

Haivision

Kraken™

Advanced Real-Time Video Transcoder
API Integrator's Guide Version 2.0

HVS-ID-INT-KRAK-200
Issue 01

Copyright

©2015 Haivision. All rights reserved.

Document Number: HVS-ID-INT-KRAK-200

Version Number: v2.0-01

This publication and the product it describes contain proprietary and confidential information. No part of this document may be copied, photocopied, reproduced, translated or reduced to any electronic or machine-readable format without prior written permission of Haivision. The information in this document is subject to change without notice. Haivision assumes no responsibility for any damages arising from the use of this document, including but not limited to, lost revenue, lost data, claims by third parties, or other damages.

If you have comments or suggestions concerning this integrator's guide, please contact:

Technical Publications Department
Haivision
4445 Garand
Montréal, Québec, H4R 2H9 Canada

Telephone: 1-514-334-5445

Toll-free (North America) 1-877-224-5445

info@haivision.com

Trademarks

The Haivision logo, Haivision, and certain other marks used herein are trademarks of Haivision. All other brand or product names identified in this document are trademarks or registered trademarks of their respective companies or organizations.

HDMI, the HDMI logo and High-Definition Multimedia Interface are trademarks or registered trademarks of HDMI Licensing LLC.

Table of Contents

About This Guide	5
About Haivision.....	6
Audience	6
Reliability of Information.....	6
Obtaining Documentation.....	6
Related Documents	7
Service Support.....	7
Document Conventions.....	7
Version 2.0 Update	9
Introduction	10
URIs for REST Resources	11
URI Structure	11
OAuth.....	12
Implementing OAuth.....	12
REST API Responses	15
Example Success Response	16
Example Error Response	17
Sorting Response Content	18
XML Entities	19
API Reference	21
Summary of Kraken API Resources	22
Syntax Conventions	23
Input Resources.....	24
Inputs API Endpoints	24
/apis/kraken/inputs	24
/apis/kraken/inputs/input- {id}	28
Output Resources.....	32
Outputs API Endpoints.....	32
/apis/kraken/outputs	32
/apis/kraken/outputs/output- {id}	36
Transcoder Resources	39
Transcoders API Endpoints.....	39

/apis/kraken/transcoders	39
/apis/kraken/transcoders/transcoder- {id}	42
Stream Resources	45
Streams API Endpoints	45
/apis/kraken/streams	45
/apis/kraken/streams/stream- {id}	48
Configuration Resources	52
Configurations API Endpoints	52
/apis/kraken/configurations	52
/apis/kraken/configurations/configuration- {id}	55
System Resources	59
System API Endpoints	59
/apis/kraken/system	59
Server Resources	60
Server API Endpoints	60
/apis/servers	60
/apis/servers/server- {id}	61
/apis/servers/server- {id} /nics	62
/apis/servers/server- {id} /nics/nic- {id}	63
 Error Codes	 64
 Example Implementation	 66
 Appendix A: Glossary of Terms	 69
 Appendix B: Warranty Information	 73

About This Guide

Welcome to the API Integrator’s Guide for Haivision’s Kraken™ Application Programming Interface (API), Version 2.0. This guide describes the API functions that can be used to interface third party management systems with the Kraken Video Transcoder.

Topics In This Section

About Haivision	6
Audience	6
Reliability of Information	6
Obtaining Documentation	6
Related Documents	7
Service Support	7
Document Conventions	7

About Haivision

Haivision is a global leader in delivering advanced video networking, digital signage, and IP video distribution solutions. Haivision offers complete end-to-end technology for video, graphics, and metadata to help customers to build, manage, and distribute their media content to users throughout an organization or across the Internet. Haivision has specific expertise in the enterprise, education, medical/healthcare, and federal/military markets.

Haivision is based in Montreal and Chicago, with technical centers in Beaverton, Oregon; Austin, Texas; and Hamburg, Germany.

Audience

This guide is directed towards qualified developers and system integrators who are familiar with XML and HTTP. Note that you can interface with the API using any programming language that supports XML and HTTP communication.

Reliability of Information

The information contained in this integrator's guide has been carefully checked and is believed to be entirely reliable. However, as Haivision improves the reliability, function, and design of its products, the possibility exists that this integrator's guide may not remain current.

If you require updated information, or any other Haivision product information, contact:

Haivision
4445 Garand
Montréal, Québec, H4R 2H9 Canada

Telephone: 1-514-334-5445
Email: info@haivision.com

Or visit our website at: <http://www.haivision.com>

Obtaining Documentation

You may download the latest software, Release Notes, Quick Start Guide, and other relevant documentation from our Download Center at:
<http://www.haivision.com/download-center/>



NOTE All customers may access the Download Center; however, a login is required. If you do not have a login, select the link to create an account.

Related Documents

In addition to this integrator's guide, the following document(s) are also available through Haivision's Download Center (see link above):

- Kraken User's Guide
- Kraken Quick Start Guide
- Kraken Software-Only Installation Guide
- Makito User's Guide
- Barracuda User's Guide
- Hai1000 (Piranha) User's Guide

Service Support

Haivision is committed to providing the service support and training needed to install, manage, and maintain your Haivision equipment.

For more information regarding service programs, training courses, or for assistance with your support requirements, contact Haivision Technical Support via our Support Portal on our website at: <http://www.haivision.com/support-portal-home>

Document Conventions

The following document conventions are used throughout this integrator's guide.



TIP The light bulb symbol 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. IMPORTANT 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, or minor to moderate injury. It may also be used to alert against unsafe practices.

Version 2.0 Update

Kraken v2.0 introduces the following new API features and enhancements.

Transcoder Resources Updates

- `streamType` has new possible value for HEVC support:

```
<videoType>hevc</videoType>
```

where `streamType` is `avc` or `hevc`. The tag is optional and the default is `avc`.

- `GET`, `POST`, and `PUT` all support a new optional tag to enable B frames (as well as “reference B frames”)

```
<bFrame>2</bFrame>
```

where the value is `-1`, `0`, `1`, `2`, `3`. The default is `-1` (i.e., “unset”, let the encoder decide).

The value `0` disables B frames and B reference frames. Values of `1–3` set a maximum number B frames to use and enable B reference frames.

For details, see [“/apis/kraken/transcoders/transcoder-`{id}`”](/apis/kraken/transcoders/transcoder-<code>{id}</code>) on page 42.

Introduction

Haivision's Kraken™ Application Programming Interface (API) provides a means for third parties to create their own products that integrate with Kraken applications.

The Kraken API is a Representational State Transfer (REST) API. REST is a style of software architecture for distributed hypermedia systems such as the World Wide Web and provides a set of rules (constraints) to which an architecture should conform.

REST is a uniform interface between components, allowing them to communicate in a standard way. Requests use the standard HTTP methods: `GET`, `POST`, `PUT`, and `DELETE`.

The effect is that your services are accessible through standard tools, and it is safe for other services and utilities to use yours in ways you did not predict.



NOTE To help keep applications stable with future versions of the API, please allow for, and ignore, unknown XML elements. When expanding the API, it may be necessary for Haivision to add elements to existing XML elements (without changing existing elements).



TIP All communication with the REST API is done through the portal server. Your base URI for requests should match the root of your portal server. In the examples, replace “https://example.haivision.com/” with the address of your portal server.

REST Informational Links

Following are some useful external references to learn more about REST:

- [Architectural Styles and the Design of Network-based Software Architectures](#) (dissertation by Roy Fielding)
- [Representational State Transfer](#) (Wikipedia entry)
- [REST in Plain English](#)
- [Explaining REST](#)
- [How to Create a REST Protocol](#)
- [REST Anti-Patterns](#)

URIs for REST Resources



NOTE “foobar” is a place holder name intended to represent whatever is being discussed.

URI Structure

Given a list of foobars, the following URI scheme provides access to the list of foobar entities and to a particular foobar:

URI	Method	Notes
/foobars	GET	This returns a collection of the foobar entities. By default items in the list are a minimal representation of a foobar entity. Note that we use the plural for the directory name.
/foobars	POST	This creates a foobar entity and returns a link to entity in the form /foobars/foobar-{id}.
/foobars/foobar-{id}	GET	This returns the full content of the foobar identified by the given id. Note that we use the singular for the entity name.
/foobars/foobar-{id}	PUT	Update the contents of a foobar entity.
/foobars/foobar-{id}	DELETE	Delete the foobar entity.

Sub-elements of a foobar entity are made available as sub-resources of /foobars/foobar-{id}, e.g.:

```
/foobars/foobar-{id}/bazs/baz-{id}/bloops/bloop-{id}
```



NOTE The ID in each URI comes from the collection preceding it. When a resource contains multiple IDs, the notation does not imply that the IDs are identical. Refer to the collection to get the ID.

OAuth

The Kraken API uses the OAuth (Open Authorization) standard for authorization when a third party application requests access. As defined in the OAuth 1.0 Protocol Abstract:

“OAuth provides a method for clients to access server resources on behalf of a resource owner (such as a different client or an end-user). It also provides a process for end-users to authorize third-party access to their server resources without sharing their credentials (typically, a username and password pair), using user-agent redirections.”

OAuth is a standardized authentication mechanism that works by signing the HTTPS request using a shared secret. The Kraken uses a “two-legged” implementation to control which applications can make use of the API. Two-legged OAuth does not provide user authentication, it only validates an application's identity.

Implementing OAuth

Implementing OAuth for the Kraken API is relatively simple and straightforward. However, it requires that both the server and client side behave the same way. Therefore, it is important to take care to avoid even minor mistakes, which can lead to authentication errors. The instructions which follow provide an overview of the signature process. Please refer the Interactive OAuth Guide listed under [“OAuth Informational Links”](#), which walks you through an interactive example of OAuth signature construction. Another great resource for understanding OAuth in action is the REST Client for Firefox, also listed below.

In both cases, leave the `Accessor Secret`, `Token` and `Token Secret` fields blank. The REST client allows you to manually enter the nonce (number used once, in this case a random string) and timestamp so you can verify that your signature is accurate. The instructions below outline how to sign a request, and store it in an HTTP header. The Kraken API also supports the signature information as part of the query string, or part of the post data, but the header is preferred.



NOTE Usage of OAuth with the Kraken API requires that calls be sent via HTTPS protocol.

