name for each CC channel separated by ' '. The values can be empty, but at least the separator ' ' must be there to ensure that the names are assigned to the correct CC channel.	—	—	5.0+
cc_langcodes	Contains the language code according to ISO-639-2 for each CC channel separated by ' '. The values can be empty, but at least the separator ' ' must be there to ensure that the names are assigned to the correct CC channel.	3 characters	—	5.0+
cc_langcomments	Contains the language comment for each CC channel separated by ' '. The values can be empty, but at least the separator ' ' must be there to ensure that the names are assigned to the correct CC channel.	—	—	5.0+
enable_scte35	SCTE-35 messages are delivered when this parameter is set to true.	[true, false]	true	5.0+

Output > Parameters (RTMP)

Element	Description	Range of Values	Default	Version
server_url	Flash Media Server URL.	—	rtmp://localhost/live	4.0+
stream_name	Stream name.	—	Livestream	4.0+
agent_id	Name of the RTMP agent.	—	KulaByte/<Version Major>.<Version Minor>; e.g., "KulaByte 4.5.1"	4.0+
playback_url	Stream playback URL.	—	—	4.0+
use_stream_bundling	Activate to send multiple streams through one RTMP connection.	[true, false]	false	4.0+
username	User name for authentication with FMS server. This value could be empty if no authentication is required.	—	—	4.0+
password	Password for authentication with FMS server. This value could be empty if no authentication is required.	—	—	4.0+
continuous_stream	Keeps output RTMP timestamps continuous.	[true,false]	true	4.0+
republish_interval	Time between attempts to publish the stream (in ms).	—	10000	4.0+

Element	Description	Range of Values	Default	Version
enable_bandwidth_control	Enables the adaptive bandwidth control function.	[true,false]	false	4.0+
bitrate_change_value_level1	Sets the value that the bandwidth control function uses to reduce the video bitrate if it detects a small bitrate difference (level1). This value is in percent of the current video encoder bitrate.	[1-99]	10	4.0+
bitrate_change_value_level2	Sets the value that the bandwidth control function uses to reduce the video bitrate if it detects bitrate difference as level 1. This value is in percent of the current video encoder bitrate. It should be higher than the BitrateChangeValueLevel1 value.	[1-99]	20	4.0+
bitrate_change_value_level3	Sets the value that the bandwidth control function uses to reduce the video bitrate if it detects bitrate difference as level 2. This value is in percent of the current video encoder bitrate. It should be higher than the BitrateChangeValueLevel2 value.	[1-99]	30	4.0+
bitrate_change_value_level4	Sets the value that the bandwidth control function uses to reduce the video bitrate if it detects bitrate difference as level 3. This value is in percent of the current video encoder bitrate. It should be higher than the BitrateChangeValueLevel3 value.	[1-99]	40	4.0+
min_bitrate	Minimum video bitrate in percent of the start video encoder bitrate. This value is the minimum value that the bandwidth control function sets to the video encoder.	[1-50]	20	4.0+
cc_in_metadata	Send CC data in onMetadata messages.	[true, false]	false true	4.5+ 4.2
cc_in_cuepoint	Send CC data in onCuePoint messages.	[true, false]	false true	4.5+ 4.2
cc_in_captioninfo	Send CC data in onCaptionInfo message.	[true, false]	true	4.2+
cc_in_textdata	Send CC data in onTextData messages.	[true, false]	false	4.5+
unpublish_on_discontinuity	Enables/disables stream republishing on a discontinuity. If disabled, ContinuousStream parameter is set to "true."	[true, false]	false	4.1+
auth_mode	Authentication mode.	[auto, adobe, limelight, akamai]	auto	4.5+

Element	Description	Range of Values	Default	Version
scte35_adcue	Sends ad insertion commands as onAdCue RTMP messages.	[true, false]	false	5.1+
scte35_cuepoint	Sends ad insertion commands as onCuePoint RTMP messages.	[true, false]	false	5.1+
scte35_cuemessage	Sends ad insertion commands as onCue RTMP messages.	[true, false]	false	5.1+

Output > Parameters (HLS)

