AJA REST API

Control Framework and Examples

Developer Guide

Version 1.0
Published January 29, 2021

AJA VIDEO SYSTEMS

Notices

Trademarks

AJA® and Because it matters.® are registered trademarks of AJA Video Systems, Inc. for use with most AJA products. AJA™ is a trademark of AJA Video Systems, Inc. for use with recorder, router, software and camera products. Because it matters.™ is a trademark of AJA Video Systems, Inc. for use with camera products.

Corvid Ultra®, Io®, Ki Pro®, KONA®, KUMO®, ROI® and T-Tap® are registered trademarks of AJA Video Systems, Inc.

AJA Control Room™, KiStor™, Science of the Beautiful™, TruScale™, V2Analog™ and V2Digital™ are trademarks of AJA Video Systems, Inc.

All other trademarks are the property of their respective owners.

Copyright

Contacting AJA Support

When calling for support, have all information at hand prior to calling. To contact AJA for sales or support, use any of the following methods:

Telephone	+1.530.271.3190
FAX	+1.530.271.3140
Web	https://www.aja.com
Support Email	support@aja.com
Sales Email	sales@aja.com

Contents

Chapter 1 – Generic AJA REST API Control Framework

Introduction

The AJA REST API documentation introduces the generic AJA REST API Control Framework, shows how to use the REST API Developer Page, and provides examples of common commands.

Initially, this documentation is focused largely on these products:

- HELO
- Ki Pro GO
- Ki Pro Ultra
- Ki Pro Ultra Plus
- Ki Pro Ultra 12G

However, the concepts and methods apply to most AJA products that have a web user interface.

AJA's REST automation API provides a platform from which you can issue commands to an AJA device's internal web server, allowing remote systems the ability to control AJA embedded or networked devices. This is done with HTTP POST and GET requests using a Representational State Transfer (REST) interface. With this control framework, you can build integration and automation scripts, using any scripting language, allowing you to take full advantage of the device's functionality.

Alternative Method for Accessing this Content

As an alternative for your convenience, you can access all of the content in this document through AJA's GitLab repository, available through the following URL:

https://www.aja.com/support/rest-api

Each chapter is available as a separate Markdown (.md) file and can be viewed through a web browser.

Terminology and Concepts

This section introduces the following key terms and concepts used in this REST API:

- REST API Developer Page
- Parameters
- Mapped and Root Parameters
- Config
- Parameter Descriptors
- Presets

REST API Developer Page

All AJA network products expose the REST API Developer Page to assist developers in controlling the device. From there, you can access reference information and tools that will inform your integration and automation development.

This page can be viewed with a browser at the IP address of the unit followed by the path **`/rest.tmpl`**. For example:

```
http://192.168.0.2/rest.tmpl
```

For more details on how to use the **rest.tmpl** page, please see: *"Chapter 2 - How to Use the REST API Developer Page" on page 13*.

Parameters

The product stores its functional state in system data items called *parameters*. Almost all operations on the system are performed by setting parameters.

The control framework for each product has many different parameters with different semantics. Each parameter has a *descriptor* associated with it that contains metadata describing how the parameter is used (such as its type, what values it can take on, etc.).

For instance, the parameter's *descriptor* determines whether changes are saved to nonvolatile memory. If the param descriptor's **`registertype`** attribute is **`persistent`**, that param will be saved permanently. If **`registertype`** is **`ephemeral`**, the value will be reset to its default value whenever the device is rebooted.

For most parameters, any changes made are applied immediately, and the new value is automatically saved to nonvolatile memory if it is persistent.

Parameters can also be internal system data that is of no particular use to the user.

Mapped and Root Parameters

Sometimes, a parameter shown in the UI, referred to as a *root parameter*, may actually be an alias for one of several alternate parameters. For example, on HELO, when working with the parameters of the *Streaming Profile*, when you set the *Video Bit Rate* parameter from the Web UI, you are actually setting the instance of that parameter that corresponds to the currently selected *Streaming Profile*.

These *root parameters* are said to be *mapped* to other parameters.

Config

Config is the database where all the parameters are stored.

Parameter Descriptors

Each *parameter* has a *descriptor* which defines the metadata for the parameter. Parameter descriptors define everything about a parameter's behavior, including its valid values and how it might interact with other parameters (for *mapped* parameters).

Parameter descriptors define aspects of parameters such as:

- Data type (*enum, string, data, integer,* etc.)
- Name (as displayed on the Web UI)
- Valid values
- Persistence behavior

- Participation in *Preset* recalls
- *Relationships* to other parameters (if any)

Presets

The active configuration can be saved into and recalled from *Presets*.

Most *parameters* are stored in the *Presets*. The intention is for the *Presets* to represent the device state completely, although there are exceptions. For example, you most likely wouldn't want your IP Address to be changed by a Preset Recall.

Some tasks with Presets can be done through the front end web UI, such as saving, recalling, exporting and importing.

These tasks and additional tasks can be done through the REST API. See the Presets tab on the **rest.tmpl** page for more details.

REST API Interface

The REST API Interface is available at the IP address of the unit followed by the path **/config**. For example, if the IP address of the unit is **192.168.0.2**, the REST API Interface would be available at the following URL:

```
http://192.168.0.2/config
```

At this URL, commands such as getting and setting parameter values can be issued to the product using standard HTTP query strings.

Simple instructions on using this API can be found at *"Chapter 2 - How to Use the REST API Developer Page" on page 13*.

HTTP URL Command Syntax

Generally, the HTTP URL command syntax consists of the following components:

- Protocol
- Domain Name
- Path
- Query String

The protocol (**http://**) and domain name (**192.168.0.2**) come first:

```
http://192.168.0.2
```

The REST API is exposed at the path **/config**. The path is added next:

```
http://192.168.0.2/config
```

Following the path, a question mark is used as a separator between the URL path and the query string:

```
http://192.168.0.2/config?
```

The query string comes next, consisting of one or more query parameters. Each query parameter consists of a field-value pair separated by an equals sign: '**=**'.

The query string parameters supported by the AJA REST API are:

action - Possible values:

- **get**
- **set**
- **connect**
- **wait_for_config_events**

paramid - This is relevant for the get or set actions. It specifies the system parameter for the specified action.

- Example: **eParamID_VIdeoInSelect**

value - This is relevant for the set action. It specifies the value that you want the specified system parameter to take. The value provided for this parameter should conform to the range of values supported by the parameter as specified in its descriptor.

First Example: Getting the Current Value of a Specific Parameter

This first example query string is used to find out the current value of the parameter **eParamID_VideoInSelect**. To construct this URL command, two query parameters are used.

```
action=get
paramid=eParamID_VideoInSelect
```

When using multiple query parameters, use an ampersand "&" to separate them. Therefore, the complete URL command looks like this:

```
http://192.168.0.2/config?action=get&paramid=eParamID_
VideoInSelect
```

JSON Response Returned

When a GET request is issued with **action=get** or **action=set**, the AJA device's web server responds with a JSON object containing four name/value pairs.

For example:

```
{"paramid":"33619978","name":"eParamID_
VideoInSelect","value":"1","value_name":"HDMI" }
```

The name/value pairs have the following meanings:

- **"paramid":"33619978"** – a unique number identifying the parameter upon which the operation was requested.
- **"name":"eParamID_VideoInSelect"** – the developer-friendly name of the parameter upon which the operation was requested.
- **"value":"1"** – the value of the parameter is 1
- **"value_name":"HDMI"** – this is a text representation of the value. It is provided as a convenience so that you don't necessarily have to look up the corresponding "text" in the parameter's Descriptor.
  - For **enum** type parameters, this is the **text** corresponding to the **value** in the Descriptor's **enum_values** list.
  - For **integer** type params, it's just a duplicate of the **value**.

Second Example: Setting the Value of a Specific Parameter

Begin with the same syntax as described earlier using the protocol, domain name, and path, adding the question mark for the separator between the URL path and the query string:

```
http://192.168.0.2/config?
```

The query string for this example consists of three query parameters:

```
action=set
paramid=eParamID_VideoInSelect
value=0
```

The value **0** for the parameter **eParamID_VideoInSelect** corresponds to SDI.

Remember to use an ampersand "&" to separate each query parameter when using multiple query parameters.

The complete URL command would look like this:

```
http://192.168.0.2/config?action=set&paramid=eParamID_
VideoInSelect&value=0
```

Parameter Descriptor Table

The term *parameter descriptor table* refers to all the tables combined. This table can be viewed in different formats as described below.

desc.json

The file **desc.json** is a JSON structure that displays all of the parameter descriptors for the system in a machine-readable format. Access desc.json with this URL:

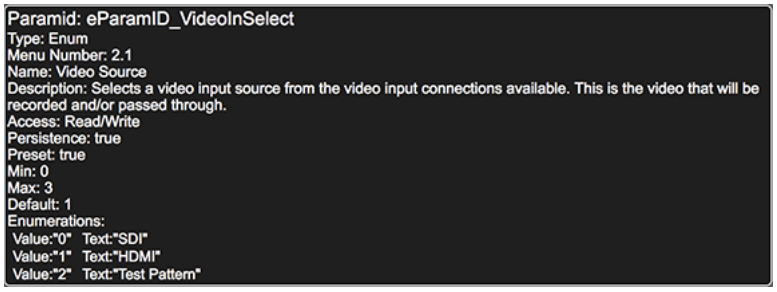
```
http://192.168.0.2/desc.json
```

descriptors.html

The file **descriptors.html** displays all of the parameter descriptors for the system in a detailed human-readable format. This page provides a filtered view of the definitive desc.json file. Access it here:

```
http://192.168.0.2/descriptors.html
```

As an example, this is how the descriptors.html page displays the parameter **`eParamID_VideoInSelect`**:



For more detailed information about parameter descriptors, please see: *"Parameter Descriptors" on page 15*.

Uploading Configuration Files

The API also supports methods to upload files for some operations such as:

- Importing presets
- Look up tables

These advanced operations are not covered by this guide.

Driving the REST API Interface

The REST API can be accessed with any modern programming language, from the command-line with tools like Telnet or cURL, or directly from a web browser.

NOTE: All examples below use the default IP Address of 192.168.0.2 for the device.

Direct Access from a Web Browser

You can *get* or *set* param values by entering a specific URL directly into the web browser's address bar.