OAuth Informational Links

- [Official Site](http://oauth.net/) (<http://oauth.net/> Official Site)
- [OAuth RFC](http://tools.ietf.org/html/rfc5849) (<http://tools.ietf.org/html/rfc5849> OAuth RFC (official spec))
- [Authoritative Guide to OAuth](http://hueniverse.com/oauth/guide/) (<http://hueniverse.com/oauth/guide/>)

- [Interactive OAuth Guide](http://hueniverse.com/2008/10/beginners-guide-to-oauth-part-iv-signing-requests/) (<http://hueniverse.com/2008/10/beginners-guide-to-oauth-part-iv-signing-requests/> Interactive OAuth Guide (Use “Create Your Own” to see the full details))
- [Simple OAuth Sample](http://developer.yahoo.com/blogs/ymn/posts/2010/04/a_twolegged_oauth_serverclient_example/) (http://developer.yahoo.com/blogs/ymn/posts/2010/04/a_twolegged_oauth_serverclient_example/ Simple OAuth Sample)
- [OAuth Libraries](http://oauth.net/code/) (<http://oauth.net/code/> OAuth Libraries)
- [RESTClient for Firefox that supports OAuth](https://addons.mozilla.org/en-US/firefox/addon/restclient/) (<https://addons.mozilla.org/en-US/firefox/addon/restclient/>) (Only fill in consumer key and consumer secret to authenticate)

Preparing for OAuth

Before you can use the Kraken API, you need to perform two steps from the Web Interface ([REST API](#) page). First, you must enable API access. Second, because OAuth uses a key pair authentication mechanism, you need to generate the credential (i.e., a key and secret pair). For details, please refer to the Kraken User’s Guide (Chapter 3: “Managing the Kraken”).

When you have retrieved this API credential, you can proceed to the next step.

Generating the Request Base String

The next step is to generate OAuth headers.

1. Generate OAuth parameters.
 - a. Generate a random nonce and store it as `oauth_nonce`.



NOTE This value should not be reused and should not be sequential.

- b. Generate a timestamp and store it as `oauth_timestamp`.
 - c. Set `oauth_consumer_key` to the Consumer Key retrieved from the [REST API](#) page (see [“Preparing for OAuth”](#) above).
 - d. Set `oauth_signature_method` to “HMAC-SHA1”. (No other methods are currently supported.)
2. Gather all parameters:
 - OAuth parameters
 - GET parameters
 - POST parameters
 3. Encode the parameters using UTF-8 standards/functions.
 4. Encode the parameters using URL standards/functions.

5. Normalize parameters (sort parameters alphabetically per <http://tools.ietf.org/html/rfc5849#section-3.4.1.3.2>).
6. Concatenate parameters together with an ampersand “&” between each, similar to HTTP GET requests.

Signing a Request with OAuth:

The next step is to generate the `oauth_signature` and place it in the Authorization header.

1. Generate the signature base string:

The base string is a combination of:

- Encoded HTTP request method (GET, POST, PUT, DELETE, etc.) (<http://tools.ietf.org/html/rfc5849#section-3.6>)
- Ampersand
- Base URI (e.g.: <https://example.haivision.com/apis/assets>) (Leave off the query portion)
- Ampersand
- Request base string (Construction noted above)

2. Encrypt the base string using HMAC-SHA1:

- The result should be base64 encoded.
- Store the result as `oauth_signature`.

3. Place OAuth values (`oauth_signature`, `oauth_consumer_key`, `oauth_nonce`, etc.) in the Authorization header, e.g.:

```
Authorization: OAuth
  oauth_consumer_key="",
  oauth_token="",
  oauth_nonce="",
  oauth_timestamp="0",
  oauth_signature_method="HMAC-SHA1",
  oauth_version="1.0",
  oauth_signature="mrhdcH0WE%2BD%2FPrETH%2Bn1Gw4PSXc%3D"
```



NOTE Note that the Authorization header should be contained on a single line. Newlines have been inserted in the above example for clarity.

The `oauth_token` is not required, and the `oauth_version` is always 1.0.

REST API Responses

Responses to a request consist of two elements: the HTTP status code and the response content. An application can act initially upon the HTTP status code (sensing success or failure) and then act specifically upon the data of the response content.

Response content is usually returned as application/xml data, with the root level of `<response>`.

Within the `<response>`, the content is context-specific. Individual API functions specify the type of response content later in this documentation.

If there is a problem processing or executing the request, the response content may contain an `<error>` element with a more application-specific error code.

For more details on the HTTP and `<error>` responses, see [“Error Codes”](#) on page 64.

Example Success Response

```
<?xml version="1.0" encoding="UTF-8" ?>
<response>
  <transcoders>
    <transcoder>
      <name>SD-1</name>
      <resolution>
        <width>720</width>
        <height>480</height>
      </resolution>
      <frameRate>2</frameRate>
      <videoBitrate>2000</videoBitrate>
      <gopSize>30</gopSize>
      <metadata>off</metadata>
      <audio>on</audio>
      <audioBitrate>128</audioBitrate>
      <videoType>avc</videoType>
      <bFrame>-1</bFrame>
      <link rel="self" type="application/xml" href=
        "https://10.6.60.203/apis/kraken/transcoders/SD-1" />
    </transcoder>
    <transcoder>
      <name>TR-2</name>
      <frameRate>2</frameRate>
      <videoBitrate>3000</videoBitrate>
      <gopSize>30</gopSize>
      <metadata>off</metadata>
      <audio>on</audio>
      <videoType>avc</videoType>
      <bFrame>-1</bFrame>
      <link rel="self" type="application/xml" href=
        "https://10.6.60.203/apis/kraken/transcoders/TR-2" />
    </transcoder>
    <transcoder>
      <name>HD-3</name>
      <resolution>
        <width>1920</width>
        <height>1080</height>
      </resolution>
      <videoBitrate>3000</videoBitrate>
      <gopSize>30</gopSize>
      <metadata>off</metadata>
      <audio>on</audio>
      <audioBitrate>128</audioBitrate>
      <videoType>avc</videoType>
      <bFrame>-1</bFrame>
      <link rel="self" type="application/xml" href=
        "https://10.6.60.203/apis/kraken/transcoders/HD-3" />
    </transcoder>
  </transcoders>
</response>
```


Example Error Response

```
HTTP/1.1 400 Bad Request
Content-Type: application/xml

<?xml version="1.0" encoding="ISO-8859-1"?>
<response>
  <error>
    <code>1011</code>
    <message>Input XML data is poorly formatted</message>
  </error>
</response>
```

Sorting Response Content

Please note that the contents of a given container (for example, the tags within a `<input>`, or the `<input>` elements inside a `<inputs>`) are not guaranteed to be returned in any particular order.

To avoid unnecessary errors, it should not be assumed that the order of elements will follow those in the example (for example, `<id>` tags may not be the first tag in an element.)

If any sorting needs to be done, then this should be carried out by the application.

XML Entities

Each of the API references below contains details about the specific XML entities used.



NOTE {foobarID} is used throughout these examples to denote the unique identifiers referenced within the XML data. Keep in mind that this is not the syntax used in the actual results for these elements.

In general a request for a list of resources (/apis/resources) will return XML such as:

```
<resources>
  <resource>
    <id>abc</id>
  </resource>
  <resource>
    <id>xyz</id>
  </resource>
</resources>
```

A request for a single resource (/apis/resources/resource-xyz) will return XML such as:

```
<resource>
  <id>xyz</id>
</resource>
```

Generic entities you may come across are as follows:

<error>

```
<error>
  <code>1011</code>
  <message>Input XML data is poorly formatted</message>
</error>
```

<link>

```
<link rel="self" type="application/xml" href=
      "https://example.haivision.com/apis/kraken/inputs/input-{inputID}"/>
```

- `rel`: describes the relationship of the link to the current entity. Values vary depending on context.
- `type`: Content-Type of the linked data
- `href`: REST-navigable link to the indicated entity

API Reference

This API command reference lists and describes the available resources for the Kraken Video Transcoder API.

Topics In This Chapter

Summary of Kraken API Resources	22
Syntax Conventions	23
Input Resources	24
Inputs API Endpoints	24
/apis/kraken/inputs	24
/apis/kraken/inputs/input-{id}	28
Output Resources	32
Outputs API Endpoints	32
/apis/kraken/outputs	32
/apis/kraken/outputs/output-{id}	36
Transcoder Resources	39
Transcoders API Endpoints	39
/apis/kraken/transcoders	39
/apis/kraken/transcoders/transcoder-{id}	42
Stream Resources	45
Streams API Endpoints	45
/apis/kraken/streams	45
/apis/kraken/streams/stream-{id}	48
Configuration Resources	52
Configurations API Endpoints	52
/apis/kraken/configurations	52
/apis/kraken/configurations/configuration-{id}	55
System Resources	59
System API Endpoints	59
/apis/kraken/system	59
Server Resources	60
Server API Endpoints	60
/apis/servers	60
/apis/servers/server-{id}	61
/apis/servers/server-{id}/nics	62
/apis/servers/server-{id}/nics/nic-{id}	63

Summary of Kraken API Resources

The Kraken API consists of resources divided into the following categories:

Category	Description
Input Resources	Use to define the source URL.
Output Resources	Use to define one or more output URLs.
Transcoder Resources	Use to define audio and video characteristics to change in the outbound stream.
Stream Resources	Use to select from defined Inputs, Transcoders and Outputs to set up real-time stream-based transcoding.
Configuration Resources	Use to save, load, delete, or get information about configurations in the system.
System Resources	Use to get the Kraken version number.
Server Resources	Use to get information for the Kraken physical server.

For a basic description of the XML entities referenced in these sections, see [“XML Entities”](#) on page 21.

“Endpoints” vs. “Methods”

In order to use this reference, keep in mind the following definitions:

- An **endpoint** is a URI (Uniform Resource Identifier) that points to a function or operation provided by the API, e.g., `/apis/kraken/inputs`.
- A **method**, for the purposes of this document, refers to the HTTP methods `GET`, `POST`, `PUT`, or `DELETE`. An HTTP method acts on a Kraken API endpoint.

For a glossary of terms used in this document, see [Appendix A \(page 69\)](#).

Syntax Conventions

The following syntax conventions are used in this reference:

Convention	Description
MS Sans Serif font	Indicates command names and options, filenames and code samples.
{ braces }	Indicates a placeholder for the name of a resource, e.g., ID part of a URL.
< >	Delineates an XML tag. The left angle bracket begins an XML element. The right angle bracket ends an XML element.
/	A slash before an element name ends the element definition.
/>	A slash followed by a right angle bracket ends the response element.
[]	Square brackets indicate optional elements or arguments.
x y	A vertical bar separates items in a list of options from which you must select one. If options are not separated by , you may use combinations.

Input Resources

The `inputs` API allows you to define and get information about the source URL.