Element	Description	Range of Values	Default	Ver
server_url	HLS server URL.	—	—	5.0+
playlist_name		—	playlist.m3u8	5.0+
segment_duration	The segment duration in milliseconds. The video GOP size must be lower than the segment duration to make sure each segment contains at least one key frame.	5.0-5.4 [1000-30000] 5.5+ [500-30000]	5.0-5.4 10000 5.5+ 6000	5.0+
segment_count	Number of segments that appear in a playlist.	[3- 5000]	10	5.0+
playback_url	An optional playback URL. If this value is empty, the playback URL is created as output of the server_url and playlist_name.	—	—	5.0+
backup_url	An optional playback URL for the backup streams. If not set, backup stream entries are not added to the master playlist.	—	—	5.0+
username	A username is required for an HTTP upload.	—	—	5.0+
password	A password is required for an HTTP upload.	—	—	5.0+
default_language	Contains the language code of the default language.	—	—	5.0+
default_video	Contains the encoder id of the video stream that should be used as default stream.	—	—	5.0+
var_playlist_url	Master playlist URL.	—	—	4.0+
var_playlist_username	Username for authentication with HTTP server (master playlist). This value could be empty if no authentication is required.	—	—	4.0+
var_playlist_password	Password for authentication with HTTP server (master playlist). This value could be empty if no authentication is required.	—	—	4.0+
keep_all_segments	Do not delete segments from the server.	[true, false]	false	5.0+

Element	Description	Range of Values	Default	Ver
live_mode	If enabled (default), only the most recent <segment_count> segments are included in the playlist. If disabled, segments are not removed from the playlist; EXT-X-PLAYLIST-TYPE tag is added with the value of EVENT while stream runs, changing to VOD after stream stop.	[true, false]	true	5.0+
directory_rollover	Enable creation of stream and volume directories on the server. This parameter only adds stream and volume directories segment/playlist URLs. If the server does not support automatic directory creation, create_folders must be set to true.	[true, false]	false	5.0+
use_file_counter_zero_padding	Use zero padding for volume, session and segment.	[true, false]	true	5.0+
create_folders	Create stream/volume folders on the server using MKCOL requests.	[true, false]	false	5.0+
segments_per_volume	Number of segments per volume folder.	5-3000	2000	5.0+
enable_cache_off	Add the line "#EXT-X-ALLOW-CACHE:NO " to playlist.	[true, false]	false	5.0+
disable_verify_peer	Disable HTTPS certificate exchange.	[true, false]	false	5.0+
cert_file	HTTPS certificate file.	—	—	5.0+
enable_webvtt	Enable WebVTT subtitles.	[true, false]	false	5.0+
use_absolute_urls	Use absolute URLs in the master playlist (otherwise, relative to master playlist location).	[true, false]	true	5.0.1+
mux_vbr_mode	Enable VBR mode (no padding) for TS multiplexer.	[true, false]	true	5.0.1+
upload_scheme_detection	Automatically detect upload mode, based on substrings found in upload URL.	[true, false]	true	5.4+
upload_scheme	Define the used upload mode for segments and playlists. <ul style="list-style-type: none"> http_put: Use HTTP PUT requests. http_post: Use HTTP POST requests limelight_hls: Use authentication and upload requests required for Limelight HLS ingest. 	[http_put, http_post, limelight_hls]	http_put	5.2+
customer_id	Customer ID to use with Limelight upload scheme.	—	—	5.4+

Element	Description	Range of Values	Default	Ver
enable_hls_buffer	Enable the HLS segment buffer function. When a segment upload is not possible the segments are not dropped until the hls_buffer_duration limit is exceeded. Requires sufficient free disk space to cache HLS segments.	[true, false]	false	5.3+
hls_buffer_duration	Defines the maximum HLS segment buffer duration in seconds.	60-86400	300	5.3+
hls_buffer_folder	Defines the folder used to buffer HLS segments.	—	/assets/cache/hls	5.3+
container	Defines the container format for video and audio segments. HEVC video streams are always contained into fMP4 segments. <ul style="list-style-type: none"> • auto: When the assigned video stream contains an HEVC stream, then fMP4 is used for all video and audio streams. Otherwise MPEG-TS is used. • ts: The H264 video and AAC audio streams are packed into MPEG-TS segments. • fmp4: The H264 video and AAC audio streams are packed into fMP4 segments. 	[auto, ts, fmp4]	auto	5.3+
enable_ext_gap	When true, URLs for the segments that failed to upload are added to the playlist together with EXT-X-GAP tag. When false, EXT-X-DISCONTINUITY tags are used instead.	[true, false]	false	5.5+