To get a param's value:

```
http://192.168.0.2/config?action=get&paramid=eParamID_XXXX
```

To set a param's value:

```
http://192.168.0.2/config?action=set&paramid=eParamID_
XXXX&value=YYYY
```

For example, to set the HELO's *Video Input* parameter to *HDMI*, enter the following into the web browser's address bar:

```
http://192.168.0.2/config?action=set&paramid=eParamID_
VideoInSelect&value=1
```

With Telnet

```
$ telnet 192.168.0.2 80

Trying 192.168.0.2...
Connected to 192.168.0.2.
Escape character is '^]'.

GET /config?action=set&paramid=eParamID_
VideoInSelect&value=1
```

With cURL

```
$ curl "http://192.168.0.2/
config?action=set&paramid=eParamID_VideoInSelect&value=1"
```

From C++

See **`class ConfigControl`** in **`config_control.h`** for complete code. Here's a relevant snippet:

```
std::string content = "?action=set&paramid=eParamID_
VideoInSelect&value=1";
std::stringstream http_request;
http_request
    << "GET /" << _handler << content << " HTTP/1.1\r\n"
    << "Host: " << _host << "\r\n"
    << "Connection: keep-alive\r\n"
    << "\r\n";


int result = ::send(_socket, http_request.str().c_str(),
    http_request.str().length(), 0);
```

From Python

NOTE: The JSON format of the descriptors is very similar to Python syntax (for dictionaries and lists), and can, in some instances, be directly interpreted as Python data structures.

With Requests

```
>>> import requests

>>> r=requests.get('http://192.168.0.2/
config?action=set&paramid=eParamID_VideoIn Select&value=1')
```

With aja_python_api

```
>>> from aja.embedded.rest.helo import Client as
HeloClient
>>> helo = HeloClient('http://192.168.0.2')
>>> helo.setParameter('eParamID_VideoInSelect', 1)
```

Chapter 2 – How to Use the REST API Developer Page

Introduction

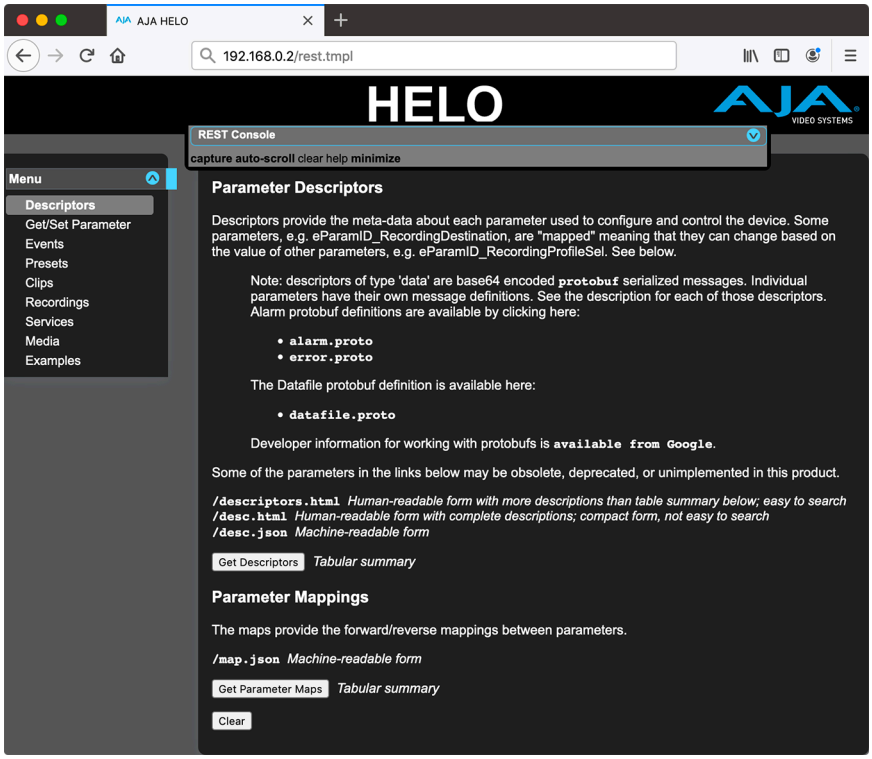
REST API Developer Page—rest.tmpl

NOTE: All examples below use the default IP Address of 192.168.0.2 for the device.

In most AJA products that use a web UI, there is a hidden REST Interface page built-in at:

```
http://192.168.0.2/rest.tmpl
```

The purpose of the REST API developer page, also referred to as **rest.tmpl**, is to show you how to use the REST API so that you can build URL strings that you will be using on an on-going basis. Once you have an understanding of how to use the REST API, you may not need to access **rest.tmpl** very frequently.



From the **rest.tmpl** page, you can make a call to the REST API with your own parameters, see the code for that call, as well as the REST API response. REST API calls can be made through a web browser, programmatically, or on the command-line with utilities such as cURL or Telnet.

The **rest.tmpl** page provides reference information such as parameter descriptors and parameter maps. From the **rest.tmpl** page, you can learn how to do the following:

- Construct your URL commands
- *get* and *set* parameter values

- Connect with the unit and wait for events and ongoing parameter changes so you can track the state of the device
- Save, recall, export, and import presets
- Retrieve a list of available network services (other REST-accessible AJA devices on the network)
- View product-specific examples of commonly used commands using both HTTP calls in a browser and by using a Telnet connection. These examples will vary product to product.

REST Console

With the REST Console, accessed through the **rest.tmpl** page, you can inspect actions taken and the device's response at the HTTP level. You can see what you have done from the **rest.tmpl** page and monitor what is going on with the device's HTTP connection.

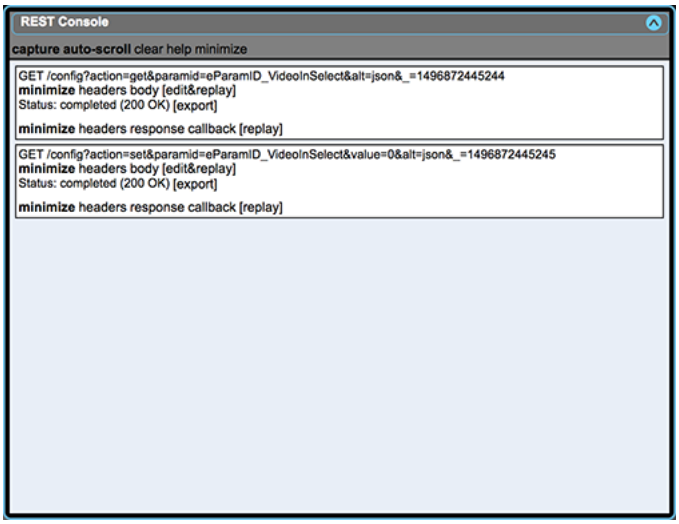
The REST Console enables you to monitor what HTTP requests get sent for every command you issue from the **rest.tmpl** page. You can only listen to the HTTP requests made from the **rest.tmpl** page.

For each HTTP interaction, you can look at the HTTP headers of the request and the response. By default, only the actual HTTP request and the response Status are shown. Other data is minimized, but can be shown by selecting one of the other display options.

Expanding and Collapsing the REST Console

Initially, the REST Console window is collapsed, as shown here:

To expand the REST Console, click on the blue circle with the down arrow located in the upper right corner of the REST Console window. Note that get and set actions are detailed in the expanded REST Console window shown here:



When expanded, the Rest Console will show all HTTP requests generated from user actions taken in the **rest.tmpl** page along with the HTTP responses from the device.

To analyze the HTTP interactions, display options are provided on the console for the request and for the response.

Click **headers** to see the response. You can also edit and replay actions.

To collapse the REST Console, click on the same blue circle (now having an up arrow) located in the upper right corner of the Console window.

Examining Headers and Responses

From the REST Console, you can analyze every HTTP request and response obtained, including the headers of both the request and the response. Examining headers and responses is especially useful.

There are 4 display options for the Request section:

- minimize
- headers
- body
- [edit&replay]

In addition, there are 5 display options for the Response section:

- minimize
- headers
- response
- callback
- [replay]

Parameter Descriptors

Parameter descriptors define the metadata for every parameter used to configure and control the device. This will vary for different AJA products and for different versions of the same product. There are four ways to view Parameter Descriptors.

1. desc.json

The file **desc.json** is a JSON structure that displays all of the parameter descriptors for the system in a machine-readable format. This is what the web UI uses, and what your scripts should use, to determine how each parameter can be utilized.

Access **desc.json** with the following URL:

```
http://192.168.0.2/desc.json
```

2. descriptors.html

The file **descriptors.html** displays all the parameter descriptors for the system in a detailed human-readable format. This page provides a filtered view of the definitive **desc.json** file. Access **descriptors.html** with the following URL:

```
http://192.168.0.2/descriptors.html
```

3. Tabular Summary View

From the Descriptors page, accessed through the Descriptors tab at **rest.tmpl**, you can click the **Get Descriptors** button. This causes a high-level summary view of the parameter descriptors to display directly on the **rest.tmpl** page. This tabular summary includes only the param ID, param name, possible values and their meaning, map type, and description.

4. As Individual JSON Data Objects

To view the complete parameter descriptor as a JSON data object for a particular ParamID, use the following syntax in the browser:

```
http://192.168.0.2/descriptors?paramid=eParamID_XXXX
```

This will return the complete JSON data object in a continuous string of text in your browser. For example, here's what the parameter **eParamID_VideoInSelect** would look like:


```
[{"param_type":"enum","descriptor_type":"enum","param_
id":"eParamID_VideoInSelect","param_name":"Video
Source","persistence_type":"persistent","register_
type":"included","factory_reset_
type":"warm","relations":{},"class_names":[],"string_
attributes": [{"n ame":"description","value":"Selects a video
input source from the video input connect ions available.
This is the video that will be recorded and/or passed
through."}, {"name":"menu_number","value":"2.1"}],"integer_
attributes":[{"name":"ParamGroup","value":2}],"enum_
values":[{"value":1,"text":"HDMI","short_text":"HDMI"},
{"value":0,"text" :"SDI","short_
text":"SDI"},{"value":2,"text":"Test Pattern","short_
text":"Test Patter n"}],"min_value":0,"max_value":3,"default_
value":1,"adjust_by":1,"scale_by":1,"offset_by":0,"display_
precision":0,"units":""}]
```