Inputs API Endpoints

/apis/kraken/inputs

HTTP Method: GET

- Description: Gets the list of inputs.
- URL Parameters:
 - none
- Media Types:
 - application/xml
- Input Data:
 - none
- Return Data:
 - `<inputs>` element containing multiple `<input>` elements.
- HTTP Return Codes:
 - 200: Results found.
 - 404: No results found.
- Example Request:
`https://example.haivision.com/apis/kraken/inputs`

- Example Response:

```
<response>
  <inputs>
    <input>
      <uuid>f7615c6b-830d-4e73-a132-b315b2c29051</uuid>
      <name>China Lake</name>
      <link rel="self" type="application/xml" href=
        "https://localhost:20343/apis/kraken/inputs/input-
        f7615c6b-830d-4e73-a132-b315b2c29051" />
    </input>
    <input>
      <uuid>e2c41f15-369a-4b0d-b772-c4142defd83d</uuid>
      <name>bluray</name>
      <link rel="self" type="application/xml" href=
        "https://localhost:20343/apis/kraken/inputs/input-
        e2c41f15-369a-4b0d-b772-c4142defd83d" />
    </input>
  </inputs>
</response>
```

/apis/kraken/inputs

HTTP Method: POST

- **Description:** Creates a new `<input>` resource.
- **URL Parameters:**
 - None
- **Media Types:**
 - `application/xml`
- **Input Data:**
 - `<input>` element, with `<url>` required.
- **Return Data:**
 - `<link>` element to the newly created input.
- **HTTP Return Codes:**
 - 201: Created.
 - 400: Bad Request.
 - 500: Internal server error.

- **Example Request:**

`https://example.haivision.com/apis/kraken/inputs`

```
<input>
  <name>Another input</name>
  <url>udp://239.207.1.3:9002</url>
  <description>Another note</description>
  <streamType>MJPEGRAW</streamType>
  <interface>https://10.6.60.202/apis/servers/server-
    bc305be293d3/nics/nic-eth1</interface>
</input>
```



NOTE The `<interface>` tag is populated with a full link to the NIC object retrieved by the `GET NIC` call.

- Example Response:

```
<response>
  <link rel="self" type="application/xml" href=
    "https://10.6.60.202/apis/kraken/inputs/input-aaef0642-47ce-
      4935-8a49-ce6f2d5d1257" />
</response>
```

/apis/kraken/inputs/input-{id}

HTTP Method: GET

- Description: Retrieves information for a specific input.
- URL Parameters:
 - none
- Media Types:
 - application/xml
- Input Data:
 - none
- Return Data:
 - `<input>` element.
- HTTP Return Codes:
 - 200: Results found.
 - 404: Unknown ID.
- Example Request:

`https://example.haivision.com/apis/kraken/inputs/input-{id}`
- Example Response:

```
<response>
  <input>
    <uuid>f5684714-5ee6-4523-ab73-c61773769b53</uuid>
    <name>Makito Input 1</name>
    <url>udp://239.19.3.100:4900</url>
    <description>Makito 1 Input on Direct TV Feed.</description>
    <streamType>MJPEGRAW</streamType>
    <interface>https://10.6.60.202/apis/servers/server-
bc305be293d3/nics/nic-eth1</interface>
    <state>stopped</state>
    <link rel="self" type="application/xml" href=
"https://example.haivision.com/apis/kraken/inputs/input-
f5684714-5ee6-4523-ab73-c61773769b53" />
  </input>
</response>
```



NOTE The `<state>` is either `stopped`, `active`, or `unset` (not part of a stream).

Only fields which are set will be returned upon `GET`. That is, if `<description>` is not set, the Kraken won't return the tag. Required fields such as `<url>` for input, for example, will always be returned since they will always have a value associated with them.

/apis/kraken/inputs/input-{id}

HTTP Method: PUT

- **Description:** Edits an `<input>` resource.
- **Media Types:**
 - `application/xml`
- **Input Data:**
 - `<input>` element, with `<url>` required.
- **Return Data:**
 - `<link>` element to updated resource.
- **HTTP Return Codes:**
 - 200: Input successfully updated.
 - 400: Bad Request.
 - 404: Unknown ID
- **Example Request:**

`https://example.haivision.com/apis/kraken/inputs/input-{id}`

```
<input>
  <name>New name</name>
  <url>udp://239.207.1.4:9004</url>
  <description>New note</description>
  <streamType>MPEG2TS</streamType>
  <interface>https://10.6.60.202/apis/servers/server-
    bc305be293d3/nics/nic-eth0</interface>
</input>
```



NOTE The `<interface>` tag is populated with a full link to the NIC object retrieved by the `GET NIC` call.

- Example Response:

```
<response>
  <link rel="self" type="application/xml" href=
    "https://10.6.60.202/apis/kraken/inputs/input-aaef0642-47ce-
    4935-8a49-ce6f2d5d1257" />
</response>
```

/apis/kraken/inputs/input-`{id}`

HTTP Method: DELETE

- Description: Removes an `<input>` resource.
- Media Types:
 - application/xml
- Input Data:
 - none
- Return Data:
 - `<link>` pointing to the inputs collection.
- HTTP Return Codes:
 - 200: Input successfully deleted.
 - 400: Bad Request.
 - 404: Unknown ID
 - 500: Server error.
- Example Request:
`https://example.haivision.com/apis/kraken/inputs/input-{id}`

- Example Response:

```
<response>
  <link rel="self" type="application/xml" href=
    "https://example.haivision.com/apis/kraken/inputs" />
</response>
```

Output Resources

The `outputs` API allows you to define and get information about one or more output URLs.

Outputs API Endpoints

/apis/kraken/outputs

HTTP Method: GET

- Description: Gets the list of outputs.
- URL Parameters:
 - none
- Media Types:
 - application/xml
- Input Data:
 - none
- Return Data:
 - `<outputs>` element containing multiple `<output>` elements.
- HTTP Return Codes:
 - 200: Results found.
 - 404: No results found.
- Example Request:
`https://example.haivision.com/apis/kraken/outputs`

- Example Response:

```
<response>
  <outputs>
    <output>
      <uuid>fcc7f606-c331-427c-bff9-644f8aa62d11</uuid>
      <name>Out 1</name>
      <link rel="self" type="application/xml" href=
        "https://localhost:20343/apis/kraken/outputs/output-
        fcc7f606-c331-427c-bff9-644f8aa62d11" />
    </output>
    <output>
      <uuid>f6e11008-6a04-4172-9a85-653c081b8871</uuid>
      <name>Out 2</name>
      <link rel="self" type="application/xml" href=
        "https://localhost:20343/apis/kraken/outputs/output-
        f6e11008-6a04-4172-9a85-653c081b8871" />
    </output>
  </outputs>
</response>
```

/apis/kraken/outputs

HTTP Method: POST

- **Description:** Creates a new `<output>` resource.
- **URL Parameters:**
 - None
- **Media Types:**
 - `application/xml`
- **Input Data:**
 - `<output>` element, with `<url>` required.
- **Return Data:**
 - `<link>` element to the newly created output.
- **HTTP Return Codes:**
 - 201: Created.
 - 400: Bad Request.
 - 500: Server error.

- **Example Request:**

`https://example.haivision.com/apis/kraken/outputs`

```
<output>
  <name>Another output</name>
  <url>udp://239.202.1.3:4900</url>
  <description>A description</description>
  <mtu>1442</mtu>
  <ttl>16</ttl>
  <tos>128</tos>
  <sap>
    <transmitSap>on</transmitSap>
    <address>224.2.127.254</address>
    <port>9875</port>
    <sessionName>A name</sessionName>
    <sessionDescription>A description</sessionDescription>
    <keywords>A keyword</keywords>
    <author>An author</author>
  </sap>
  <interface>https://10.6.60.202/apis/servers/server-
    bc305be293d3/nics/nic-eth1</interface>
</output>
```



NOTE The `<interface>` tag is populated with a full link to the NIC object retrieved by the `GET NIC` call.

- Example Response:

```
<response>
<link rel="self" type="application/xml" href=
  "https://10.6.60.202/apis/kraken/outputs/output-d7948861-42b2-4731-
  9465-c1325a9b7a4d" />
</response>
```

/apis/kraken/outputs/output-**{id}**

HTTP Method: GET

- **Description:** Retrieves information for a specific output.
- **URL Parameters:**
 - none
- **Media Types:**
 - application/xml
- **Input Data:**
 - none
- **Return Data:**
 - `<output>` element.
- **HTTP Return Codes:**
 - 200: Results found.
 - 404: Unknown ID.
- **Example Request:**
`https://example.haivision.com/apis/kraken/outputs/output-{id}`
- **Example Response:**

```
<response>
  <output>
    <uuid>f5684714-5ee6-4523-ab73-c61773769b53</uuid>
    <name>Makito Output 1</name>
    <url>udp://239.1.10.29:4900</url>
    <description>Makito Output 1</description>
    <state>stopped</state>
    <mtu>1490</mtu>
    <ttl>64</ttl>
    <tos>184</tos>
    <link rel="self" type="application/xml" href=
      "https://example.haivision.com/apis/kraken/outputs/output-
        ef8da11d-af76-437b-bc2a-5b3afced8e3f" />
  </output>
</response>
```

/apis/kraken/outputs/output-`{id}`

HTTP Method: PUT

- Description: Edits an `<output>` resource.
- Media Types:
 - `application/xml`
- Input Data:
 - `<output>` element, with `<url>` required.
- Return Data:
 - `<link>` element to updated resource.
- HTTP Return Codes:
 - 200: Output successfully updated.
 - 400: Bad Request.
 - 404: Unknown ID.
 - 500: Server error.
- Example Request:
`https://example.haivision.com/apis/kraken/outputs/output-{id}`

```
<output>
  <name>New name</name>
  <url>udp://239.202.1.4:4900</url>
  <description>New description</description>
  <mtu>500</mtu>
  <ttl>24</ttl>
  <tos>156</tos>
  <sap>
    <transmitSap>on</transmitSap>
    <address>224.2.127.254</address>
    <port>9875</port>
    <sessionName>A name</sessionName>
    <sessionDescription>A description</sessionDescription>
    <keywords>A keyword</keywords>
    <author>An author</author>
  </sap>
  <interface>https://10.6.60.202/apis/servers/server-
    bc305be293d3/nics/nic-eth1</interface>
</output>
```



NOTE The `<interface>` tag is populated with a full link to the NIC object retrieved by the `GET NIC` call.

- Example Response:

```
<response>
  <link rel="self" type="application/xml" href=
    "https://10.6.60.202/apis/kraken/outputs/output-d7948861-42b2-
    4731-9465-c1325a9b7a4d" />
</response>
```

/apis/kraken/outputs/output-`{id}`

HTTP Method: DELETE

- Description: Removes an `<output>` resource.
- Media Types:
 - application/xml
- Input Data:
 - none
- Return Data:
 - `<link>` to outputs collection.
- HTTP Return Codes:
 - 200: Output successfully deleted.
 - 400: Bad Request.
 - 404: Unknown ID.
 - 500: Server error.
- Example Request:
`https://example.haivision.com/apis/kraken/outputs/output-{id}`
- Example Response:

```
<response>
  <link rel="self" type="application/xml" href=
    "https://example.haivision.com/apis/kraken/outputs" />
</response>
```

Transcoder Resources

The `transcoders` API allows you to define and get information about audio and video characteristics to change in the outbound stream.