Output > Parameters (File)

Element	Description	Range of Values	Default	Version
directory	<p>Directory of the output media file. If this parameter is omitted, the encoded file is automatically saved in the default recording directory.</p> <p>4.2+ With version 4.2 the start time sub folder isn't added anymore. Now it is possible to save all recording files from different encodes into one folder.</p>	<p>Minimum one character.</p> <p>4.3+ The folder name can contain the following placeholders:</p> <ul style="list-style-type: none"> • %year% – four-digit year value. • %month% – two-digit month value • %day% – two-digit day value. • %hour% – two-digit 24-hour value. • %minute% – two-digit minute value. • %second% – two-digit second value. <p>The time value is creation time (default value) of a new recording or the start time of the encode, depending on the "UseStartTime" parameter.</p>	<p>"/assets/archive"</p> <p>"/assets/archive/%year%-%month%-%day%-%hour%-%minute%-%second%"</p>	<p>4.0-4.2</p> <p>4.3+</p>

Element	Description	Range of Values	Default	Version
filename	File name of the output media file.	<p>Minimum of one character</p> <p>4.3+ The folder name could contain the following place holders:</p> <ul style="list-style-type: none"> • %year% – Four digits year value. • %month% – Two digits month value. • %day% – Two digits day value. • %hour% – Two digits 24-hour value. • %minute% – Two digits minute value. • %second% – Two digits second value. • %index% – Four digits index value. <p>The time value is creation time (default value) of a new recording file or the start time of the encode , depending on the "UseStartTime" parameter. If the file name doesn't contain an %index% place holder, the index is added to the end of the file name. The *.srt files with CC content contain the channel number [1-4] with a dash (e.g. -1) at the end of the file name plus the new .srt suffix. Together with the new folder configuration it is possible that recording files with same names but from difference encode/transcode sessions could be created. In these cases, the already existing files would be overwritten! To avoid this, ensure that the recording file paths are different between different encode/transcode runs.</p>	"Archive"	4.0+
OutputFormat	Output file format.	[MP4, JPEG2000, 3GPP]	MP4	4.0-4.1
Compatibility	Sets the compatibility format.	[Standard, PSP, iPod, ISMA, QuickTime, Flash, iPhone]	Standard	4.0-4.1
VideoSequenceLength	Audio/Video interleaving size	—	10	4.0+
StartFrame	Timecode offset in frames	—	—	4.0-4.1
AtomOrder	Sets the atom order.	[MoovFirst, MdatFirst]	MoovFirst	4.0-4.1
CaptureMode	Enables/disables live capture mode.	[true, false]	—	4.0-4.1

Element	Description	Range of Values	Default	Version
KeepAUD	Keeps the AU Delimiter for H.264/AVC muxing.	[true, false]	—	4.0-4.1
KeepPS	Keeps the SPS and PPS for H.264/AVC muxing.	[true, false]	—	4.0-4.1
RemoveSEI	Remove SEI.	[true, false]	false	4.0-4.1
SmallAtoms	Force the usage of small atoms.	[true, false]	—	4.0-4.1
LiveMode	Enables/disables the use of the stream time from the input media samples.	[true, false]	false	4.0-4.1
MediaTimeOffset	Media time offset for "edts" atom.	—	—	4.0-4.1
RemoveFiller	Remove filling bytes from NAL units.	[true, false]	false	4.0-4.1
SplitOnDuration	Enables/Disables splitting of files based on file duration. If the maximum file size (2GB) is reached before the duration is met, a new file will be started, regardless of the time specification.	[true, false]	false	4.0+
FileDuration	The duration of each file in seconds before a new chapter is automatically started.	—	14400 (4 hours)	4.0+
SplitOnSize	Enables/Disables splitting of files based on file size.	[true, false]	false	4.0+
FileSize	Specifies the size of each file in KB before a new chapter is automatically started.	—	2097152 (2 GB)	4.0+
SplitOnManual	Enables/Disables the manual splitting of files.	[true, false]	true	4.0+
AutoStart	Automatically start archiving on encode start.	[true, false]	false	4.0+
TempDirectory	Directory for temp files.	—	—	4.0+