To properly format the text to make it easier to read, copy the entire result into something like JSON Pretty Print. For example, this same parameter descriptor shown through JSON Pretty Print looks like this:


```
[
  {
    "param_type": "enum",
    "descriptor_type": "enum",
    "param_id": "eParamID_VideoInSelect",
    "param_name": "Video Source",
    "persistence_type": "persistent",
    "register_type": "included",
    "factory_reset_type": "warm",
    "relations": {

    },
    "class_names": [

    ],
    "string_attributes": [
      {
        "name": "description",
        "value": "Selects a video input source from the video
                  input connections available. This is the video
                  that will be recorded and/or passed through."
      },
      {
        "name": "menu_number",
        "value": "2.1"
      }
    ],
    "integer_attributes": [
      {
        "name": "ParamGroup",
        "value": 2
      }
    ],
    "enum_values": [
      {
        "value": 1,
        "text": "HDMI",
        "short_text": "HDMI"
      },
      {
        "value": 0,
        "text": "SDI",
        "short_text": "SDI"
      },
      {
        "value": 2,
        "text": "Test Pattern",
        "short_text": "Test Pattern"
      }
    ],
    "min_value": 0,
    "max_value": 3,
    "default_value": 1,
    "adjust_by": 1,
    "scale_by": 1,
    "offset_by": 0,
    "display_precision": 0,
    "units": ""
  }
]
```

Parameter Descriptor Major Attributes

The major *attributes* defined in the Parameter Descriptors include:

- Type
- Name
- Persistence
- Values

Type

This is the data type of the parameter. Types include:

- **`enum`** – Possible values are defined by JSON objects listed in the Descriptor's **`enum_values`** attribute. To set these parameters, use the integer value of the JSON object's **`value`** key. The UI will display the value of the object's **`text`** key. (There is an exception to this in HELO. The **`ProfileSel`** parameters are defined as **`enum`** params, but the **`enum_values`** attribute is an empty list. Their possible values are defined with the **`min`** and **`max`** attributes, like an **`integer`** type.)
- **`string`** – Used for names, etc.
- **`data`** – Used for data "blobs", usually encoded as "protobuf" buffers. (See https://developers.google.com/protocol-buffers/ for more information about protocol buffers.)
- **`integer`** – Possible values are inferred from the Descriptor's **`min`** and **`max`** attributes. If the **`adjust_by`** attribute is anything other than 1, you should be careful to only set the value to multiples of the **`adjust_by`**. Also, if **`scale_by`** is not 1, **`offset_by`** is not 0, or **`display_precision`** is not 0, you will need to take extra precautions.
- **`octets`** – An integer that uses IP-address style encoding (like **`192.168.0.2`**).
- **`octets_read_only`** – Used on some products for version numbers, etc.

Name

This is the name of the parameter that is shown on the device's web UI. For example, the **`eParamID_VideoInSelect`** descriptor has this attribute:

```
"param_name":"Video Source"
```

Persistence

The descriptor attribute **`persistence_type`** determines whether a particular parameter is stored in flash (and restored after a power cycle). Possible values are:

- **`persistent`**
- **`ephemeral`**

Values

The **`values`** attribute shows the possible values that a parameter can take on. This varies depending on the parameter's **`param_type`**.

Example "param_type":"enum"

The parameter **eParamID_VideoInSelect** has **"param_type":"enum"**. The possible values are listed with the **"enum_values"** attribute:

```
"enum_values": [
  {
    "value": 1,
    "text": "HDMI",
    "short_text": "HDMI"
  },
  {
    "value": 0,
    "text": "SDI",
    "short_text": "SDI"
  },
  {
    "value": 2,
    "text": "Test Pattern",
    "short_text": "Test Pattern"
  }
]
```

Example "param_type":"integer"

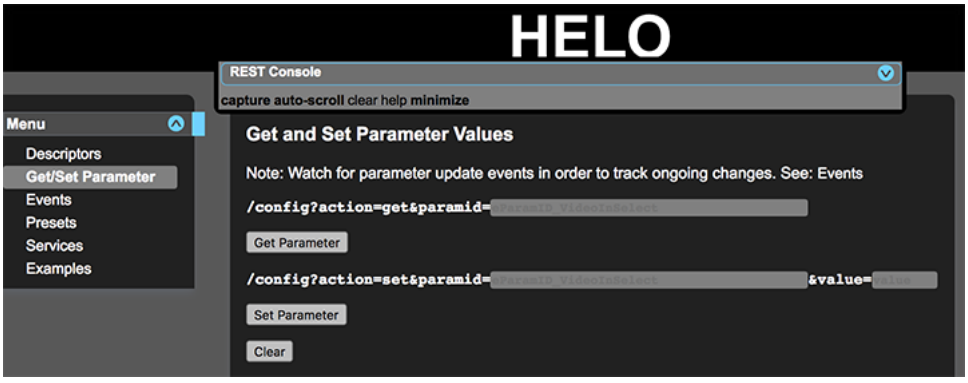
The parameter **eParamID_StreamingVideoBitrate_1** has **"param_type":"integer"**. The possible values are bounded by the **"min_value"** and **"max_value"** attributes:

```
"min_value":100,
"max_value":20000
```

Get and Set Parameters

You can learn how to *get* and *set* parameters by referencing the **Get/Set Parameter** tab on the **rest.tmpl** page.

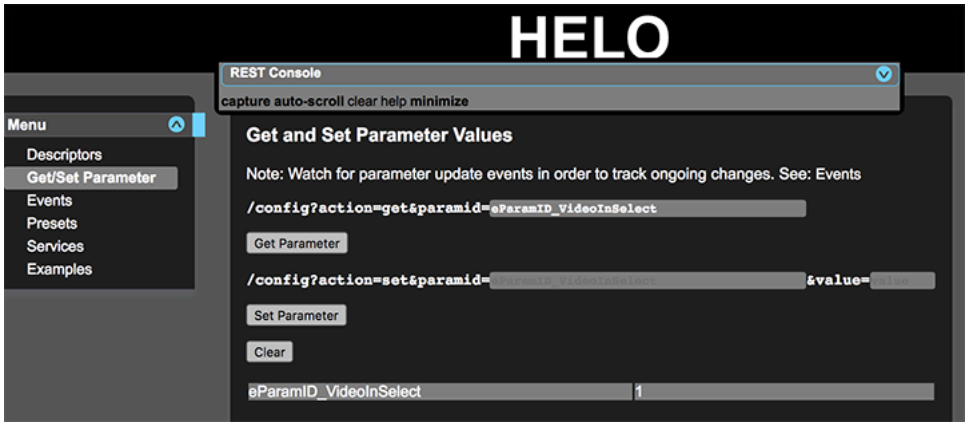
See also: *"Parameter Descriptors" on page 15* for details about how to see the entire list of parameter descriptors.



To Get a Parameter

To find out the current value of the parameter **eParamID_VideoInSelect**:

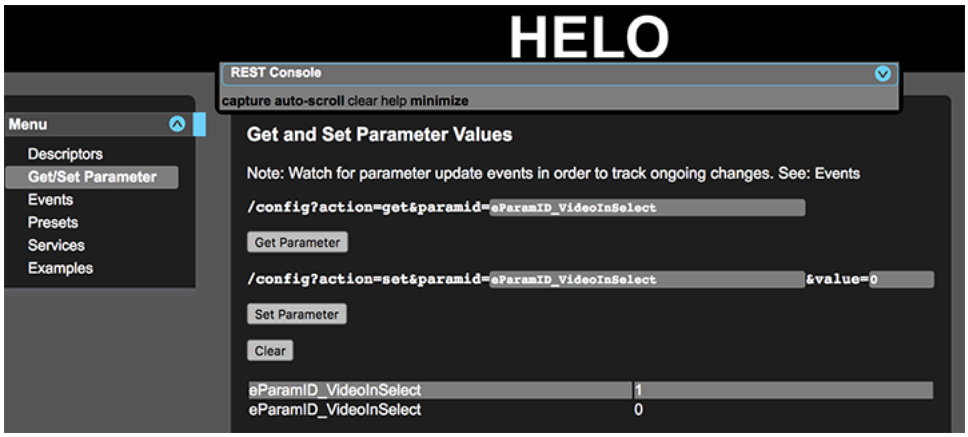
1. Enter the parameter name **eParamID_VideoInSelect** into the empty *paramid* field above the **Get Parameter** button.
2. Click **Get Parameter**. In this example, the current value "1" displays lower on the page. The value "1" corresponds to "HDMI." This means that the Video Source is currently set to HDMI.



To Set a Parameter

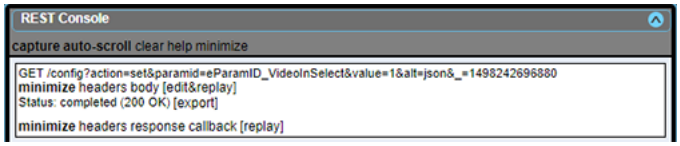
To set the value of the parameter **`eParamID_VideoInSelect`**:

1. Enter the parameter name **`eParamID_VideoInSelect`** into the empty *paramid* field above the **Set Parameter** button.
2. Enter the value you want to use into the empty *value* field. In this example, the value "0" is entered.
3. Click **Set Parameter**. The updated value displays lower on the page. The value "0" corresponds to "SDI." This means that the Video Source is now set to SDI.

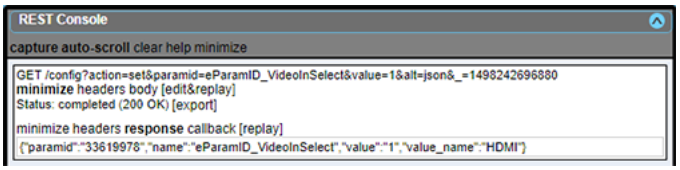


Seeing the JSON Response in the Console

To see the JSON response, expand the console.



Then click the **response** link in the footer. The JSON response displays immediately below as shown here:



See also: *"JSON Response Returned" on page 9.*

Configuring Parameters Using HTTP URLs

NOTE: All examples below use the default IP Address of 192.168.0.2 for the device.

The exact same actions shown above with the **Get Parameter** and **Set Parameter** buttons can also be done with the URL syntax shown on the **Get/Set Parameter** tab. For example, to *get* the value of the **`eParamID_VideoInSelect`** parameter with a URL command, use the following:

```
http://192.168.0.2/config?action=get&paramid=eParamID_
VideoInSelect
```

Similarly, to *set* the value of the **`eParamID_VideoInSelect`** parameter with a URL command, use the following:

```
http://192.168.0.2/config?action=set&paramid=eParamID_
VideoInSelect&value=0
```

See also: *"HTTP URL Command Syntax" on page 8* for more information.

Parameter Mappings and Root Parameters

Mappings determine which parameters are relevant at any given time, based on the values of other parameters. Root (or alias) parameters are those that are associated with the specific Recording Profile or Streaming Profile that is currently selected.

Events

Provided as a developer aid, *Events* offers a way for you to subscribe to real-time updates whenever a parameter value changes.

Discovering a Parameter Name

If you are writing a script but you don't know which specific parameter is involved with a certain kind of change, you can find out through using Events.