Transcoders API Endpoints

/apis/kraken/transcoders

HTTP Method: GET

- Description: Gets the list of transcoders.
- URL Parameters:
 - none
- Media Types:
 - application/xml
- Input Data:
 - none
- Return Data:
 - `<transcoders>` element containing multiple `<transcoder>` elements.
- HTTP Return Codes:
 - 200: Results found.
 - 404: No results found.
- Example Request:
`https://example.haivision.com/apis/kraken/transcoders`
- Example Response:

```
<response>
  <transcoders>
    <transcoder>
      <uuid>f00a2077-a6ff-4335-92a9-6a7fd543e20d</uuid>
      <name>SD</name>
      <link rel="self" type="application/xml" href=
        "https://example.haivision.com/apis/kraken/transcoders/
        transcoder-f00a2077-a6ff-4335-92a9-6a7fd543e20d" />
    </transcoder>
  </transcoders>
</response>
```

/apis/kraken/transcoders

HTTP Method: POST

- **Description:** Creates a new `<transcoder>` resource.
- **URL Parameters:**
 - None
- **Media Types:**
 - `application/xml`
- **Input Data:**
 - `<transcoder>` element, with `<name>` required.
- **Return Data:**
 - `<link>` to newly created transcoder.
- **HTTP Return Codes:**
 - 201: Created.
 - 400: Bad Request.
 - 500: Server error.
- **Example Request:**

`https://example.haivision.com/apis/kraken/transcoders`

```
<transcoder>
  <name>SD-4</name>
  <resolution>
    <width>720</width>
    <height>480</height>
  </resolution>
  <frameRate>30</frameRate>
  <videoBitrate>3000</videoBitrate>
  <gopSize>30</gopSize>
  <metadata>off</metadata>
  <audio>on</audio>
  <audioBitrate>256</audioBitrate>
</transcoder>
```



NOTE As of Version 1.3, Frame rate is now an absolute value, rather than a divisor.

POST commands to `transcoder` will set any unmarked fields to their default value, which would appear as a 0 for resolution, GOP, and audio and video bitrates.

- Example Response:

```
<response>
  <link rel="self" type="application/xml" href=
    "https://example.haivision.com/apis/kraken/transcoders/transcoder-
      f00a2077-a6ff-4335-92a9-6a7fd543e20d" />
</response>
```

/apis/kraken/transcoders/transcoder-{id}

HTTP Method: GET

- **Description:** Retrieves information for a specific transcoder.
- **URL Parameters:**
 - none
- **Media Types:**
 - application/xml
- **Input Data:**
 - none
- **Return Data:**
 - <transcoder> element.
- **HTTP Return Codes:**
 - 200: Results found.
 - 404: Unknown ID.

- **Example Request:**

```
https://example.haivision.com/apis/kraken/transcoders/transcoder-{id}
```

- **Example Response:**

```
<?xml version="1.0" encoding="UTF-8" ?>
<response>
  <transcoder>
    <uuid>a4eeea19-9f7f-4f41-b2c9-2d2035615b4d</uuid>
    <name>generic</name>
    <frameRate>1</frameRate>
    <videoBitrate>3000</videoBitrate>
    <gopSize>0</gopSize>
    <metadata>off</metadata>
    <audio>on</audio>
    <audioBitrate>228</audioBitrate>
    <videoType>avc</videoType>
    <bFrame>-1</bFrame>
    <state>active</state>
    <link rel="self" type="application/xml" href=
    "https://10.65.11.198/apis/kraken/transcoders/transcoder-a4eeea19-
    9f7f-4f41-b2c9-2d2035615b4d" />
  </transcoder>
</response>
```

/apis/kraken/transcoders/transcoder-{id}

HTTP Method: PUT

- **Description:** Edits a `<transcoder>` resource.
- **Media Types:**
 - `application/xml`
- **Input Data:**
 - `<transcoder>` element, with `<name>` required.
- **Return Data:**
 - `<link>` element to updated resource.
- **HTTP Return Codes:**
 - 200: Transcoder successfully updated.
 - 400: Bad Request.
 - 404: Unknown ID.
 - 500: Server error.

- **Example Request:**

```
https://example.haivision.com/apis/kraken/transcoders/transcoder-{id}
```

```
<transcoder>
  <name>frank</name>
  <resolution>
    <width>800</width>
    <height>600</height>
  </resolution>
  <frameRate>0</frameRate>
  <videoBitrate>2000</videoBitrate>
  <gopSize>30</gopSize>
  <metadata>off</metadata>
  <audio>on</audio>
  <audioBitrate>128</audioBitrate>
  <videoType>avc</videoType>
  <bFrame>2</bFrame>
</transcoder>
```



NOTE PUT commands to `transcoder` will overwrite any fields that have not been set with their default value.

- Example Response:

```
<response>
  <link rel="self" type="application/xml" href=
    "https://example.haivision.com/apis/kraken/transcoders/
      transcoder-f00a2077-a6ff-4335-92a9-6a7fd543e20d" />
</response>
```

/apis/kraken/transcoders/transcoder-{id}

HTTP Method: DELETE

- Description: Removes a `<transcoder>` resource.
- Media Types:
 - application/xml
- Input Data:
 - none
- Return Data:
 - `<link>` to transcoders collection.
- HTTP Return Codes:
 - 200: Transcoder successfully deleted.
 - 400: Bad Request.
 - 404: Unknown ID.
 - 500: Server error.

- Example Request:

```
https://example.haivision.com/apis/kraken/transcoders/transcoder-
{id}
```

- Example Response:

```
<response>
  <link rel="self" type="application/xml" href=
    "https://example.haivision.com/apis/kraken/transcoders" />
</response>
```

Stream Resources

The `streams` API allows you to select from defined Inputs, Transcoders and Outputs to set up real-time stream-based transcoding. You can also get information about the streams in the system.

Streams API Endpoints

/apis/kraken/streams

HTTP Method: GET

- Description: Gets the list of streams.
- URL Parameters:
 - none
- Media Types:
 - application/xml
- Input Data:
 - none
- Return Data:
- `<streams>` element containing multiple `<stream>` elements.
- HTTP Return Codes:
 - 200: Results found.
 - 404: No results found.
- Example Request:
`https://example.haivision.com/apis/kraken/streams`
- Example Response:

```
<response>
  <streams>
    <stream>
      <uuid>ee0d583a-e628-4541-971f-1b5d847fec31</uuid>
      <name>Session 1</name>
      <state>active</state>
      <link rel="self" type="application/xml" href=
        "https://example.haivision.com/apis/kraken/streams/stream-
          ee0d583a-e628-4541-971f-1b5d847fec31" />
    </stream>
  </streams>
</response>
```

/apis/kraken/streams

HTTP Method: POST

- **Description:** Creates a new `<stream>` resource.
- **URL Parameters:**
 - None
- **Media Types:**
 - `application/xml`
- **Input Data:**
 - `<stream>` element, with `<name>` required.
- **Return Data:**
 - `<link>` to newly created stream
- **HTTP Return Codes:**
 - 201: Created.
 - 400: Bad Request.
 - 500: Server error.
- **Example Request:**

`https://example.haivision.com/apis/kraken/streams`

```
<stream>
  <uuid>ee0d583a-e628-4541-971f-1b5d847fec31</uuid>
  <name>Stream 124</name>
  <state>stopped</state>
  <description>Stream Description</description>
  <input>https://10.1.40.89/apis/kraken/inputs/input-c769e651-b6a5-
    4547-94d5-f06bd2877a65</input>
  <transcoder>https://10.1.40.89/apis/kraken/transcoders/transcoder-
    9ff39a54-309c-4834-aa33-2b59549d0a1d</transcoder>
  <outputs>
    <output>https://10.1.40.89/apis/kraken/outputs/output-
      a87b141c-a4d2-4b04-bb20-d709b0b26225</output>
    <output>https://10.1.40.89/apis/kraken/outputs/output-
      ded3a62b-1a57-4352-a9bf-a336a97f4e15</output>
  </outputs>
</stream>
```



NOTE If state is not supplied, it is assumed to be stopped.

- Example Response:

```
<response>
  <link rel="self" type="application/xml" href=
    "https://example.haivision.com/apis/kraken/streams/stream-
    ee0d583a-e628-4541-971f-1b5d847fec31" />
</response>
```

/apis/kraken/streams/stream-`{id}`

HTTP Method: GET

- Description: Retrieves information for a specific stream.
- URL Parameters:
 - none
- Media Types:
 - application/xml
- Input Data:
 - none
- Return Data:
- `<stream>` **element**.
- HTTP Return Codes:
 - 200: Results found.
 - 404: Unknown ID.
- Example Request:
`https://example.haivision.com/apis/kraken/streams/stream-{id}`

- Example Response:

```
<response>
  <stream>
    <uuid>d4701f54-ac37-4768-8e8f-a1000ab59f05</uuid>
    <name>Session1</name>
    <state>stopped</state>
    <description>4mbps CBR</description>
    <input>
      <name>Mako720</name>
      <link rel="input" type="application/xml" href=
        "https://10.1.40.89/apis/kraken/inputs/input-c769e651-b6a5-
        4547-94d5-f06bd2877a65" />
    </input>
    <transcoder>
      <name>Transcoder1 4mbps CBR</name>
      <link rel="transcoder" type="application/xml" href=
        "https://10.1.40.89/apis/kraken/transcoders/transcoder-
        9ff39a54-309c-4834-aa33-2b59549d0a1d" />
    </transcoder>
    <outputs>
      <output>
        <name>Output1</name>
        <link rel="output" type="application/xml" href=
          "https://10.1.40.89/apis/kraken/outputs/output-a87b141c-
          a4d2-4b04-bb20-d709b0b26225" />
      </output>
      <output>
        <name>Output2</name>
        <link rel="output" type="application/xml" href=
          "https://10.1.40.89/apis/kraken/outputs/output-ded3a62b-
          1a57-4352-a9bf-a336a97f4e15" />
      </output>
    </outputs>
    <link rel="self" type="application/xml" href=
      "https://10.1.40.89/apis/kraken/streams/stream-d4701f54-ac37-
      4768-8e8f-a1000ab59f05" />
  </stream>
</response>
```

/apis/kraken/streams/stream-{id}

HTTP Method: PUT

- **Description:** Edits a `<stream>` resource.
- **Media Types:**
 - `application/xml`
- **Input Data:**
 - `<stream>` element, with `<name>` required.
- **Return Data:**
 - `<link>` element to updated resource.
- **HTTP Return Codes:**
 - 200: Stream successfully updated.
 - 400: Bad Request.
 - 404: Unknown ID.
 - 500: Server error.