Output > Parameters (MPEG TS)

Element	Description	Range of Values	Default	Version
url	Video stream URL " srt://<ip-address>:<port> ", e.g., " srt://10.20.30.40:1234 ", where the IP address is the receiving machine. If SRT_MODE is set to "listener" the URL must be " srt://0.0.0.0:<port> ". 4.5: New supported protocols are: "udp://" and "rtp://".	—	—	4.4+

Element	Description	Range of Values	Default	Version
port	Destination port number. Is only required if the URL doesn't contain a port.	—	—	4.4+
ttl	Configure the used time-to-live value.	≥ 1	20	4.4+
mtu	Used maximum transmission unit.	[280-1500]	1496	4.4+
tos	Used type of service value	Byte value in hex format.	0xB8 --> "Low Delay"	4.4+
video_pid	Contains the PID value that should be used for the video stream. When more than one video encoder is assigned to the output, the PIDs for each video stream must be separated by a ' ' and in the same order as in video_encoder.	—	—	5.0+
audio_pid	Contains the PID value that should be used for the audio stream. When more than one audio encoder is assigned to the output the PIDs for each audio stream must be separated by a ' ' and in the same order as in audio_encoder.	—	—	5.0+
scte35_pid	Contains the PID value that should be used for a SCTE-35 stream.	—	309 (0x135)	5.0+
scte35_descriptors	Specifies which SCTE-35 descriptors are added to PMT: no descriptors, Cue Identifier descriptor, Stream Identifier descriptor, or both.	[none, cue, stream, cue_stream]	cue_stream	5.5+
scte35_cue_stream_type	The value for the cue_stream_type field in the Cue Identifier descriptor. Default is 0x01 (All Commands).	[0-255]	1	5.5+
srt_latency	Acceptable latency in milliseconds	—	125	4.4+
srt_overhead_bandwidth	Acceptable Bandwidth Overhead in Percent in the Range of 5 to 100	[5-100]	25	4.4+
srt_password	The password phrase for the encryption.	Minimum of 10 and maximum of 80 characters.	""	4.4+
srt_encryption	Used encryption mode and key length.	[none, aes128, aes192, aes256]	none	4.4+
srt_mode	Used SRT mode.	[listener, caller, rendezvous]	listener	4.4+
in_video_queue_limit	Max count of video frames stored in the input queue.	[4-2000]	80	4.4+
in_audio_queue_limit	Max count of audio packets stored in the input queue.	[4-2000]	160	4.4+
network_adapter	Network interface to use for the stream. Either its name (e.g. "eth0") or IP address (e.g. "10.20.30.40").	—	auto	5.0.1+

Output > Parameters (MPEG DASH)

Element	Description	Range of Values	Default	Version
<code>serverurl</code>	Main ingest URL for upload.	A valid URL with http or https protocol.	—	5.0+
<code>baseurl</code>	Contains the base URL for playback. The complete playback URL contains the baseurl and manifest_name. <code><baseurl>/<manifest_name></code>	A valid URL with http or https protocol.	—	5.0+
<code>baseurl_backup</code>	Contains the backup base URL for playback.	A valid URL with http or https protocol.	—	5.0+
<code>manifest_name</code>	Manifest file name	—	manifest.mpd	5.0+
<code>segment_duration</code>	Segment duration in seconds	1 to 10 Segment duration must be greater than the GOP duration.	6	5.0+
<code>timeshift_depth</code>	Used timeshift depth in seconds.	30 to 3600	300	5.0+
<code>username</code>	Username required for upload.	—	—	5.0+
<code>password</code>	Password required for upload.	—	—	5.0+

Global Encoder Parameters

Global encoder parameters are used to overwrite default or preset settings for various type of components within an encoder. These parameters are stored in the following file:

```
/opt/haivision/etc/param.xml
```

The `param.xml` file is a simple XML file with an entry for every single parameter.

```
<config>
  <param comp="MC_AVC" param="cpu_optimization" value="CPU_SSE3"/>
</config>
```

The above example sets the CPU optimization for all AVC encoders to SSE3 mode (default was auto detection).

These settings are applied to the components after the channel is loaded and the corresponding media pipeline is created and configured.

- The "comp" attribute contains the type of *target component* for the parameter.
- The "param" and the "value" attributes contain the *parameter/value pairs*.