For example, if you want to know which parameter needs to be modified in order to start streaming on your HELO, start event polling. Then, from the front end web UI, start streaming. The resulting parameter change will be shown in Events polling.

Presets

A *Preset* is a saved collection of values for all of the parameters that are defined as *register member* (those whose Descriptors have the **`"register_type"`** attribute defined as **`"included"`**).

Most of the parameters in the system are **`"included"`**. Some basic configuration parameters are excluded and are not affected by Preset recalls. Examples of parameters that are **`"excluded"`** are:

- **eParamID_IPAddress**
- **eParamID_DHCPState**

The **Presets** page, accessed through the Presets tab at **rest.tmpl**, demonstrates how to work with Presets through the REST API.

Saving Presets

You can save Presets using the parameter **eParamID_RegisterSave**. Select a numerical value from 1 to 20 to place at the end of the URL string. Here is an example of the URL syntax:

```
http://192.168.0.2/config?action=set&paramid=eParamID_
RegisterSave&value=4
```

Recalling Presets

Recall a Preset using the parameter **eParamID_RegisterRecall**. Specify a numerical value from 1 to 20 at the end of the URL string. Here is an example of the URL syntax:

```
http://192.168.0.2/config?action=set&paramid=eParamID_
RegisterRecall&value=2
```

Services

Available Network Services are other REST-accessible AJA devices available on a network. To view available network services in a human-readable format, click the **Get Network Services** button (accessed from the **Services** tab at **rest.tmpl**). A list of detected REST-accessible AJA devices will display directly on the Services **rest.tmpl** page.

Examples

As a reference for developers, the **Examples** tab of the **rest.tmpl** page provides product-specific examples of commonly used commands using both HTTP calls in a browser and by using a Telnet connection. These examples will vary product to product.

Topics include the following:

- Get and Set Video Input Selection
- Start and Stop Recording and Streaming
- Get and Set Recording Name

Chapter 3 – HELO Commands

To access this chapter online, please see:

https://gitlab.aja.com/pub/rest_api/-/blob/master/HELO/03_HELO_Commands.md

Recording and Streaming Commands

The HELO subsystem, which controls video delivery, is known as the Replicator. You can control Recording and Streaming independently with the **`ReplicatorCommand`** parameter. Here is its descriptor:

eParamID_ReplicatorCommand Parameter Descriptor


```
[
  {
    "param_type": "enum",
    "descriptor_type": "enum",
    "param_id": "eParamID_ReplicatorCommand",
    "param_name": "Replicator Command",
    "persistence_type": "ephemeral",
    "register_type": "excluded",
    "factory_reset_type": "never",
    "relations": {

    },
    "class_names": [
      "volatile",
      "leave_enabled_during_transport"
    ],
    "string_attributes": [
      {
        "name": "description",
        "value": "Transport command"
      },
      {
        "name": "menu_number",
        "value": "125.1"
      }
    ],
    "integer_attributes": [

    ],
    "enum_values": [
      {
        "value": 0,
        "text": "No Command",
        "short_text": "No Command"
      },
      {
        "value": 1,
        "text": "Record Command",
        "short_text": "Record Command"
      },
      {
        "value": 2,
        "text": "Stop Recording Command",
        "short_text": "Stop Recording Command"
      },
```

```
        {
          "value": 3,
          "text": "Stream Command",
          "short_text": "Stream Command"
        },
        {
          "value": 4,
          "text": "Stop Streaming Command",
          "short_text": "Stop Streaming Command"
        },
        {
          "value": 5,
          "text": "Shutdown",
          "short_text": "Shutdown"
        }
      ],
      "min_value": 0,
      "max_value": 5,
      "default_value": 0,
      "adjust_by": 1,
      "scale_by": 1,
      "offset_by": 0,
      "display_precision": 0,
      "units": ""
    }
]
```


Start/Stop Recording

To start recording, set the **eParamID_ReplicatorCommand** to the value of **1**.

```
http://192.168.0.2/config?action=set&paramid=eParamID_
ReplicatorCommand&value=1
```

To stop recording, set the **eParamID_ReplicatorCommand** to the value of **2**.

```
http://192.168.0.2/config?action=set&paramid=eParamID_
ReplicatorCommand&value=2
```

Start/Stop Streaming

To start streaming, set the **eParamID_ReplicatorCommand** to the value of **3**.

```
http://192.168.0.2/config?action=set&paramid=eParamID_
ReplicatorCommand&value=3
```

To stop streaming, set the **eParamID_ReplicatorCommand** to the value of **4**.

```
http://192.168.0.2/config?action=set&paramid=eParamID_
ReplicatorCommand&value=4
```

Selecting Profiles

You can select Recording and Streaming Profiles independently with the **eParamID_RecordingProfileSel** and **eParamID_StreamingProfileSel** parameters.

eParamID_RecordingProfileSel Parameter Descriptor


```
[
  {
    "param_type": "enum",
    "descriptor_type": "enum",
    "param_id": "eParamID_RecordingProfileSel",
    "param_name": "Recording Profile",
    "persistence_type": "persistent",
    "register_type": "included",
    "factory_reset_type": "warm",
    "relations": {

    },
    "class_names": [

    ],
    "string_attributes": [
      {
        "name": "description",
        "value": "Selects a recording profile."
      },
      {
        "name": "menu_number",
        "value": "86.1"
      }
    ],
    "integer_attributes": [
      {
        "name": "ParamGroup",
        "value": 5
      }
    ],
    "enum_values": [

    ],
    "min_value": 0,
    "max_value": 9,
    "default_value": 0,
    "adjust_by": 1,
    "scale_by": 1,
    "offset_by": 0,
    "display_precision": 0,
    "units": ""
  }
]
```


To Select a Recording Profile

Set the **eParamID_RecordingProfileSel** to a numeric value with a minimum value of **0** and a maximum value of **9**.

```
http://192.168.0.2/config?action=set&paramid=eParamID_
RecordingProfileSel&value=X
```

eParamID_StreamingProfileSel Parameter Descriptor


```
[
  {
    "param_type": "enum",
    "descriptor_type": "enum",
    "param_id": "eParamID_StreamingProfileSel",
    "param_name": "Streaming Profile",
    "persistence_type": "persistent",
    "register_type": "included",
    "factory_reset_type": "warm",
    "relations": {

    },
    "class_names": [

    ],
    "string_attributes": [
      {
        "name": "description",
        "value": "Selects a streaming profile."
      },
      {
        "name": "menu_number",
        "value": "87.1"
      }
    ],
    "integer_attributes": [
      {
        "name": "ParamGroup",
        "value": 6
      }
    ],
    "enum_values": [

    ],
    "min_value": 0,
    "max_value": 9,
    "default_value": 0,
    "adjust_by": 1,
    "scale_by": 1,
    "offset_by": 0,
    "display_precision": 0,
    "units": ""
  }
]
```


To Select a Streaming Profile

Set the **eParamID_StreamingProfileSel** to a numeric value with a minimum value of **0** and a maximum value of **9**.

```
http://192.168.0.2/config?action=set&paramid=eParamID_
StreamingProfileSel&value=X
```

Setting the Recording Name

You can assign a base name for each recorded file.

To Set the Recording Name

To change the filename prefix from the default of "clip" to "Biology Lecture," for example, use the following command:

```
http://192.168.0.2/config?action=set&paramid=eParamID_
FilenamePrefix&value=Biology%20Lecture
```

Considerations about URLs and Recording Names

- URLs may not contain spaces. If you wish to set a value that contains spaces, all of the spaces need to be replaced by "+" or "%20".
- Similarily, URLs may not contain other characters with special meanings. Characters like ?, &, %, /, \, : and others have special meanings in URLs and must be encoded.
- URLs may not contain non-ASCII characters.
- To set a value that contains chacters with special meanings or non-ASCII characters (including but not limited to kanji, katakana and hirigana, cyrillic, arabic, hebrew, and latin characters with diacritical marks such as ñ, à, ç, é, ê, í, ó, ö, ...), encode the special and non-ASCII characters using "URL Encoding". URL Encoding is also known as "Percent Encoding". This web page will perform the encoding for you:
  - https://www.w3schools.com/tags/ref_urlencode.asp

Checking the Status of Recording and Streaming

The current operating state of the Replicator's Recording engine and Streaming engine are available through the parameter descriptors **eParamID_ReplicatorRecordState** and **eParamID_ReplicatorStreamState**. For example:

```
http://192.168.0.2/config?action=get&paramid=eParamID_
ReplicatorRecordState
```

The response should look something like the following:

```
{"paramid":"2097225226","name":"eParamID_
ReplicatorRecordState","value":"1","value_name":"eRRSIdle"}
```

This shows the record state value is **1**, which corresponds to the name **eRRSIdle** — Replicator Record State Idle.

eParamID_ReplicatorRecordState Parameter Descriptor


```
[
  {
    "param_type": "enum",
    "descriptor_type": "enum",
    "param_id": "eParamID_ReplicatorRecordState",
    "param_name": "Replicator Record State",
    "persistence_type": "ephemeral",
    "register_type": "excluded",
    "factory_reset_type": "never",
    "relations": {

    },
    "class_names": [
      "readonly",
      "panel_hidden"
    ],
    "string_attributes": [
      {
        "name": "description",
        "value": "Current replicator record state."
      },
      {
        "name": "menu_number",
        "value": "125.1"
      }
    ],
    "integer_attributes": [

    ],
    "enum_values": [
      {
        "value": 0,
        "text": "eRRSUninitialized",
        "short_text": "eRRSUninitialized"
      },
      {
        "value": 1,
        "text": "eRRSIdle",
        "short_text": "eRRSIdle"
      },
      {
        "value": 2,
        "text": "eRRSRecording",
        "short_text": "eRRSRecording"
      },
      {
        "value": 3,
        "text": "eRRSFailingInIdle",
        "short_text": "eRRSFailingInIdle"
      },
      {
        "value": 4,
        "text": "eRRSFailingInRecord",
        "short_text": "eRRSFailingInRecord"
      },
      {
        "value": 5,
        "text": "eRRSShutdown",
        "short_text": "eRRSShutdown"
      }
    ],
    "min_value": 0,
    "max_value": 5,
```

```
    "default_value": 0,
    "adjust_by": 1,
    "scale_by": 1,
    "offset_by": 0,
    "display_precision": 0,
    "units": ""
  }
]
```


Recalling Presets

You can recall a specific preset by setting the parameter **eParamID_RegisterRecall** to the specific numeric value you want.