- **Example Request:**

`https://example.haivision.com/apis/kraken/streams/stream-{id}`

```
<stream>
  <uuid>ee0d583a-e628-4541-971f-1b5d847fec31</uuid>
  <name>Stream 124</name>
  <state>stopped</state>
  <description>Stream Description</description>
  <input>https://10.1.40.89/apis/kraken/inputs/input-c769e651-b6a5-4547-94d5-f06bd2877a65</input>
  <transcoder>https://10.1.40.89/apis/kraken/transcoders/transcoder-9ff39a54-309c-4834-aa33-2b59549d0a1d</transcoder>
  <outputs>
    <output>https://10.1.40.89/apis/kraken/outputs/output-a87b141c-a4d2-4b04-bb20-d709b0b26225</output>
    <output>https://10.1.40.89/apis/kraken/outputs/output-ded3a62b-1a57-4352-a9bf-a336a97f4e15</output>
  </outputs>
</stream>
```

- Example Response:

```
<response>
  <link rel="self" type="application/xml" href=
    "https://example.haivision.com/apis/kraken/streams/stream-
      ee0d583a-e628-4541-971f-1b5d847fec31" />
</response>
```

/apis/kraken/streams/stream-{id}

HTTP Method: DELETE

- Description: Removes a <stream> resource.
- Media Types:
 - application/xml
- Input Data:
 - none
- Return Data:
 - <link> to streams collection.
- HTTP Return Codes:
 - 200: Stream successfully deleted.
 - 400: Bad Request.
 - 404: Unknown ID.
 - 500: Server error.
- Example Request:
`https://example.haivision.com/apis/kraken/streams/stream-{id}`
- Example Response:

```
<response>
  <link rel="self" type="application/xml" href=
    "https://example.haivision.com/apis/kraken/streams" />
</response>
```

Configuration Resources

The `configurations` API allows you to save the current configuration, load saved configurations, set the default configuration to load on startup, or delete a saved configuration. You can also get information about the configurations in the system.

Configurations API Endpoints

/apis/kraken/configurations

HTTP Method: GET

- Description: Gets the list of configurations.
- URL Parameters:
 - none
- Media Types:
 - application/xml
- Input Data:
 - none
- Return Data:
- `<configurations>` **element containing multiple** `<configuration>` **elements.**
- HTTP Return Codes:
 - 200: Results found.
 - 404: No results found.
- Example Request:

`https://example.haivision.com/apis/kraken/configurations`

- Example Response:

```
<response>
  <configurations>
    <configuration>
      <uuid>83d0df6f-b1f9-421e-8942-06054703d265</uuid>
      <name>ConfigA</name>
      <defaultConfig>yes</defaultConfig>
      <link rel="self" type="application/xml" href=
        "https://example.haivision.com/apis/kraken/configurations/
        configuration-83d0df6f-b1f9-421e-8942-06054703d265" />
    </configuration>
    <configuration>
      <uuid>da8b2745-ba57-4b0d-bee9-be527d310961</uuid>
      <name>ConfigB</name>
      <defaultConfig>no</defaultConfig>
      <link rel="self" type="application/xml" href=
        "https://example.haivision.com/apis/kraken/configurations/
        configuration-da8b2745-ba57-4b0d-bee9-be527d310961" />
    </configuration>
  </configurations>
</response>
```

/apis/kraken/configurations

HTTP Method: POST

- **Description:** Creates a new `<configuration>` resource (i.e., saves the current configuration).
- **URL Parameters:**
 - None
- **Media Types:**
 - `application/xml`
- **Input Data:**
 - `<configuration>` element, with `<name>` required.
- **Return Data:**
 - `<link>` to newly created configuration
- **HTTP Return Codes:**
 - 201: Created.
 - 400: Bad Request.
 - 500: Server error.

- **Example Request:**

`https://example.haivision.com/apis/kraken/configurations`

```
<configuration>
  <name>ConfigA</name>
</configuration>
```

- **Example Response:**

```
<response>
  <link rel="self" type="application/xml" href=
    "https://example.haivision.com/apis/kraken/configurations/
      configuration-83d0df6f-b1f9-421e-8942-06054703d265" />
</response>
```

/apis/kraken/configurations/configuration-{id}

HTTP Method: GET

- Description: Retrieves information for a specific configuration.
- URL Parameters:
 - none
- Media Types:
 - application/xml
- Input Data:
 - none
- Return Data:
- <configuration> **element**.
- HTTP Return Codes:
 - 200: Results found.
 - 404: Unknown ID.
- Example Request:
`https://example.haivision.com/apis/kraken/configurations/configuration-{id}`
- Example Response:

```
<response>
  <configuration>
    <uuid>83d0df6f-b1f9-421e-8942-06054703d265</uuid>
    <name>ConfigA</name>
    <defaultConfig>yes</defaultConfig>
    <link rel="self" type="application/xml" href=
      "https://example.haivision.com/apis/kraken/configurations/
      configuration-83d0df6f-b1f9-421e-8942-06054703d265" />
    </configuration>
  </response>
```

/apis/kraken/configurations/configuration-`{id}`

HTTP Method: PUT

- **Description:** Edits a `<configuration>` resource.
- **Media Types:**
 - `application/xml`
- **Input Data:**
 - `<configuration>` element, with one or both of the following required:
 - `<activate>` to load the configuration, and/or
 - `<default>` to make the configuration to load upon bootup
- **Return Data:**
 - `<link>` element to updated resource.
- **HTTP Return Codes:**
 - 200: Configuration successfully updated.
 - 400: Bad Request.
 - 404: Unknown ID.
 - 500: Server error.

- Example Requests:

`https://example.haivision.com/apis/kraken/configurations/configuration-{id}`

To load the configuration:

```
<configuration>
  <activate>1</activate>
</configuration>
```

To make the configuration the default upon bootup:

```
<configuration>
  <default>1</default>
</configuration>
```

To load the configuration and make it the default upon bootup:

```
<configuration>
  <activate>1</activate>
  <default>1</default>
</configuration>
```



NOTE The `<activate>` or `<default>` status is either 1 (yes) or 0 (no).

- Example Response:

```
<response>
  <link rel="self" type="application/xml" href=
    "https://example.haivision.com/apis/kraken/configurations/
    configuration-83d0df6f-b1f9-421e-8942-06054703d265" />
</response>
```

/apis/kraken/configurations/configuration-{id}

HTTP Method: DELETE

- **Description:** Removes a `<configuration>` resource.
- **Media Types:**
 - `application/xml`
- **Input Data:**
 - `none`
- **Return Data:**
 - `<link>` to configurations collection.
- **HTTP Return Codes:**
 - 200: Configuration successfully deleted.
 - 400: Bad Request.
 - 404: Unknown ID.
 - 500: Server error.
- **Example Request:**
`https://example.haivision.com/apis/kraken/configurations/configuration-{id}`
- **Example Response:**

```
<response>
  <link rel="self" type="application/xml" href=
    "https://example.haivision.com/apis/kraken/configurations"/>
</response>
```

System Resources

The `system` API allows you to get the Kraken software version number.

System API Endpoints

/apis/kraken/system

HTTP Method: GET

- Description: Gets the version number for the Kraken.
- URL Parameters:
 - none
- Media Types:
 - application/xml
- Input Data:
 - none
- Return Data:
- `<system>` **element**.
- HTTP Return Codes:
 - 200: Results found.
 - 404: No results found.
- Example Request:
`https://example.haivision.com/apis/kraken/system`
- Example Response:

```
<response>
  <system>
    <version>1.1.0-22823</version>
  </system>
</response>
```

Server Resources

The `servers` API allows you to get information for the Kraken physical server.

Server API Endpoints

/apis/servers

HTTP Method: GET

- Example Request:

```
https://example.haivision.com/apis/servers
```

- Example Response:

```
<response>
  <servers>
    <server>
      <id>bc305be293d3</id>
      <link rel="self" type="application/xml" href=
        "https://10.6.60.202/apis/servers/server-bc305be293d3" />
    </server>
  </servers>
</response>
```

/apis/servers/server-{id}

HTTP Method: GET

- Example Request:

`https://example.haivision.com/apis/servers/server-bc305be293d3`

- Example Response:

```
<response>
  <server>
    <id>bc305be293d3</id>
    <link rel="self" type="application/xml" href=
      "https://10.6.60.202/apis/servers/server-bc305be293d3" />
    <link rel="nics" type="application/xml" href=
      "https://10.6.60.202/apis/servers/server-bc305be293d3/nics"
    />
  </server>
</response>
```

/apis/servers/server-{id}/nics

HTTP Method: GET

- Example Request:

```
GET NICS https://example.haivision.com/apis/servers/server-  
bc305be293d3/nics
```

- Example Response:

```
<response>  
  <nics>  
    <nic>  
      <id>lo</id>  
      <name>lo</name>  
      <link rel="self" type="application/xml" href=  
        "https://10.6.60.202/apis/servers/server-  
        bc305be293d3/nics/nic-lo" />  
    </nic>  
    <nic>  
      <id>eth0</id>  
      <name>eth0</name>  
      <link rel="self" type="application/xml" href=  
        "https://10.6.60.202/apis/servers/server-  
        bc305be293d3/nics/nic-eth0" />  
    </nic>  
    <nic>  
      <id>eth1</id>  
      <name>eth1</name>  
      <link rel="self" type="application/xml" href=  
        "https://10.6.60.202/apis/servers/server-  
        bc305be293d3/nics/nic-eth1" />  
    </nic>  
  </nics>  
</response>
```

/apis/servers/server-{id}/nics/nic-{id}

HTTP Method: GET

- Example Request:

```
GET NIC https://example.haivision.com/apis/servers/server-  
bc305be293d3/nics/nic-eth0
```

- Example Response:

```
<response>  
  <nic>  
    <id>eth0</id>  
    <name>eth0</name>  
    <link rel="self" type="application/xml" href=  
      "https://10.6.60.202/apis/servers/server-  
        bc305be293d3/nics/nic-eth0" />  
  </nic>  
</response>
```

Error Codes

An error condition will return a specified HTTP status code on the requested action.

If the error is processed internally by the Kraken, the returned data may also contain an `<error>` entity (see [“XML Entities”](#) on page 25).