Target Component for Parameter (comp)	Component
RTMP_Source	RTMP source component
Default_Source	Default image component
TS_Source	MPEG Transport stream source component
AVC_ENC	AVC encoder component
AAC_ENC	AAC encoder component
RTMP_Sink	RTMP sink or output component
INPUT_Mux	Input multiplexer
Device_DeckLink_Baseband	Decklink source card component (4.1)
HTTP_Sink	HLS sink or output component
File	Recording file (MP4 format) output component
Src_Jpeg_Sink	Preview JPEG image component
TS_Sink	MPEG TS output SRT, UPD and RTP
HW_AVC_ENC	Hardware AVC encoder
HEVC_ENC	HEVC encoder
HW_HEVC_ENC	Hardware HEVC encoder
HLS_Sink	Extended HSL output

Target Component for Parameter (comp)	Component
DASH_MP4_Sink	MPEG DASH output
Device_AJA_Baseband	Input component for AJA capture cards

Warranties

1-Year Limited Hardware Warranty

Haivision warrants its hardware products against defects in materials and workmanship under normal use for a period of ONE (1) YEAR from the date of equipment shipment ("Warranty Period"). If a hardware defect arises and a valid claim is received within the Warranty Period, at its option and to the extent permitted by law, Haivision will either (1) repair the hardware defect at no charge, or (2) exchange the product with a product that is new or equivalent to new in performance and reliability and is at least functionally equivalent to the original product. A replacement product or part assumes the remaining warranty of the original product or ninety (90) days from the date of replacement or repair, whichever is longer. When a product or part is exchanged, any replacement item becomes your property and the replaced item becomes Haivision's property.

EXCLUSIONS AND LIMITATIONS

This Limited Warranty applies only to hardware products manufactured by or for Haivision that can be identified by the "Haivision" trademark, trade name, or logo affixed to them. The Limited Warranty does not apply to any non-Haivision hardware products or any software, even if packaged or sold with Haivision hardware. Manufacturers, suppliers, or publishers, other than Haivision, may provide their own warranties to the end user purchaser, but Haivision, in so far as permitted by law, provides their products "as is".

Haivision does not warrant that the operation of the product will be uninterrupted or error-free. Haivision does not guarantee that any error or other non-conformance can or will be corrected or that the product will operate in all environments and with all systems and equipment. Haivision is not responsible for damage arising from failure to follow instructions relating to the product's use.

This warranty does not apply:

- (a) to cosmetic damage, including but not limited to scratches, dents and broken plastic on ports;
- (b) to damage caused by accident, abuse, misuse, flood, fire, earthquake or other external causes;
- (c) to damage caused by operating the product outside the permitted or intended uses described by Haivision;
- (d) to a product or part that has been modified to alter functionality or capability without the written permission of Haivision; or
- (e) if any Haivision serial number has been removed or defaced.

TO THE EXTENT PERMITTED BY LAW, THIS WARRANTY AND REMEDIES PROVIDED ABOVE ARE EXCLUSIVE AND IN LIEU OF ALL OTHER WARRANTIES, REMEDIES AND CONDITIONS, WHETHER ORAL OR WRITTEN, STATUTORY, EXPRESS OR IMPLIED. AS PERMITTED BY APPLICABLE LAW, HAIVISION SPECIFICALLY DISCLAIMS ANY AND ALL STATUTORY OR IMPLIED WARRANTIES,

INCLUDING, WITHOUT LIMITATION, WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND WARRANTIES AGAINST HIDDEN OR LATENT DEFECTS. IF HAIVISION CANNOT LAWFULLY DISCLAIM STATUTORY OR IMPLIED WARRANTIES THEN TO THE EXTENT PERMITTED BY LAW, ALL SUCH WARRANTIES SHALL BE LIMITED IN DURATION TO THE DURATION OF THIS EXPRESS WARRANTY AND TO REPAIR OR REPLACEMENT SERVICE AS DETERMINED BY HAIVISION IN ITS SOLE DISCRETION. No Haivision reseller, agent, or employee is authorized to make any modification, extension, or addition to this warranty. If any term is held to be illegal or unenforceable, the legality or enforceability of the remaining terms shall not be affected or impaired.