To Recall Preset #2

For example, to recall Preset #2, make the following HTTP call in a browser:

```
http://192.168.0.2/config?action=set&paramid=eParamID_
RegisterRecall&value=2
```

Wait for a few moments. Then you can check whether the recall completed successfully with this HTTP call:

```
http://192.168.0.2/config?action=set&paramid=eParamID_
RegisterRecallResult
```

The resulting HTTP response should look something like:


```
{"paramid":"1560412160","name":"eParamID_
RegisterRecallResult","value":"2","value_name":"Complete"}
```


This shows the result value is **2**, which corresponds to the name **Complete**.

Dealing with Authentication

You can disable or enable authentication with the **`eParamID_Authentication`** parameter.

eParamID_Authentication Parameter Descriptor


```
[
  {
    "param_type": "enum",
    "descriptor_type": "enum",
    "param_id": "eParamID_Authentication",
    "param_name": "User Authentication",
    "persistence_type": "persistent",
    "register_type": "excluded",
    "factory_reset_type": "cold",
    "relations": {},
    "class_names": [
      "volatile"
    ],
    "string_attributes": [
      {
        "name": "description",
        "value": "Enables or disables password protection for the
                  web interface for this system."
      },
      {
        "name": "menu_number",
        "value": "50.9"
      }
    ],
    "integer_attributes": [
      {
        "name": "ParamGroup",
        "value": 12
      }
    ],
    "enum_values": [
      {
        "value": 0,
        "text": "Disabled",
        "short_text": "Disabled"
      },
      {
        "value": 1,
        "text": "Login",
        "short_text": "Login"
      }
    ],
    "min_value": 0,
    "max_value": 1,
    "default_value": 0,
    "adjust_by": 1,
    "scale_by": 1,
    "offset_by": 0,
    "display_precision": 0,
    "units": ""
  }
]
```

To Disable Authentication

If you want to disable authentication, use the following HTTP call in a browser:

```
http://192.168.0.2/config?action=set&paramid=eParamID_
Authentication&value=0
```

To Enable Authentication

If you want to enable authentication, use the following HTTP call in a browser:

```
http://192.168.0.2/config?action=set&paramid=eParamID_
Authentication&value=1
```

When User Authentication is disabled—that is, when **`eParamID_Authentication`** is set to the value of **`0`**—REST accesses can be completed without worrying about passwords or authentication. However, when User Authentication is enabled (**`eParamID_Authentication`** is set to the value of **`1`**) then REST accesses to HELO as described elsewhere will fail with an error that looks like this:

```
<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
<html><head><title>404 Not Found</title></head><body>
The document requested was not found on this server.
</body></html>
```

Making REST Commands to HELO when Authentication is Enabled

In order for REST commands to HELO to succeed when User Authentication is enabled, use the following steps:

1. Send a request to authenticate to HELO which contains the password using POST.
2. Save the cookie that the web server sends back in the authentication response.
3. Provide the cookie in each request from then on.

Example: Get Record State with Authentication Enabled

Via cURL with a password of "honk":

1. Authenticate with password via HTTP POST.

```
$ curl -d "password_provided=honk" -X POST
"http://192.168.0.2/authenticator/login" -i
HTTP/1.1 200 OK
Server: Seminole/2.71 (Linux; DF7)
Date: Mon, 16 Sep 2019 19:13:11 GMT
Connection: keep-alive
Cache-Control: no-cache
Content-Type: text/html
Set-Cookie: session=0000000e-5Qn9uwpiYiMpCxePZekIwjeREBsBA
6k; path=/
Transfer-encoding: chunked

{"login":"success"}
```

2. Save the cookie returned by the web server. The cookie sent back in the response above is:

```
session=0000000e-5Qn9uwpiYiMpCxePZekIwjeREBsBA6k; path=/
```

3. Return the cookie with each REST request. In this example, the cookie is simply cut and pasted onto the command line. In production it would typically be stored in a file or variable.

```
$ curl --cookie "session=0000000e-5Qn9uwpiYiMpC
xePZekIwjeREBsBA6k; path=/" "192.168.0.2/config?a
ction=get&paramid=eParamID_ReplicatorRecordState"
{"paramid":"2097225226","name":"eParamID_
ReplicatorRecordState","value":"3","value_name":"eRRSFa
ilingInIdle"}
```

Repeat step 3 as desired.

Downloading Clips

You can download video clips that are in .mov format from your HELO device using the cURL command line utility.

Before you can download clips, the HELO device first needs to be in Data Transfer mode (Data – LAN mode), which can be done with the **eParamID_MediaState** parameter. This parameter defines whether the HELO device is in normal Record-Stream mode or Data Transfer mode. If a transfer is in progress when this parameter is set back to Record-Stream mode, the transfer will be aborted.

Once you have completed downloading clips, return the value of the parameter **eParamID_MediaState** back to **"0"** so that recording and streaming can be performed again.

eParamID_MediaState


```
[
  {
    "param_type": "enum",
    "descriptor_type": "enum",
    "param_id": "eParamID_MediaState",
    "param_name": "MediaState",
    "persistence_type": "ephemeral",
    "register_type": "included",
    "factory_reset_type": "warm",
    "relations": {},
    "class_names": [
        "webapp_hidden"
    ],
    "string_attributes": [
        {
          "name": "description",
          "value": " Define whether the HELO device is in normal
                     Record-Stream or Data Transfer mode."
        },
        {
          "name": "menu_number",
          "value": "12.1"
        }
    ],
    "integer_attributes": [
        {
          "name": "ParamGroup",
          "value": 3
        }
    ],
    "enum_values": [
        {
          "value": 0,
          "text": "Record-Stream",
          "short_text": "Record-Stream"
        },
        {
          "value": 1,
          "text": "Data-LAN",
          "short_text": "Data-LAN"
        }
    ],
    "min_value": 0,
    "max_value": 3,
    "default_value": 0,
    "adjust_by": 1,
    "scale_by": 1,
    "offset_by": 0,
    "display_precision": 0,
    "units": ""
  }
]
```


Data - LAN mode Command

To put the HELO device into Data - LAN mode, set the **eParamID_MediaState** parameter to the value of "**1**".

```
http://192.168.0.2/config?action=set&paramid=eParamID_
MediaState&value=1
```

Record - Stream mode Command

To put the HELO device back into Record - Stream mode so that recording and streaming can be performed again, set the **eParamID_MediaState** parameter to the value of "**0**".

```
http://192.168.0.2/config?action=set&paramid=eParamID_
MediaState&value=0
```

Download Command

To download a .mov file from a HELO device, use the cURL command line utility with a terminal prompt.

In the following command example, **--output** instructs the server to create a new file rather than sending the download to your terminal window.

example-filename-downloaded.mov refers to the file name of the file after it has been downloaded. You can use any file name you like. It doesn't need to match the file name of the file you are downloading from the HELO device. **example-filename-source.mov** refers to the file name of the .mov file that is residing on your HELO device.

```
curl -v0 --output example-filename-downloaded.mov
http://192.168.0.2/media0/example-filename-source.mov
```

Substitute the example IP address **192.168.0.2** with your actual HELO device URL address.

Here is an example of the HTTP download request that AJA's browser-based javascript download initiates:

```
GET /media0/example-filename-source.mov HTTP/1.1
Host: 192.168.0.2
Connection: keep-alive
```

This will initiate the download of a single file/clip from the currently selected media.

Here is an example of the response that the client will get:

```
HTTP/1.0 200 OK
Content-Type: video/quicktime
Content-disposition: attachment; filename=example-
filename-source.mov
Content-Length: 359354368
```

Chapter 4 – Common Ki Pro GO Commands

To access this chapter online, please see:

https://gitlab.aja.com/pub/rest_api/-/blob/master/KiPro-GO/04_Ki-Pro-GO_Commands.md

Universal Transport: Start, Stop, Record, Fast Forward

The Ki Pro GO subsystem which controls video transport is known as Transport. You can control Universal Transport commands such as Start, Stop, Record, and Fast Forward independently with the **`TransportCommand`** parameter.

eParamID_TransportCommand Parameter Descriptor


```
[
  {
    "param_type": "enum",
    "descriptor_type": "enum",
    "param_id": "eParamID_TransportCommand",
    "param_name": "Transport Command",
    "persistence_type": "ephemeral",
    "register_type": "excluded",
    "factory_reset_type": "never",
    "relations": {},
    "class_names": [
      "volatile"
    ],
    "string_attributes": [
      {
        "name": "description",
        "value": "Transport command."
      }
    ],
    "integer_attributes": [],
    "enum_values": [
      {
        "value": 0,
        "text": "No Command",
        "short_text": "No Command"
      },
      {
        "value": 1,
        "text": "Play Command",
        "short_text": "Play Command"
      },
      {
        "value": 3,
        "text": "Record Command",
        "short_text": "Record Command"
      },
      {
        "value": 4,
        "text": "Stop Command",
        "short_text": "Stop Command"
      },
```

```
{
  "value": 5,
  "text": "Fast Forward",
  "short_text": "Fast Forward"
},
{
  "value": 6,
  "text": "Fast Reverse",
  "short_text": "Fast Reverse"
},
{
  "value": 7,
  "text": "Single Step Forward",
  "short_text": "Single Step Forward"
},
{
  "value": 8,
  "text": "Single Step Reverse",
  "short_text": "Single Step Reverse"
},
{
  "value": 9,
  "text": "Next Clip",
  "short_text": "Next Clip"
},
{
  "value": 10,
  "text": "Previous Clip",
  "short_text": "Previous Clip"
},
{
  "value": 11,
  "text": "Variable Speed Play",
  "short_text": "Variable Speed Play"
},
{
  "value": 12,
  "text": "Preroll",
  "short_text": "Preroll"
},
{
  "value": 13,
  "text": "Assemble Edit",
  "short_text": "Assemble Edit"
},
{
  "value": 14,
  "text": "Cue",
  "short_text": "Cue"
},
{
  "value": 15,
  "text": "Shutdown",
  "short_text": "Shutdown"
},
{
  "value": 16,
  "text": "Play At System Time",
  "short_text": "Play At System Time"
},
```

```
      {
        "value": 17,
        "text": "Record At System Time",
        "short_text": "Record At System Time"
      },
      {
        "value": 18,
        "text": "Go To Idle",
        "short_text": "Go To Idle"
      }
    ],
    "min_value": 0,
    "max_value": 18,
    "default_value": 0,
    "adjust_by": 1,
    "scale_by": 1,
    "offset_by": 0,
    "display_precision": 0,
    "units": ""
  }
]
```


Play Command

To initiate the play command, set the **eParamID_TransportCommand** to the value of "1".

```
http://192.168.0.2/config?action=set&paramid=eParamID_
TransportCommand&value=1
```

Stop Command (once to pause, twice to stop)

The first time the stop command is made, the unit will pause. The second time the stop command is made, the unit will stop (idle).