XML <code><error></code> Code	HTTP Status Code	Error Message	Common Cause
1000	500 Bad Request	Failed to create new resource	Submitted data was not sufficient, missing required data.
1001	404 Not Found	No results found	Request completed successfully, but no results were found.
1002	404 Not Found	Unknown id	A specific resource was requested but not found (deleted or bad ID specified).
1003	500 Internal Server Error	Error executing SQL query	An internal function failed while executing the request.
1004	500 Bad Request	Failed to update resource	Tried to update a nonexistent entity (bad ID)
1005	500 Bad Request	Failed to delete resource	Tried to delete a nonexistent entity (bad ID)
1006	501 Not Implemented	Unknown API function requested	Tried to access a nonexistent / inactive API location
1007	400 Bad Request	Unknown HTTP method	Tried to use a non-standard HTTP method (other than GET, POST, PUT, or DELETE)
1008	400 Bad Request	Unrecognized URI structure	
1009	501 Not Implemented	HTTP method not implemented	Requested an action not utilized by the target API (e.g., POST on a query-only API)
1010	501 Not Implemented	Function not implemented	
1011	400 Bad Request	Input XML data is poorly formatted	XML content submitted had syntax or validation problems.

XML <error> Code	HTTP Status Code	Error Message	Common Cause
1012	500 Internal Server Error	Error while executing	
1013	400 Bad Request	Unrecognized arguments	
1014	401 Not Authorized	Not Authorized	Authentication credentials are invalid, expired, or removed. -or- Accessing API site via HTTP protocol instead of HTTPS
1015	403 Forbidden	API functions not enabled	API access is disabled in server configuration
1016	503 Service Unavailable	Service provider for this API is unavailable	A server process that supports this API call was unable to be reached. It may be down, inaccessible, or overloaded.

Example Implementation

Following is a working PHP example that uses OAuth authentication to retrieve an input list.

```
<?php
// OAuth library obtained from
    http://oauth.googlecode.com/svn/code/php/OAuth.php.
require_once('OAuth.php');

// Establish an OAuth consumer based on our admin 'credentials'
$CONSUMER_KEY = 'U5k4J0/397WeUaRNWTs+CA';1
$CONSUMER_SECRET = 'iWU2RE2GYn5G8Fcj28+zIw';2
$consumer = new OAuthConsumer($CONSUMER_KEY, $CONSUMER_SECRET, NULL);

// Setup OAuth request based our previous credentials and query
$base_feed = 'https://example.haivision.com/apis/kraken/inputs';3
$params = array();
$request = OAuthRequest::from_consumer_and_token($consumer, NULL, 'GET',
    $base_feed, $params);

// Sign the constructed OAuth request using HMAC-SHA1
$request->sign_request(new OAuthSignatureMethod_HMAC_SHA1(), $consumer, NULL);

// Make signed OAuth request to the Contacts API server
$url = $base_feed;
echo send_request($request->get_normalized_http_method(), $url, $request->to_header());

/**
 * Makes an HTTP request to the specified URL
 * @param string $http_method The HTTP method (GET, POST, PUT, DELETE)
 * @param string $url Full URL of the resource to access
 * @param string $auth_header (optional) Authorization header
 * @param string $postData (optional) POST/PUT request body
 * @return string Response body from the server
 */
function send_request($http_method, $url, $auth_header=null, $postData=null) {
    $curl = curl_init($url);
    curl_setopt($curl, CURLOPT_RETURNTRANSFER, true);
    curl_setopt($curl, CURLOPT_FAILONERROR, false);
    curl_setopt($curl, CURLOPT_SSL_VERIFYPEER, false);
    curl_setopt($curl, CURLOPT_SSL_VERIFYHOST, false);
```

```

switch($http_method) {
    case 'GET':
        if ($auth_header) {
            curl_setopt($curl, CURLOPT_HTTPHEADER, array($auth_header));
        }
        break;
    case 'POST':
        curl_setopt($curl, CURLOPT_HTTPHEADER, array('Content-Type:
application/atom+xml',
                                                    $auth_header));
        curl_setopt($curl, CURLOPT_POST, 1);
        curl_setopt($curl, CURLOPT_POSTFIELDS, $postData);
        break;
    case 'PUT':
        curl_setopt($curl, CURLOPT_HTTPHEADER, array('Content-Type:
application/atom+xml',
                                                    $auth_header));
        curl_setopt($curl, CURLOPT_CUSTOMREQUEST, $http_method);
        curl_setopt($curl, CURLOPT_POSTFIELDS, $postData);
        break;
    case 'DELETE':
        curl_setopt($curl, CURLOPT_HTTPHEADER, array($auth_header));
        curl_setopt($curl, CURLOPT_CUSTOMREQUEST, $http_method);
        break;
}
$response = curl_exec($curl);
if (!$response) {
    $response = curl_error($curl);
}
curl_close($curl);
return $response;
}

/**
 * Joins key:value pairs by inner_glue and each pair together by outer_glue
 * @param string $inner_glue The HTTP method (GET, POST, PUT, DELETE)
 * @param string $outer_glue Full URL of the resource to access
 * @param array $array Associative array of query parameters
 * @return string Urlencoded string of query parameters
 */
function implode_assoc($inner_glue, $outer_glue, $array) {
    $output = array();
    foreach($array as $key => $item) {
        $output[] = $key . $inner_glue . urlencode($item);
    }
    return implode($outer_glue, $output);
}
?>

```

1. Key generated from Web interface REST API page. See [“Preparing for OAuth”](#) on page 13.
2. Secret generated from Web interface REST API page. See [“Preparing for OAuth”](#) on page 13.

3. For the list of available URLs, see [“API Reference”](#) on page 21.

APPENDIX A: Glossary of Terms

AES	Advanced Encryption Standard
API	Application Programming Interface. For the purposes of this document, API refers to the collection of entities, operations and supporting materials provided with the Kraken API.
Audio Bitrate	The number of bits used per unit of time to represent an audio stream. Measured in kilobits per second (kbps).
AVC	Advanced Video Coding. A standard for video compression, used for the recording, compression, and distribution of high definition video.
CBR	Constant Bit Rate. The transcoder will generate a constant number of bits over a period of time.
CDN	Content Delivery Network.
CLI	Command Line Interface.
CRADA	Cooperative Research and Development Agreement.
Endpoint	A URI that points to a function or operation provided by the API, e.g., <code>/apis/demos</code> .
FEC	Forward Error Correction.
Frame Rate	The video frame rate per second.
Furnace	Haivision's IP video management server.
GOP	Group of Pictures. Kraken
HEVC	High Efficiency Video Coding. Also known as H.265 and MPEG-H Part 2. HEVC is a draft video compression standard, currently under development as a successor to H.264/MPEG-4 AVC (Advanced Video Coding).
Hi	Term use to refer to a high quality video encoding characterization of a given video input.
HLS	HTTP Live Streaming. An HTTP-based media streaming communications protocol created by Apple® Inc. as part of their QuickTime® and iPhone® software systems.

I-Frame	Intra Coded Picture, usually referred to as a reference frame. An I-Frame contains the full image of the picture (i.e., it is not a delta).
Input Presets	New set of input settings grouped under a central theme, which can be saved and recalled for later use.
JITC	Joint Interoperability Test Command.
JMIT	JITC Motion Imagery Tool.
Kraken	Haivision's real-time stream-based video transcoder.
KLV	Key Length Value. Refers to metadata packets.
Lo	Term use to refer to a low quality video encoding characterization of a given video input.
MAC Address	Media Access Control address. A unique identifier assigned to a network interface card, usually assigned by the network card manufacturer.
Macroblock	<p>An image compression component used on still images and video frames. The size of a block depends on the codec and is usually a multiple of 4. (In modern codecs such as H.264, the macroblock size is fixed at 16x16 pixels.)</p> <p>Each picture of a video – either a frame or a field – is partitioned into as many macroblocks as necessary to cover the picture area. These macroblocks serve as the basic element for operations such as spatial/temporal compression, motion compensation, and re-encoding.</p>
Method	For the purposes of this document, this refers to the HTTP methods GET, POST, PUT, or DELETE. An HTTP method acts on a Kraken API endpoint.
MPEG TS	MPEG Transport Stream.
MTU	Maximum Transmission Unit. Specifies the maximum allowed size of IP packets for the encoded or transcoded stream.
NDPP	Network Device Protection Profile.
NIC	Network Interface Card.
OAuth	Open Authorization. An open standard for authorization.
PID	Packet Identification Number.
PIN	Personal Identification Number.
PMT	Program Map Table, a collection of PIDs available in a transport stream.
Resolution	The stream output resolution, i.e., the number of lines per frame and pixels per line to be transcoded.

REST	Representational State Transfer. A style of software architecture for distributed hypermedia systems.
RTMP	Real Time Messaging Protocol. A protocol for streaming audio, video and data over the Internet, used primarily between an Adobe® Flash player and a server.
Session	New set of recording attributes grouped under a central theme, which can be saved and recalled for later use.
ST	Security Target.
SVC	Scalable Video Coding. An extension of the video compression standard H.264/MPEG-4 AVC.
ToS	Type of Service. Specifies the desired quality of service (QoS). This value will be assigned to the Type of Service field of the IP Header for the outgoing streams.
Transcode	A digital-to-digital conversion; encoded input is usually changed on output (typically to a different format or bitrate).
TTL	Time-to Live for stream packets. Specifies the number of router hops the Stream packet is allowed to travel/pass before it must be discarded.
UI	User interface.
URI	Uniform Resource Identifier. The Web naming/addressing technology that uses short strings to identify resources.
URL	Uniform Resource Locator. A specific type of URI . For the purposes of this document, URI and URL are used interchangeably.
VBR	Variable Bit Rate. VBR streams vary the amount of output data per time segment. VBR allows a higher bitrate to be allocated to the more complex segments of media streams while less space is allocated to less complex segments.
Video Bitrate	The number of bits used per unit of time to represent a video stream. Measured in kilobits per second (kbps).
Viper	Haivision's Multi-Stream Recording, Streaming & Publishing Appliance
VoD	Video On Demand. An interactive technology that allows users to select and view programming in real time or download programs and view them later.

XML Entity

An XML opening and closing tag in combination with its payload, e.g., the “demo” entity refers to:

```
<demo>  
  <id>myID</id>  
  <name>myName</name>  
  <value>myValue</value>  
</demo>
```

XML Tag

A named XML entity, e.g. <demo>.

APPENDIX B: Warranty Information

Haivision One (1) Year Limited 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 Knowledge Base <http://haivision.com/support/knowledge-base>. If the product is still not functioning properly after making use of these resources, please contact your Authorized Reseller or Haivision at <http://support.haivision.com> using the information provided in the documentation. The Authorized Reseller or Haivision 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.

Software End User License Agreement

READ BEFORE USING

THIS SOFTWARE END USER LICENSE AGREEMENT (“AGREEMENT”) IS FOR ANY OR ALL OF THE HAIVISION SOFTWARE PRODUCT(S) LICENSED, DOWNLOADED, INSTALLED AND/OR ACTIVATED BY YOU (“PRODUCT”). THE PRODUCT IS PROTECTED BY NATIONAL AND INTERNATIONAL COPYRIGHT LAWS AND TREATIES.