EXCEPT AS PROVIDED IN THIS WARRANTY AND TO THE EXTENT PERMITTED BY LAW, HAIVISION IS NOT RESPONSIBLE FOR DIRECT, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES RESULTING FROM ANY BREACH OF WARRANTY OR CONDITION, OR UNDER ANY OTHER LEGAL THEORY, INCLUDING BUT NOT LIMITED TO LOSS OF USE; LOSS OF REVENUE; LOSS OF ACTUAL OR ANTICIPATED PROFITS (INCLUDING LOSS OF PROFITS ON CONTRACTS); LOSS OF THE USE OF MONEY; LOSS OF ANTICIPATED SAVINGS; LOSS OF BUSINESS; LOSS OF OPPORTUNITY; LOSS OF GOODWILL; LOSS OF REPUTATION; LOSS OF, DAMAGE TO OR CORRUPTION OF DATA; OR ANY INDIRECT OR CONSEQUENTIAL LOSS OR DAMAGE HOWSOEVER CAUSED INCLUDING THE REPLACEMENT OF EQUIPMENT AND PROPERTY, ANY COSTS OF RECOVERING, PROGRAMMING, OR REPRODUCING ANY PROGRAM OR DATA STORED OR USED WITH HAIVISION PRODUCTS AND ANY FAILURE TO MAINTAIN THE CONFIDENTIALITY OF DATA STORED ON THE PRODUCT. THE FOREGOING LIMITATION SHALL NOT APPLY TO DEATH OR PERSONAL INJURY CLAIMS, OR ANY STATUTORY LIABILITY FOR INTENTIONAL AND GROSS NEGLIGENT ACTS AND/OR OMISSIONS.

OBTAINING WARRANTY SERVICE

Before requesting warranty service, please refer to the documentation accompanying this hardware product and the Haivision Support Portal <https://support.haivision.com>. If the product is still not functioning properly after making use of these resources, please contact Haivision or Authorized Reseller using the information provided in the documentation. When calling, Haivision or Authorized Reseller will help determine whether your product requires service and, if it does, will inform you how Haivision will provide it. You must assist in diagnosing issues with your product and follow Haivision's warranty processes.

Haivision may provide warranty service by providing a return material authorization ("RMA") to allow you to return the product in accordance with instructions provided by Haivision or Authorized Reseller. You are fully responsible for delivering the product to Haivision as instructed, and Haivision is responsible for returning the product if it is found to be defective. Your product or a replacement product will be returned to you configured as your product was when originally purchased, subject to applicable updates. Returned products which are found by Haivision to be not defective, out-of-warranty or otherwise ineligible for warranty service will be shipped back to you at your expense. All replaced products and parts, whether under warranty or not, become the property of Haivision. Haivision may require a completed pre-authorized form as security for the retail price of the replacement product. If you fail to return the replaced product as instructed, Haivision will invoice for the pre-authorized amount.

APPLICABLE LAW

This Limited Warranty is governed by and construed under the laws of the Province of Quebec, Canada.

This Limited Hardware Warranty may be subject to Haivision's change at any time without prior notice.

EULA - End User License Agreement

READ BEFORE USING

THE LICENSED SOFTWARE IS PROTECTED BY COPYRIGHT LAWS AND TREATIES. READ THE TERMS OF THE FOLLOWING END USER (SOFTWARE) LICENSE AGREEMENT ("AGREEMENT") CAREFULLY BEFORE ACCESSING THE LICENSED SOFTWARE. BY SCANNING THE QR CODE TO REVIEW THIS AGREEMENT AND/OR ACCESSING THE LICENSED SOFTWARE, YOU CONFIRM YOUR ACCEPTANCE OF THIS AGREEMENT. IF YOU DO NOT AGREE TO THESE TERMS, HAIVISION IS UNWILLING TO LICENSE THE LICENSED SOFTWARE TO YOU AND YOU ARE NOT AUTHORIZED TO ACCESS THE LICENSED SOFTWARE.

Click the following link to view the Software End-User License Agreement: [Haivision EULA.pdf](#)

If you have questions, please contact legal@haivision.com

SLA - Service Level Agreement

1. Introduction

This Service Level and Support supplement forms a part of and is incorporated into the Service Agreement (the "Agreement") between You and Haivision Network Video Inc. ("Haivision"). Capitalized terms used but not otherwise defined in this supplement shall have the meaning ascribed to them in the Agreement. Haivision may, upon prior written notice to You, amend this supplement to incorporate improvements to the service levels and support commitments at no additional cost to You. This supplement applies only to those products and services set forth below.