To initiate the stop command, set the **eParamID_TransportCommand** to the value of "4".

```
http://192.168.0.2/config?action=set&paramid=eParamID_
TransportCommand&value=4
```

Record Command

To start recording, set the **eParamID_TransportCommand** to the value of "3".

```
http://192.168.0.2/config?action=set&paramid=eParamID_
TransportCommand&value=3
```

Fast Forward Command

To fast forward, set the **eParamID_TransportCommand** to the value of "5".

```
http://192.168.0.2/config?action=set&paramid=eParamID_
TransportCommand&value=5
```

Selecting the Headphone Audio Channel

You can select the Encode Channel that will feed the headphones and VU Meters with the **eParamID_HeadphoneMonitorChannel** parameter.

eParamID_HeadphoneMonitorChannel Parameter Descriptor


```
[
  {
    "param_type": "enum",
    "descriptor_type": "enum",
    "param_id": "eParamID_HeadphoneMonitorChannel",
    "param_name": "Headphone Audio",
    "persistence_type": "persistent",
    "register_type": "included",
    "factory_reset_type": "warm",
    "relations": {},
    "class_names": [],
    "string_attributes": [
      {
        "name": "description",
        "value": "Selects the Encode Channel that will feed the
                  headphones and VU Meters"
      },
      {
        "name": "menu_number",
        "value": "6.2"
      }
    ],
    "integer_attributes": [
      {
        "name": "ParamGroup",
        "value": 2
      }
    ],
    "enum_values": [
      {
        "value": 0,
        "text": "Channel 1",
        "short_text": "Channel 1"
      },
      {
        "value": 1,
        "text": "Channel 2",
        "short_text": "Channel 2"
      },
      {
        "value": 2,
        "text": "Channel 3",
        "short_text": "Channel 3"
      },
      {
        "value": 3,
        "text": "Channel 4",
        "short_text": "Channel 4"
      }
    ],
    "min_value": 0,
    "max_value": 3,
    "default_value": 0,
    "adjust_by": 1,
    "scale_by": 1,
    "offset_by": 0,
    "display_precision": 0,
    "units": ""
  }
]
```

To Select the Headphone Channel

To select Channel 1, set the **eParamID_HeadphoneMonitorChannel** to the value of "0".

```
http://192.168.0.2/config?action=set&paramid=eParamID_
HeadphoneMonitorChannel&value=0
```

To select Channel 2, set the **eParamID_HeadphoneMonitorChannel** to the value of "1".

```
http://192.168.0.2/config?action=set&paramid=eParamID_
HeadphoneMonitorChannel&value=1
```

To select Channel 3, set the **eParamID_HeadphoneMonitorChannel** to the value of "2".

```
http://192.168.0.2/config?action=set&paramid=eParamID_
HeadphoneMonitorChannel&value=2
```

To select Channel 4, set the **eParamID_HeadphoneMonitorChannel** to the value of "3".

```
http://192.168.0.2/config?action=set&paramid=eParamID_
HeadphoneMonitorChannel&value=3
```

Selecting the HDMI Monitor Channel

You can select which channel to display on the HDMI monitor output with the **eParamID_HDMIMonitorChannel** parameter.

eParamID_HDMIMonitorChannel Parameter Descriptor


```
[
  {
    "param_type": "enum",
    "descriptor_type": "enum",
    "param_id": "eParamID_HDMIMonitorChannel",
    "param_name": "HDMI Monitor Channel",
    "persistence_type": "persistent",
    "register_type": "included",
    "factory_reset_type": "warm",
    "relations": {},
    "class_names": [],
    "string_attributes": [
      {
        "name": "description",
        "value": "Which channel to display on the HDMI monitor
                  output. All Channels will display all video on
                  a quad split"
      },
      {
        "name": "menu_number",
        "value": "6.1"
      }
    ],
    "integer_attributes": [
      {
        "name": "ParamGroup",
        "value": 2
      }
    ],
```

```
    "enum_values": [
      {
        "value": 0,
        "text": "Channel 1",
        "short_text": "Channel 1"
      },
      {
        "value": 1,
        "text": "Channel 2",
        "short_text": "Channel 2"
      },
      {
        "value": 2,
        "text": "Channel 3",
        "short_text": "Channel 3"
      },
      {
        "value": 3,
        "text": "Channel 4",
        "short_text": "Channel 4"
      },
      {
        "value": 4,
        "text": "All Channels",
        "short_text": "All Channels"
      }
    ],
    "min_value": 0,
    "max_value": 4,
    "default_value": 4,
    "adjust_by": 1,
    "scale_by": 1,
    "offset_by": 0,
    "display_precision": 0,
    "units": ""
  }
]
```


To Select the Channel to Display on the HDMI Monitor Output

To select Channel 1, set the **eParamID_HDMIMonitorChannel** to the value of "0".

```
http://192.168.0.2/config?action=set&paramid=eParamID_
HDMIMonitorChannel&value=0
```

To select Channel 2, set the **eParamID_HDMIMonitorChannel** to the value of "1".

```
http://192.168.0.2/config?action=set&paramid=eParamID_
HDMIMonitorChannel&value=1
```

To select Channel 3, set the **eParamID_HDMIMonitorChannel** to the value of "2".

```
http://192.168.0.2/config?action=set&paramid=eParamID_
HDMIMonitorChannel&value=2
```

To select Channel 4, set the **eParamID_HDMIMonitorChannel** to the value of "3".

```
http://192.168.0.2/config?action=set&paramid=eParamID_
HDMIMonitorChannel&value=3
```

To select all channels, set the **eParamID_HDMIMonitorChannel** to the value of "4".

```
http://192.168.0.2/config?action=set&paramid=eParamID_
HDMIMonitorChannel&value=4
```

Selecting the SDI Monitor Channel

You can select which channel to display on the SDI monitor output with the **eParamID_SDIMonitorChannel** parameter.

eParamID_SDIMonitorChannel Parameter Descriptor


```
[
  {
    "param_type": "enum",
    "descriptor_type": "enum",
    "param_id": "eParamID_SDIMonitorChannel",
    "param_name": "SDI Monitor Channel",
    "persistence_type": "persistent",
    "register_type": "included",
    "factory_reset_type": "warm",
    "relations": {},
    "class_names": [],
    "string_attributes": [
      {
        "name": "description",
        "value": "Which channel to display on the SDI monitor
                  output. All Channels will display all video on
                  a quad split"
      },
      {
        "name": "menu_number",
        "value": "6.0"
      }
    ],
    "integer_attributes": [
      {
        "name": "ParamGroup",
        "value": 2
      }
    ],
    "enum_values": [
      {
        "value": 0,
        "text": "Channel 1",
        "short_text": "Channel 1"
      },
      {
        "value": 1,
        "text": "Channel 2",
        "short_text": "Channel 2"
      },
      {
        "value": 2,
        "text": "Channel 3",
        "short_text": "Channel 3"
      },
```

```
        {
          "value": 3,
          "text": "Channel 4",
          "short_text": "Channel 4"
        },
        {
          "value": 4,
          "text": "All Channels",
          "short_text": "All Channels"
        }
      ],
      "min_value": 0,
      "max_value": 4,
      "default_value": 4,
      "adjust_by": 1,
      "scale_by": 1,
      "offset_by": 0,
      "display_precision": 0,
      "units": ""
    }
]
```


To Select the Channel to Display on the SDI Monitor Output

To select Channel 1, set the **eParamID_SDIMonitorChannel** to the value of "0".

```
http://192.168.0.2/config?action=set&paramid=eParamID_
SDIMonitorChannel&value=0
```

To select Channel 2, set the **eParamID_SDIMonitorChannel** to the value of "1".

```
http://192.168.0.2/config?action=set&paramid=eParamID_
SDIMonitorChannel&value=1
```

To select Channel 3, set the **eParamID_SDIMonitorChannel** to the value of "2".

```
http://192.168.0.2/config?action=set&paramid=eParamID_
SDIMonitorChannel&value=2
```

To select Channel 4, set the **eParamID_SDIMonitorChannel** to the value of "3".

```
http://192.168.0.2/config?action=set&paramid=eParamID_
SDIMonitorChannel&value=3
```

To select all channels, set the **eParamID_SDIMonitorChannel** to the value of "4".

```
http://192.168.0.2/config?action=set&paramid=eParamID_
SDIMonitorChannel&value=4
```

Selecting Video Input 1

You can select the input used for video input Channel 1 record/Idle with the **eParamID_VideoInput_1** parameter.

eParamID_VideoInput_1 Parameter Descriptor


```
[
  {
    "param_type": "enum",
    "descriptor_type": "enum",
    "param_id": "eParamID_VideoInput_1",
    "param_name": "Video Input 1",
    "persistence_type": "persistent",
    "register_type": "included",
    "factory_reset_type": "warm",
    "relations": {},
    "class_names": [],
    "string_attributes": [
      {
        "name": "description",
        "value": "Select the input used for Ch1 record/Idle"
      },
      {
        "name": "menu_number",
        "value": "1.1"
      }
    ],
    "integer_attributes": [
      {
        "name": "ParamGroup",
        "value": 2
      }
    ],
    "enum_values": [
      {
        "value": 0,
        "text": "SDI 1",
        "short_text": "SDI 1"
      },
      {
        "value": 1,
        "text": "SDI 2",
        "short_text": "SDI 2"
      },
      {
        "value": 2,
        "text": "SDI 3",
        "short_text": "SDI 3"
      },
      {
        "value": 3,
        "text": "SDI 4",
        "short_text": "SDI 4"
      },
      {
        "value": 4,
        "text": "HDMI 1",
        "short_text": "HDMI 1"
      },
      {
        "value": 5,
        "text": "HDMI 2",
        "short_text": "HDMI 2"
      },
      {
        "value": 6,
        "text": "HDMI 3",
        "short_text": "HDMI 3"
      },
```

```
        {
          "value": 7,
          "text": "HDMI 4",
          "short_text": "HDMI 4"
        }
      ],
      "min_value": 0,
      "max_value": 7,
      "default_value": 0,
      "adjust_by": 1,
      "scale_by": 1,
      "offset_by": 0,
      "display_precision": 0,
      "units": ""
    }
]
```


To Select the input used for video input Channel 1 record/ Idle

To select SDI Channel 1, set the **ePARAMID_VIDEOINPUT_1** to the value of "0".

```
http://192.168.0.2/config?action=set&paramid=eParamID_
VideoInput_1&value=0
```

To select SDI Channel 2, set the **ePARAMID_VIDEOINPUT_1** to the value of "1".

```
http://192.168.0.2/config?action=set&paramid=eParamID_
VideoInput_1&value=1
```