READ THE TERMS OF THE FOLLOWING AGREEMENT CAREFULLY. BY CLICKING THE ACCEPT BUTTON ON THIS AGREEMENT, OPENING THE SHRINKWRAP AROUND OR USING THE PRODUCT OR ANY PORTION THEREOF, OR BY USING OR DISTRIBUTING ANY VIDEO INFORMATION ENCODED BY, DECODED BY OR OTHERWISE MANIPULATED OR PASSED THROUGH THE PRODUCT, YOU CONFIRM YOUR ACCEPTANCE OF THIS AGREEMENT.

THIS AGREEMENT IS A LEGAL AGREEMENT BETWEEN YOU (A SINGLE CORPORATE ENTITY) AND HAIVISION. IF YOU DO NOT AGREE TO THESE TERMS, HAIVISION IS UNWILLING TO LICENSE THE PRODUCT TO YOU AND YOU ARE NOT AUTHORIZED TO INSTALL OR USE THE PRODUCT.

NOTWITHSTANDING SECTION 6.5 BELOW, THIS AGREEMENT ONLY GOVERNS THE PRODUCT(S) IF A SEPARATE SOFTWARE END USER LICENSE AGREEMENT HAS NOT BEEN SIGNED PRIOR TO THIS AGREEMENT FOR THE PRODUCT OR THE AGREEMENT IS NOT SUPERCEDED BY A SEPARATE SOFTWARE END USER LICENSE AGREEMENT FOR THE PRODUCT AT A LATER DATE.

1. DEFINITIONS

- 1.1. Entitlement.** The collective set of applicable documents (e.g., warranty, support and maintenance documents, data sheets, etc.) authorized by Haivision Network Video or its affiliate Haivision (collectively, “Haivision”) evidencing your obligation to pay associated fees (if any) for the license, associated Services, and the authorized scope of use of Product under this Agreement.
- 1.2. License Fee.** License Fee shall mean the consideration paid to Haivision for use of the Product. The License Fee is part or all of the price paid for the relevant Product.
- 1.3. Product.** Product shall mean the executable version of Haivision’s computer software, program or code, in object code format (specifically excluding source code), together with any related material including, but not limited to the hardware, Reference Manuals or database schemas provided for use in connection with the Product and including, without limitation, all Upgrades through the date of installation.
- 1.4. Reference Manuals.** Reference Manuals shall mean the most current version of the documentation for use in connection with the Product provided by Haivision to You.
- 1.5. Third-Party Content.** Services or materials, which are not proprietary to Haivision or may not be part of the materials of the company, entity or individual using the Product.
- 1.6. Updates.** Updates shall mean any periodic software releases, additions, fixes, and enhancements thereto, release notes for the Product and related Reference Manuals, (other than those defined elsewhere in this section as Upgrades) which have no value apart from their operation as part of the Product and which add minor new functions to the Product, but none so significant as to warrant classification as an Upgrade, which may be provided by Haivision to fix critical or non-critical problems in the Product on a scheduled, general release basis. Updates to the Product (“Version”) are denoted by number changes to the right of the decimal point for a version and revision number (for example, going from 2.0.0 to 2.1.0).

- 1.7. Upgrades.** Upgrades shall mean any modification to the Product made by Haivision, which are so significant, in Haivision's sole discretion, as to warrant their exclusion under the current license grant for the Product. Upgrades of Product are denoted by number changes to the left of the decimal point for a release number (for example, going from 2.0 to 3.0).
- 1.8. You (or Your).** The legal entity specified in the Entitlement, or for evaluation purposes, the entity performing the evaluation.

2. RIGHTS AND RESTRICTIONS

- 2.1. License to Use.** Subject to the terms and conditions set forth herein and subject to the terms of your Entitlement, Haivision hereby grants to You a non-exclusive, personal, limited and nontransferable right and license to use the Product in accordance with the terms of this Agreement. This license is granted to You and not, by implication or otherwise, to any parent, subsidiary or affiliate of Yours without Haivision's specific prior written consent. This license is for the limited use of the Product by You for the purpose of creating, managing, distributing and viewing IP Video assets. This license does not grant any license for content whatsoever. All rights not expressly granted to You by this Agreement are reserved by Haivision.
- 2.2. Restrictions.**
- (a) **Reproduction.** You shall not copy, modify, distribute, use or allow access to any of the Product, except as explicitly permitted under this Agreement and only in the quantities designated in the Entitlement. However, You have the right to make copies of the Product solely for archival purposes, but only in quantities necessary and typical for your Organization. You shall not modify, adapt, translate, export, prepare derivative works from, decompile, reverse engineer, disassemble or otherwise attempt to derive source code, hardware designs or other proprietary information from the Product or any internal data files generated by the Product, or use the Product embedded in any third party hardware or software. You shall also not use the Product in an attempt to, or in conjunction with, any device, program or service designed to circumvent technological measures employed to control access to, or the rights in other work protected by copyright laws. You shall not remove, modify, replace or obscure Haivision's copyright and patent notices, trademarks or other proprietary rights notices affixed to or contained within any Product. No right is granted hereunder for any third party who obtains access to any Product through You to use the Product to perform services for third parties. Most sublicensing arrangements are prohibited under this Agreement. However, if You are a Reseller, You are permitted to sublicense the Product to single end-users under terms and conditions similar to the provisions of this Agreement; however, You are responsible and liable pursuant to the terms and conditions of this Agreement for Your sublicensees' actions and failures to take required actions with respect to the Product.
 - (b) **Ownership.** The Product is conditionally licensed and not sold. As between the parties, Haivision and/or its licensors owns and shall retain all right, title and interest in and to all of the Product, including all copyrights, patents, trade secret rights, trademarks and other intellectual property rights therein, and nothing in this Agreement shall be deemed to transfer to You any ownership or title to the Product. You agree that you will not remove, alter or otherwise obscure any proprietary rights notices appearing in the Product. All Haivision technical data and computer software is commercial in nature and developed solely at private expense.

3. TERM AND TERMINATION

- 3.1. Term.** The license and service term are set forth in your Entitlement(s). Additionally, this Agreement may be terminated without cause by You upon thirty (30) days written notice to Haivision.

- 3.2. Termination for Breach.** Your rights under this Agreement will terminate immediately without notice from Haivision if You materially breach this Agreement or take any action in derogation of Haivision's rights to the Product. Haivision may terminate this Agreement should any Software become, or in Haivision's reasonable opinion likely to become, the subject of a claim of intellectual property infringement or trade secret misappropriation.
- 3.3. Termination for Bankruptcy.** Haivision may terminate this Agreement, effective immediately, if You file, or have filed against You, a petition for voluntary or involuntary bankruptcy or pursuant to any other insolvency law, makes or seeks to make a general assignment for the benefit of its creditors or applies for, or consents to, the appointment of a trustee, receiver or custodian for a substantial part of its property.
- 3.4. Termination; Effect; Survival.** Upon the termination of this Agreement for any reason:
- (a) All license rights granted hereunder shall terminate;
 - (b) You shall immediately pay to Haivision all amounts due and outstanding as of the date of such termination or expiration; and
 - (c) You shall return to Haivision all Product and all Haivision Reference Manuals or certify that all such Product and Reference Manuals have been destroyed. Notwithstanding any termination of this Agreement, the following provisions of this Agreement shall survive for the relevant period of time set forth therein, if any: Sections [2.2](#), [4](#), [5](#) and [6](#).

4. REPRESENTATIONS, DISCLAIMER AND LIMITATION OF LIABILITY

- 4.1. Limited Warranty.** Haivision warrants that: (i) the Product will operate substantially in accordance with the Reference Manuals provided and (ii) any media on which the Product is provided will be free of material damage and defects in materials and workmanship under normal use for a term of ninety (90) days (the "Warranty Period") after its delivery date. As Your sole and exclusive remedy for any breach of this warranty, Haivision will use its commercially reasonable efforts to correct any failure of the Product to operate substantially in accordance with the Reference Manuals which is not the result of any improper or unauthorized operation of the Product and that is timely reported by You to Haivision in writing within the Warranty Period, provided that in lieu of initiating commercially reasonable efforts to correct any such breach, Haivision may, in its absolute discretion, either: (i) replace the Product with other software or technology which substantially conforms to the Reference Manuals or (ii) refund to You a portion of the fee paid for the relevant Product, whereupon this Agreement shall terminate. This warranty shall immediately terminate if You or any third party makes or attempts to make any modification of any kind whatsoever to the Product, engages in any improper or unauthorized operation of the Product, including uses prohibited by the Entitlement or installs or uses the Product on or in connection with any hardware or software not specified in the Entitlement or product data sheets.
- 4.2. Warranty Disclaimers.** THE EXPRESS WARRANTIES SET FORTH IN SECTION [4.1](#) ABOVE IN RESPECT TO THE PRODUCT ARE IN LIEU OF ALL OTHER WARRANTIES, WHETHER EXPRESS OR IMPLIED, OR STATUTORY, REGARDING THE PRODUCT, OR ITS OPERATION, FUNCTIONALITY, PERFORMANCE, MERCHANTABILITY OR FITNESS FOR ANY PARTICULAR PURPOSE, INCLUDING ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE AND NON-INFRINGEMENT OF THIRD PARTY RIGHTS (ALL OF WHICH ARE DISCLAIMED). HAIVISION DOES NOT WARRANT THAT ANY OF THE PRODUCT(S) WILL MEET ALL OF YOUR NEEDS OR REQUIREMENTS, OR THAT THE USE OF ANY OF THE PRODUCT(S) WILL BE UNINTERRUPTED OR ERROR-FREE, OR THAT ALL ERRORS WILL BE DETECTED OR CORRECTED.
- 4.3. Liability Limitation.** IN NO EVENT SHALL HAIVISION OR ITS OFFICERS, EMPLOYEES, AGENTS, REPRESENTATIVES, OR MEMBERS, NOR ANYONE ELSE WHO HAS BEEN

INVOLVED IN THE CREATION, PRODUCTION OR DELIVERY OF THE PRODUCT, BE LIABLE TO YOU, YOUR CUSTOMERS OR TO ANY OTHER THIRD PARTY FOR CONSEQUENTIAL, INDIRECT, INCIDENTAL, PUNITIVE OR SPECIAL DAMAGES, LOST PROFITS, LOSS OF USE, INTERRUPTION OF BUSINESS OR FOR ANY DAMAGES FOR ANY BREACH OF THE TERMS OF THIS AGREEMENT OR FOR LOST OR CORRUPTED DATA ARISING FROM ANY CLAIM OR ACTION HEREUNDER, BASED ON CONTRACT, TORT OR OTHER LEGAL THEORY (INCLUDING NEGLIGENCE) AND WHETHER OR NOT SUCH PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. HAIVISION SHALL NOT BE LIABLE FOR DAMAGES FOR ANY CAUSE WHATSOEVER IN AN AMOUNT IN EXCESS OF THE FEE PAID TO HAIVISION BY YOU FOR THE RELEVANT PRODUCT.