2. Definitions

- "Audience Member" means an individual or entity that accesses Your Published Media Objects through a public URL.
- "Access Service" means the service provided by Haivision VCMS that verifies an Audience Member's credentials.
- "Digital Media File" means a computer file containing text, audio, video, or other content.
- "Outage" is a 12-minute period of consecutive failed attempts by all six agents to PING the domain on the Haivision Streaming Media network.
- "Published Media Object" means a Digital Media File with a public URL.
- "Transaction" means the creation of a right for an Audience Member to access a Media Object and the completion of an order logged in the order history service.

3. Service Levels for the Video Content Management System

The service levels in this [Section 3](#) apply only to the hosted version of Haivision VCMS and the Haivision VCMS development kit (collectively, the "Standard Hosted Components" of Haivision Video Cloud Services). Subject to the exceptions noted in [Section 4](#) below, the aforementioned components of Haivision Video Cloud Services will be available for use over the course of each calendar month as follows:

Type of Access	Definition	Availability Level
Write Functions	<ul style="list-style-type: none"> • Access to all functions through the administrative user interface. • Ability to add or modify objects and metadata through the application programming interface (“API”) • Ability of ingest service to check for new or updated files or feeds 	99.999%
Read-Only Functions	<ul style="list-style-type: none"> • Ability to retrieve data through the API • Ability for Audience Members to authenticate through the Access Service • Ability for Audience Members to play Published Media Objects • Ability for Audience Members to play Haivision VCMS-authenticated or entitled Published Media Objects • Ability to complete Transactions 	99.999%

4. Exceptions to Availability for the VCMS

The Standard Hosted Components may not be available for use under the following circumstances, and in such case such periods of unavailability shall not be counted against Haivision Video Cloud for purposes of calculating availability:

- a. Normal Maintenance, Urgent Maintenance and Upgrades as defined in the table below;
- b. Breach of the Agreement by You as defined in the Agreement;
- c. The failure, malfunction, or modification of equipment, applications, or systems not controlled by Haivision Video Cloud;
- d. Any third party, public network, or systems unavailability;
- e. Acts of Force Majeure as defined in the Agreement;
- f. Modification of software made available to You as part of Haivision Video Cloud Services by You or a third party acting on Your behalf; and
- g. Any third party product or service not incorporated into Haivision Video Cloud Services or any third party plug-in.

Haivision Video Cloud shall make commercially reasonable efforts to notify, or work with, applicable third parties to repair or restore Haivision VCMS functionality affected by such exceptions.

Type of Maintenance	Purpose	Write Functions Available	Read Functions Available	Maximum Time Per Month	Continuous Time in Mode (Max)	Window (Central Time)	Min Notice
Normal	<ul style="list-style-type: none"> • Preventive maintenance on the software/hardware components of Haivision VCMS • Addition of new features/functions • Repair errors that are not immediately affecting Your use of Haivision VCMS 	No	Yes	10 Hours	6 Hours	10:00p m - 5:00a m	48 Hours
Urgent	<ul style="list-style-type: none"> • Repair errors that are immediately affecting Your use of Haivision VCMS 	No	Yes	30 Minutes	15 Minutes	Any Time	3 Hours

Type of Maintenance	Purpose	Write Functions Available	Read Functions Available	Maximum Time Per Month	Continuous Time in Mode (Max)	Window (Central Time)	Min Notice
Upgrades	<ul style="list-style-type: none"> Perform upgrades on software or hardware elements necessary to the long term health or performance of Haivision VCMS, but which, due to their nature, require that certain components of Haivision VCMS to be shut down such that no access is possible 	No	No	1 Hour	1 Hour	12:00am - 4:00am M-F	5 Days