To select SDI Channel 3, set the **ePARAMID_VIDEOINPUT_1** to the value of "2".

```
http://192.168.0.2/config?action=set&paramid=eParamID_
VideoInput_1&value=2
```

To select SDI Channel 4, set the **ePARAMID_VIDEOINPUT_1** to the value of "3".

```
http://192.168.0.2/config?action=set&paramid=eParamID_
VideoInput_1&value=3
```

To select HDMI Channel 1, set the **ePARAMID_VIDEOINPUT_1** to the value of "4".

```
http://192.168.0.2/config?action=set&paramid=eParamID_
VideoInput_1&value=4
```

To select HDMI Channel 2, set the **ePARAMID_VIDEOINPUT_1** to the value of "5".

```
http://192.168.0.2/config?action=set&paramid=eParamID_
VideoInput_1&value=5
```

To select HDMI Channel 3, set the **ePARAMID_VIDEOINPUT_1** to the value of "6".

```
http://192.168.0.2/config?action=set&paramid=eParamID_
VideoInput_1&value=6
```

To select HDMI Channel 4, set the **ePARAMID_VIDEOINPUT_1** to the value of "7".

```
http://192.168.0.2/config?action=set&paramid=eParamID_
VideoInput_1&value=7
```

Selecting Audio Input 1

You can select the audio input used for audio input Channel 1 with the **ePARAMID_AUDIOINPUT_1** parameter, using either embedded audio from the channel's input video or analog audio.

eParamID_VideoInput_1 Parameter Descriptor


```
[
  {
    "param_type": "enum",
    "descriptor_type": "enum",
    "param_id": "eParamID_AudioInput_1",
    "param_name": "Audio Input 1",
    "persistence_type": "persistent",
    "register_type": "included",
    "factory_reset_type": "warm",
    "relations": {},
    "class_names": [],
    "string_attributes": [
      {
        "name": "description",
        "value": "Set audio using embedded audio from channel's
                  input video or use analog"
      },
      {
        "name": "menu_number",
        "value": "1.2"
      }
    ],
    "integer_attributes": [
      {
        "name": "ParamGroup",
        "value": 2
      }
    ],
    "enum_values": [
      {
        "value": 0,
        "text": "Follows Video",
        "short_text": "Follows Video"
      },
      {
        "value": 1,
        "text": "Analog",
        "short_text": "Analog"
      }
    ],
    "min_value": 0,
    "max_value": 1,
    "default_value": 0,
    "adjust_by": 1,
    "scale_by": 1,
    "offset_by": 0,
    "display_precision": 0,
    "units": ""
  }
]
```


To Select the audio source used for audio input Channel 1

To select the embedded audio from the channel's input video, set the **eParamID_AudioInput_1** to the value of "0".

```
http://192.168.0.2/config?action=set&paramid=eParamID_
AudioInput_1&value=0
```

To select analog audio, set the **eParamID_AudioInput_1** to the value of "1".

```
http://192.168.0.2/config?action=set&paramid=eParamID_
AudioInput_1&value=1
```

Downloading Clips

You can download video clips that are in .mov format from your Ki Pro GO device using the cURL command line utility.

Before you can download clips, the Ki Pro GO device first needs to be in Data Transfer mode (Data – LAN mode), which can be done with the **`eParamID_MediaState`** parameter. This parameter defines whether the Ki Pro GO device is in normal Record-Play mode or Data Transfer mode. If a transfer is in progress when this parameter is set back to Record-Play mode, the transfer will be aborted.

eParamID_MediaState


```
[
  {
     "param_type": "enum",
     "descriptor_type": "enum",
     "param_id": "eParamID_MediaState",
     "param_name": "Media State",
     "persistence_type": "ephemeral",
     "register_type": "included",
     "factory_reset_type": "warm",
     "relations": {},
     "class_names": [],
     "string_attributes": [
        {
           "name": "description",
           "value": "Define whether the Ki Pro device is in
                     normal Record-Play or Data Transfer mode."
        },
        {
           "name": "menu_number",
           "value": "12.1"
        }
     ],
     "integer_attributes": [
        {
           "name": "ParamGroup",
           "value": 3
        }
     ],
     "enum_values": [
        {
           "value": 0,
           "text": "Record-Play",
           "short_text": "Record-Play"
        },
        {
           "value": 1,
           "text": "Data - LAN",
           "short_text": "Data - LAN"
        }
     ],
     "min_value": 0,
     "max_value": 3,
     "default_value": 0,
     "adjust_by": 1,
     "scale_by": 1,
     "offset_by": 0,
     "display_precision": 0,
     "units": ""
  }
]
```

Data - LAN mode Command

To put the Ki Pro devide into Data - LAN mode, set the **eParamID_MediaState** parameter to the value of "1".

```
http://192.168.0.2/config?action=set&paramid=eParamID_
MediaState&value=1
```

eParamID_StoragePath

The **eParamID_StoragePath** stores the path to the location on the filesystem to record or play clips (for example, '/mnt/S1/AJA').

NOTE: The path '/media' can be used instead of '/mnt/S1/AJA' because '/media' points to the currently active storage path.


```
[
  {
     "param_type": "string",
     "descriptor_type": "string",
     "param_id": "eParamID_StoragePath",
     "param_name": "Storage Path",
     "persistence_type": "ephemeral",
     "register_type": "excluded",
     "factory_reset_type": "never",
     "relations": {},
     "class_names": [
        "readonly"
     ],
     "string_attributes": [
        {
           "name": "description",
           "value": "Path to location on filesystem to record or
                     play clips."
        },
        {
           "name": "menu_number",
           "value": "123.1"
        }
     ],
     "integer_attributes": [],
     "min_length": 0,
     "max_length": 20,
     "default_value": ""
  }
]
```


To retrieve the current storage path for your Ki Pro device, use this **get** command:

```
http://192.168.0.2/config?action=get&paramid=eParamID_
StoragePath
```

An example of the REST API response to the above command is:


```
{"paramid":"2063663370","name":"eParamID_
StoragePath","value":"/mnt/S1/AJA","value_name":""}
```


Download Command

To download a .mov file from a Ki Pro device, use the cURL command line utility with a terminal prompt.

In the following command example, **--output** instructs the server to create a new file rather than sending the download to your terminal window.

example-filename-downloaded.mov refers to the file name of the file after it has been downloaded. You can use any file name you like. It doesn't need to match the file name of the file you are downloading from the Ki Pro device. **example-filename-source.mov** refers to the file name of the .mov file that is residing on your Ki Pro device.

```
curl -v0 --output example-filename-downloaded.mov http://
192.168.0.2/media/example-filename-source.mov
```

Substitute the example IP address **192.168.0.2** with your actual Ki Pro device URL address.

Here is an example of the HTTP download request that AJA's browser-based javascript download initiates:

```
GET /media/example-filename-source.mov HTTP/1.1
Host: 192.168.0.2
Connection: keep-alive
```

This will initiate the download of a single file/clip from the currently selected media.

Here is an example of the response that the client will get:

```
HTTP/1.0 200 OK
Content-Type: video/quicktime
Content-disposition: attachment; filename=example-
filename-source.mov
Content-Length: 359354368
```

Chapter 5 – Common Ki Pro Ultra Commands

To access this chapter online, please see:

https://gitlab.aja.com/pub/rest_api/-/blob/master/KiProUltra/05_Ki-Pro-Ultra_Commands.md

Universal Transport: Start, Stop, Record, Fast Forward

The Ki Pro Ultra subsystem that controls video transport is known as Transport. You can control Universal Transport commands such as Start, Stop, Record, and Fast Forward independently with the TransportCommand parameter.

eParamID_TransportCommand Parameter Descriptor


```
[
  {
    "param_type": "enum",
    "descriptor_type": "enum",
    "param_id": "eParamID_TransportCommand",
    "param_name": "Transport Command",
    "persistence_type": "ephemeral",
    "register_type": "excluded",
    "factory_reset_type": "never",
    "relations": {},
    "class_names": [
      "volatile"
    ],
    "string_attributes": [
      {
        "name": "description",
        "value": "Transport command."
      },
      {
      "name": "menu_number",
      "value": "125.1"
      }
   ],
    "integer_attributes": [],
    "enum_values": [
      {
        "value": 0,
        "text": "No Command",
        "short_text": "No Command"
      },
      {
        "value": 1,
        "text": "Play Command",
        "short_text": "Play Command"
      },
      {
        "value": 3,
        "text": "Record Command",
        "short_text": "Record Command"
      },
      {
        "value": 4,
        "text": "Stop Command",
        "short_text": "Stop Command"
      },
```

```
{
  "value": 5,
  "text": "Fast Forward",
  "short_text": "Fast Forward"
},
{
  "value": 6,
  "text": "Fast Reverse",
  "short_text": "Fast Reverse"
},
{
  "value": 7,
  "text": "Single Step Forward",
  "short_text": "Single Step Forward"
},
{
  "value": 8,
  "text": "Single Step Reverse",
  "short_text": "Single Step Reverse"
},
{
  "value": 9,
  "text": "Next Clip",
  "short_text": "Next Clip"
},
{
  "value": 10,
  "text": "Previous Clip",
  "short_text": "Previous Clip"
},
{
  "value": 11,
  "text": "Variable Speed Play",
  "short_text": "Variable Speed Play"
},
{
  "value": 12,
  "text": "Preroll",
  "short_text": "Preroll"
},
{
  "value": 13,
  "text": "Assemble Edit",
  "short_text": "Assemble Edit"
},
{
  "value": 14,
  "text": "Cue",
  "short_text": "Cue"
},
{
  "value": 15,
  "text": "Shutdown",
  "short_text": "Shutdown"
},
{
  "value": 16,
  "text": "Play At System Time",
  "short_text": "Play At System Time"
},
```

```
        {
          "value": 17,
          "text": "Record At System Time",
          "short_text": "Record At System Time"
        },
        {
          "value": 18,
          "text": "Go To Idle",
          "short_text": "Go To Idle"
        }
      ],
      "min_value": 0,
      "max_value": 18,
      "default_value": 0,
      "adjust_by": 1,
      "scale_by": 1,
      "offset_by": 0,
      "display_precision": 0,
      "units": ""
    }
]
```


Play Command

To initiate the play command, set the **eParamID_TransportCommand** to the value of "1".

```
http://192.168.0.2/config?action=set&paramid=eParamID_
TransportCommand&value=1
```

Stop Command (once to pause, twice to stop)

The first time the stop command is made, the unit will pause. The second time the stop command is made, the unit will stop (idle).