5. INDEMNIFICATION

5.1. Indemnification by Haivision.

- (a) Haivision shall indemnify and hold You harmless against any and all actions, claims, losses, damages, liabilities, awards, costs and expenses (including reasonable attorneys' fees) ("Claims") arising out of (i) any accusation or purported violation of any third person's US and Canadian copyright, trademark, patent rights or trade secrets, proprietary information on account of Your use of the Product when used in accordance with the terms of this Agreement, or (ii) relating to or arising out of any negligence or willful misconduct on the part of Haivision or any breach by Haivision of the terms of this Agreement or any Maintenance and Support Agreement, or applicable law. You shall promptly notify Haivision in writing of any such Claim and promptly tender the control of the defense and settlement of any such Claim to Haivision. Haivision shall thereafter undertake the defense of any such Claim using counsel of its choice. You shall cooperate with Haivision, in defending or settling such Claim at the expense of Haivision; provided that Haivision shall not settle any Claim against You which would require the payment of money by You without the prior written consent of You, which consent shall not be unreasonably withheld. You shall have the right to consult and provide input into the defense with counsel of its choice at its own expense. Haivision shall not reimburse You for any expenses incurred by You without the prior written approval of Haivision, which approval shall not be unreasonably withheld.
- (b) If any Product is, or in the opinion of Haivision may become, the subject of any Claim for infringement, then Haivision may, or if it is adjudicatively determined that any of the Product infringes in the manner described above (except to the extent that any translation, modification, addition or deletion or combination by You is the sole source of such Claim), then Haivision shall, at its option, either (i) procure for You the right to continue use of the Product for the term hereof, (ii) replace or modify the Product with other suitable and reasonably equivalent products so that the Product becomes non-infringing, or (iii) terminate this Agreement and refund to You a portion of the fee paid for the relevant Product.
- (c) Haivision shall have no liability for: (i) the use of other than the then current release of the Product; (ii) the use of the Product other than as set forth in its accompanying documentation and as permitted herein; (iii) the modification of any of the Product by any party other than Haivision; or (iv) any infringement arising from the use of any Product by You after Haivision has issued a written notice to You requiring You to cease using such Product when Haivision exercises its option to terminate the License pursuant to Section 3.2 (collectively, "Exclusions"). SECTION 5.1 STATES HAIVISION'S ENTIRE OBLIGATION WITH RESPECT TO ANY CLAIM REGARDING THE INTELLECTUAL PROPERTY RIGHTS OF ANY THIRD PARTY.

- 5.2. Indemnification by You.** You shall indemnify and hold Haivision harmless against any and all Claims directly or indirectly arising out of, or in any manner whatsoever associated or connected with Your performance, purported performance or non-performance of your rights and obligations under this

Agreement, and against any and all Claims incurred by or on behalf of any of the foregoing in the investigation or defense of any and all such Claims.

6. OTHER PROVISIONS

- 6.1. Export and Other Restrictions.** This Agreement, and all Your rights and Your obligations under this Agreement, are subject to all applicable Canadian and U.S. Government laws and regulations relating to exports including, but not limited to, the U.S. Department of Commerce Export Administration Act and its associated Regulations and all administrative acts of the U.S. Government thereunder. In the event the Product or the Hardware is exported from the United States or re-exported from a foreign destination, You shall ensure that the distribution and export/re-export of the Product or the Hardware is in compliance with all laws, regulations, orders, or other restrictions of the U.S. Export Administration Act and its associated Regulations. You agree that neither you nor any of your Affiliates will export/re-export any Product, any hardware on which the Product is loaded or embedded, technical data, process, or service, directly or indirectly, to any country for which the Canadian government or United States government (or any agency thereof) requires an export license, other governmental approval, or letter of assurance, without first obtaining such license, approval or letter.
- 6.2. Content.** Your data and/or your use of the Product may not: (i) interfere in any manner with the functionality or proper working of the Product; (ii) stream any material that is copyrighted, protected by trade secret or otherwise subject to third party proprietary rights, including privacy and publicity rights, unless You are the owner of such rights or have permissions from the rightful owner to post the material; (iii) constitute, promote, facilitate or permit any illegal activities, including without limitation, activities that might be libelous or defamatory, invasive of privacy or publicity rights, abusive or otherwise malicious or harmful to any person or entity; (iv) distribute, share or facilitate unauthorized data, malware, viruses, Trojan horses, spyware, worms or other malicious or harmful distributions; or (v) otherwise violate, misappropriate or infringe the intellectual property, privacy, publicity, contractual or other proprietary rights of any third party.
- 6.3. Consent to Use Data.** You agree that Haivision may collect and use technical data and related information, including but not limited to technical information about Your device, system and application software and peripherals, that is gathered periodically to facilitate the provision of software updates, product support and other services to You (if any) related to the Product. Haivision may use this information, as long as it is in a form that does not personally identify You, to improve its products or to provide services or technologies to You.
- 6.4. Transfer and Assignment.** Haivision may assign, sublicense, or transfer this Agreement and/or any or all of its rights or obligations hereunder. You may not assign, transfer or delegate any of its rights or obligations hereunder (whether by operation of law or otherwise) without the prior written consent of Haivision. For purposes of the preceding sentence, and without limiting its generality, any merger, consolidation or reorganization involving You (regardless of whether You are a surviving or disappearing entity) will be deemed to be a transfer of rights, obligations or performance under this Agreement for which Haivision's prior written consent is not required. Any unauthorized assignment, transfer or delegation by You shall be null and void. This Agreement is binding upon and inures to the benefit of the parties hereto and their respective permitted successors and assigns.
- 6.5. Waiver and Amendment.** No modification, amendment or waiver of any provision of this Agreement shall be effective, unless in writing signed by both parties. No failure or delay by either party in exercising any right, power or remedy under this Agreement, except as specifically provided herein, shall operate as a waiver of any such right, power or remedy. Without limiting the foregoing, any additional legal terms and conditions submitted by You in any other documents, including but not limited to the Entitlement, shall be of no legal force or effect.

- 6.6. Enforcement by Third Party.** For any Product licensed by Haivision from other suppliers, the applicable supplier is a third party beneficiary of this Agreement with the right to enforce directly the obligations set forth in this Agreement against You.
- 6.7. Third Party Content.** Haivision is not responsible for examining or evaluating the data, accuracy, completeness, timeliness, validity, copyright compliance, legality, decency, quality or any other aspect of any Third Party Content. Haivision does not warrant or endorse and does not assume and will not have any liability or responsibility to You or any other person for any Third Party content. You agree that any Third Party Content may contain proprietary information and material that is protected by applicable intellectual property and other laws, including but not limited to copyright, and that you will not use such proprietary content, information or materials in any way whatsoever except for permitted uses of the Third Party Content.
- 6.8. Third Party Royalties.** Your further reuse, retransmission, rebroadcast, display or other distribution of your Third Party Content using the Product may require that you obtain a license from and / or pay royalties to the owners of certain third party audio and video formats. You are solely responsible for obtaining such licenses and paying such royalties.
- 6.9. Governing Law/Submission to Jurisdiction.** This Agreement shall be governed by and construed in accordance with the laws of the Province of Québec, Canada and the Laws of Canada applicable therein (excluding any conflict of laws rule or principle, foreign or domestic), exclusive of the U.N. Convention on the International Sale of Goods. You hereby consent to the jurisdiction of any provincial or federal court located within the Province of Quebec and waive any objection which You may have based on improper venue or forum non conveniens to the conduct of any proceeding in any such court.
- 6.10. Severability.** If any provision of this Agreement is held by a court of competent jurisdiction to be contrary to law, such provision shall be changed and interpreted so as to best accomplish the objectives of the original provision to the fullest extent allowed by law and the remaining provisions of this Agreement shall remain in full force and effect.
- 6.11. Force Majeure.** Neither party shall be liable to the other party for any failure or delay in performance to the extent that such delay or failure is caused by fire, flood, explosion, war, terrorism, embargo, government requirement, labor problems, export controls, failure of utilities, civil or military authority, act of God, act or omission of carriers or other similar causes beyond its control. If any such event of force majeure occurs, the party delayed or unable to perform shall give immediate notice to the other party, and the party affected by the other's delay or inability to perform may elect, at its sole discretion, to terminate this Agreement or resume performance once the condition ceases, with an option in the affected party to extend the period of this Agreement up to the length of time the condition endured. Unless written notice is given within 30 calendar days after the affected party is notified of the condition, the latter option shall be deemed selected. During an event of force majeure, the affected party shall exercise reasonable effort to mitigate the effect of the event of force majeure.
- 6.12. Entire Agreement.** This Agreement, together with the Entitlement and all other documents that are incorporated by reference herein, constitutes the sole and entire agreement between Haivision and You with respect to the subject matter contained herein, and supersedes all prior and contemporaneous understandings, agreements, representations and warranties, both written and oral, with respect to such subject matter.
- 6.13. Language.** The parties confirm that it is their wish that this Agreement, together with the Entitlement and any other documents relating hereto, have been and shall be drawn up in the English language only. Les parties confirment que c'est leur volonté expresse que ce contrat et tous documents y étant relative, y compris les bons de commande, le avis, le annexes, les autorisations, les pièces jointes et les amendments soient rédigés en langue anglaise seulement.

6.14. Headings Not Controlling. The headings used in this Agreement are for reference purposes only and shall not be deemed a part of this Agreement.

6.15. US Government Rights. Some Products are commercial computer software, as such, term is defined in 48 C.F.R. §2.101. Accordingly, if You, as the Licensee, is the US Government or any contractor therefor, You shall receive only those rights with respect to the Product and Reference Materials as are granted to all other end users under license, in accordance with:

- (a) 48 C.F.R. §227.7201 through 48 C.F.R. §227.7204, with respect to the Department of Defense and their contractors; or
- (b) 48 C.F.R. §12.212, with respect to all other US Government licensees and their contractors.

6.16. Notices. All notices, requests, consents, claims, demands, waivers and other communications hereunder shall be in writing and shall be deemed to have been given:

- (a) When delivered by hand (with written confirmation of receipt);
- (b) When received by the addressee if sent by a nationally recognized overnight courier (receipt requested);
- (c) On the date sent by facsimile (with confirmation of transmission) if sent during normal business hours of the recipient, and on the next business day if sent after normal business hours of the recipient; or
- (d) On the third day after the date mailed, by certified or registered mail, return receipt requested, postage prepaid. Such communications must be sent to the respective parties at the addresses set forth on the Entitlement (or to such other address as may be designated by a party from time to time in accordance with this Section **6.16**).

If you have questions, please contact Haivision Systems Inc., at 4445 Garand, Montréal, Québec, H4R 2H9 Canada or legal@haivision.com.