5. Credits for Downtime for the VCMS

Haivision Video Cloud will grant a credit allowance to You if You experience Downtime in any calendar month and you notify Haivision Video Cloud thereof within ten (10) business days after the end of such calendar month. In the case of any discrepancy between the Downtime as experienced by You and the Downtime as measured by Haivision Video Cloud, the Downtime as measured by Haivision Video Cloud shall be used to calculate any credit allowance set forth in this section. Such credit allowance shall be equal to the pro-rated charges of one-half day of Fees for each hour of Downtime or fraction thereof. The term “Downtime” shall mean the number of minutes that Standard Hosted Components are unavailable to You during a given calendar month below the availability levels thresholds in [Section 3](#), but shall not include any unavailability resulting from any of the exceptions noted in [Section 4](#). Within thirty (30) days after the end of any calendar month in which Downtime occurred below the availability levels thresholds in [Section 3](#), Haivision Video Cloud shall provide You with a written report detailing all instances of Downtime during the previous month. Any credit allowances accrued by You may be offset against any and all Fees owed to Haivision Video Cloud pursuant to the Agreement, provided that a maximum of one month of credit may be accrued per month.

6. Support Services for the VCMS

Support for Haivision Video Cloud Services as well as the Application Software (defined as the VCMS application software components that Haivision licenses for use in conjunction with the Video Cloud Services) can be reached at hvc-techsupport@haivision.com and shall be available for all Your support requests. Haivision Video Cloud will provide 24x7 monitoring of the Standard Hosted Components.

Cases will be opened upon receipt of request or identification of issue, and incidents will be routed and addressed according to the following:

Severity Level	Error State Description	Status Response Within	Incident Resolution within
1 - Critical Priority	Renders Haivision VCMS inoperative or causes Haivision VCMS to fail catastrophically.	15 minutes	4 hours
2 - High Priority	Affects the operation of Haivision VCMS and materially degrades Your use of Haivision VCMS.	30 minutes	6 hours
3 - Medium Priority	Affects the operation of Haivision VCMS, but does not materially degrade Your use of Haivision VCMS.	2 hours	12 hours

Severity Level	Error State Description	Status Response Within	Incident Resolution within
4 - Low Priority	Causes only a minor impact on the operation of Haivision VCMS.	1 business day	3 business days

7. Service Levels for Haivision Streaming Media Service

Haivision agrees to provide a level of service demonstrating 99.9% Uptime. The Haivision Streaming Media Service will have no network Outages.

The following methodology will be employed to measure Streaming Media Service availability:

Agents and Polling Frequency

- a. From six (6) geographically and network-diverse locations in major metropolitan areas, Haivision’s Streaming Media will simultaneously poll the domain identified on the Haivision Streaming Media network.
- b. The polling mechanism will perform a PING operation, sending a packet of data and waiting for a reply. Success of the PING operation is defined as a reply being received.
- c. Polling will occur at approximately 6-minute intervals.
- d. Based on the PING operation described in (b) above, the response will be assessed for the purpose of measuring Outages.

If an Outage is identified by this method, the customer will receive (as its sole remedy) a credit equivalent to the fees for the day in which the failure occurred.

Haivision reserves the right to limit Your use of the Haivision Streaming Media network in excess of Your committed usage in the event that Force Majeure events, defined in the Agreement, such as war, natural disaster or terrorist attack, result in extraordinary levels of traffic on the Haivision Streaming Media network.

8. Credits for Outages of Haivision Streaming Media Service

If the Haivision Streaming Media network fails to meet the above service level, You will receive (as your sole remedy) a credit equal to Your or such domain’s committed monthly service fee for the day in which the failure occurs, not to exceed 30 days of fees.

9. No Secondary End User Support

UNDER NO CIRCUMSTANCES MAY YOU PROVIDE CONTACT INFORMATION FOR HAIVISION SERVICES TO CUSTOMERS OR AUDIENCE MEMBERS OR OTHER THIRD PARTIES WITHOUT HAIVISION’S EXPRESS PRIOR WRITTEN CONSENT.

Getting Help

<p>General Support</p>	<p>North America (Toll-Free) 1 (877) 224-5445</p> <p>International 1 (514) 334-5445</p> <p><i>and choose from the following:</i> Sales - 1, Cloud Services - 3, Support - 4</p>
<p>Managed Services</p>	<p>U.S. and International 1 (512) 220-3463</p>
<p>Fax</p>	<p>1 (514) 334-0088</p>
<p>Support Portal</p>	<p>https://support.haivision.com</p>
<p>Product Information</p>	<p>info@haivision.com</p>