To initiate the stop command, set the **eParamID_TransportCommand** to the value of "4".

```
http://192.168.0.2/config?action=set&paramid=eParamID_
TransportCommand&value=4
```

Record Command

To start recording, set the **eParamID_TransportCommand** to the value of "3".

```
http://192.168.0.2/config?action=set&paramid=eParamID_
TransportCommand&value=3
```

Fast Forward Command

To fast forward, set the **eParamID_TransportCommand** to the value of "5".

```
http://192.168.0.2/config?action=set&paramid=eParamID_
TransportCommand&value=5
```

Setting Playback Speed (v3.1.1 and above)

The Ki Pro Ultra subsystem that controls video transport speed is known as TransportRequestedSpeed.

You can control the speed of playing forward and playing in reverse with the **TransportRequestedSpeed** parameter. Set the parameter value to whatever forward or reverse speed you need by using a floating-point value representing your desired playback speed. The value can be any positive or negative floating-point value. For example:

- 1 = normal playback speed
- 16 = 16x playback speed
- -1.5 = 1.5x reverse speed
- 0 = pause
- 0.5 = 0.5x playback speed

eParamID_TransportRequestedSpeed Parameter Descriptor


```
[
  {
    "param_type": "string",
    "descriptor_type": "string",
    "param_id": "eParamID_TransportRequestedSpeed",
    "param_name": "Requested Speed",
    "persistence_type": "ephemeral",
    "register_type": "excluded",
    "factory_reset_type": "never",
    "relations": {},
    "class_names": [],
    "string_attributes": [
      {
        "name": "description",
        "value": "Speed requested for variable speed playback.
                  Only used with the eTCVarPlay transport
                  command."
      },
      {
        "name": "menu_number",
        "value": "125.1"
      }
    ],
    "integer_attributes": [],
    "min_length": 0,
    "max_length": 20,
    "default_value": "0.0"
  }
]
```


Normal Playback Speed Command

To initiate normal playback speed, set the **eParamID_TransportRequestedSpeed** to the value of "1".

```
http://192.168.0.2/config?action=set&paramid=eParamID_
TransportRequestedSpeed&value=1
```

16x Playback Speed Command

To initiate 16x playback speed, set the **eParamID_TransportRequestedSpeed** to the value of "16".

```
http://192.168.0.2/config?action=set&paramid=eParamID_
TransportRequestedSpeed&value=16
```

-1.5x Playback Speed Command

To initiate -1.5x playback speed, set the **eParamID_TransportRequestedSpeed** to the value of "-1.5".

```
http://192.168.0.2/config?action=set&paramid=eParamID_
TransportRequestedSpeed&value=-1.5
```

Pause Command

To pause playback, set the **eParamID_TransportRequestedSpeed** to the value of "0".

```
http://192.168.0.2/config?action=set&paramid=eParamID_
TransportRequestedSpeed&value=0
```

0.5 Speed Command

To initiate half speed playback, set the **eParamID_TransportRequestedSpeed** to the value of "0.5".

```
http://192.168.0.2/config?action=set&paramid=eParamID_
TransportRequestedSpeed&value=0.5
```

Formatting Media

Using both the **eParamID_FileSystemFormat** parameter and the **eParamID_StorageCommand** parameter, you can format media as either HFS+ (also known as Mac OS Extended), or ExFAT.

The **eParamID_FileSystemFormat** defines what type of disk formatting will be done when the **eParamID_StorageCommand** param is set.

Use the **eParamID_StorageCommand** to erase and format the media.

eParamID_FileSystemFormat Parameter Descriptor


```
[
  {
     "param_type": "enum",
     "descriptor_type": "enum",
     "param_id": "eParamID_FileSystemFormat",
     "param_name": "File System Formatting",
     "persistence_type": "persistent",
     "register_type": "excluded",
     "factory_reset_type": "never",
     "relations": {},
     "class_names": [],
     "string_attributes": [
        {
           "name": "description",
           "value": "Defines what file system format will be used
                     when formatting media via 6.2 Format Media."
        },
        {
           "name": "menu_number",
           "value": "16.0"
        }
     ],
     "integer_attributes": [
        {
           "name": "ParamGroup",
           "value": 3
        }
     ],
     "enum_values": [
        {
           "value": 0,
           "text": "HFS+",
           "short_text": "HFS+"
        },
        {
           "value": 1,
           "text": "ExFAT",
           "short_text": "ExFAT"
        }
     ],
     "min_value": 0,
     "max_value": 1,
     "default_value": 0,
     "adjust_by": 1,
     "scale_by": 1,
     "offset_by": 0,
     "display_precision": 0,
     "units": ""
  }
]
```


Specify Media Format as HFS+ Command

To indicate that media formatting should be done as HFS+, set the **eParamID_FileSystemFormat** to the value of "0".

```
http://192.168.0.2/config?action=set&paramid=eParamID_
FileSystemFormat&value=0
```

Specify Media Format as ExFAT Command

To indicate that media formatting should be done as ExFAT, set the **`eParamID_FileSystemFormat`** to the value of "1".

```
http://192.168.0.2/config?action=set&paramid=eParamID_
FileSystemFormat&value=1
```

eParamID_StorageCommand Parameter Descriptor


```
[
  {
     "param_type": "enum",
     "descriptor_type": "enum",
     "param_id": "eParamID_StorageCommand",
     "param_name": "Storage Command",
     "persistence_type": "ephemeral",
     "register_type": "excluded",
     "factory_reset_type": "never",
     "relations": {},
     "class_names": [
        "panel_hidden",
        "volatile"
     ],
     "string_attributes": [
        {
           "name": "description",
           "value": "Internal parameter for controlling use of
                     media."
        },
        {
           "name": "menu_number",
           "value": "123.1"
        }
     ],
     "integer_attributes": [],
     "enum_values": [
        {
           "value": 0,
           "text": "None",
           "short_text": "None"
        },
        {
           "value": 1,
           "text": "Acquire",
           "short_text": "Acquire"
        },
        {
           "value": 2,
           "text": "Release",
           "short_text": "Release"
        },
        {
           "value": 3,
           "text": "Rollover",
           "short_text": "Rollover"
        },
        {
           "value": 4,
           "text": "Erase",
           "short_text": "Erase"
        }
     ],
```

```
        "min_value": 0,
        "max_value": 4,
        "default_value": 0,
        "adjust_by": 1,
        "scale_by": 1,
        "offset_by": 0,
        "display_precision": 0,
        "units": ""
    }
]
```


Format Media Command

To erase and format media, set the **eParamID_StorageCommand** parameter to the value of "4" (Erase).

```
http://192.168.0.2/config?action=set&paramid=eParamID_
StorageCommand&value=4
```

Creating Sub-Clips on PAK Media

You can create sub-clips from larger clips on the PAK media, isolating a section of the clip to create a separate sub-clip. The Ki Pro Ultra subsystem that controls creating sub-clips is known as **mediaedit**. You can use the REST API to send a **mediaedit** command that will initiate a separate process to create the new clip.

NOTE: The 'mediaedit' command is separate from the typical 'config' commands frequently referenced elsewhere in the AJA REST API.

The **mediaedit** REST API supports the following actions:

- Subclip: Starts a sub-clip job to create a new sub-clip
- Status: Returns the status of a job
- Kill: Kills a running job

Requirements

1. The original clip and the subclip MUST be on the same drive or the process will hang. We do not support writing to the second drive.
2. There must be room remaining on the drive or the process will error out.
3. The TC (timecode) values (either start or duration) must be limited to a single clip. It will not span clips or create blank frames in the new clip.
4. Creating sub-clips is supported only with MOV files.

Start a Sub-Clip Job

Example Sub-Clip Command

```
http://192.168.0.2/mediaedit?action=subclip&arg1=/mnt/S1/AJA/
clip04.mov&arg2=/mnt/S1/AJA/subclip04.mov&arg3=10:00:20:00&a
rg4=00:01:00:00
```

Return Fields

error - Contains any error that occurred upon execution. It will be set to 'none' when no error has occurred and the job has launched.

id - Will be present if the job executed. It contains the job ID for this job and will be dynamic for each job. You'll need this ID to perform future actions on this job.

Example Sub-Clip Output

```
error: none
id: 3146
```

Command Syntax

To build a sub-clip command, use the following syntax:

After the protocol (for example, **`http://`**) and domain name (for example, **`192.168.0.2`**), the media edit API is exposed at the path **`/mediaedit`**.

```
http://192.168.0.2/mediaedit
```

Following the path, a question mark is used as a separator between the URL path and the action command:

```
http://192.168.0.2/mediaedit?
```

The **`action=subclip`** command comes next:

```
http://192.168.0.2/mediaedit?action=subclip
```

Next, add an ampersand '&', followed by the four arguments, each separated by ampersands '&': 'arg1&arg2&arg3&arg4'.

Example of Four Arguments

1. arg1 - path to the input clip

   ```
   arg1=/mnt/S1/AJA/clip04.mov
   ```

2. arg2 - path to the output clip

   ```
   arg2=/mnt/S1/AJA/subclip04.mov
   ```

3. arg3 - timecode inside input clip to start sub-clip

   ```
   arg3=01:27:33:03
   ```

4. arg4 - timecode duration of sub-clip

   ```
   arg4=00:01:00:00
   ```

Use complete paths for 'arg1' and 'arg2'. For instance, if your PAK media is mounted in slot one of the device, the path to the media will be:

```
/mnt/S1/AJA
```

If your PAK media is mounted in slot two, the path will be:

```
/mnt/S2/AJA
```

For names of clips and sub-clip files, include the **`.mov`** extension.

Return the Status of a Job

To find out the status of a job, you'll need to pass the ID of the job in question.

Example Status Command

```
http://192.168.1.44/mediaedit?action=status&id=3146
```

Return Fields

error - Returns any error that has occurred. This will return 'none' until the job is no longer running. Once the job is no longer running, if the job completed successfully, the error will continue to return 'none'.

id - Returns the ID of the job.

status - Returns the current status of the job. This is job specific, but is usually something like a percent completed or reason for failure.

running - "yes" if the job is still running, "no" if the job has finished.

time_now - Time stamp of the request (all time stamps are in Unix time)

time_start - Time stamp when job was started

time_stop - Time stamp when job ended (will be 0 when the job is still running)

Example Output

```
error: none
id: 3146
status: 44.75% complete
running: yes
time_now: 1472148646
time_start: 1472148607
time_stop: 0
```

Kill a Running Job

To kill a running job, you'll need to pass the ID of the job in question.

Example Kill Job Command

```
http://192.168.1.44/mediaedit?action=kill&id=3146
```

Return Fields

error - Contains any error that occurred when attempting to kill the job. It will be set to 'none' when no error has occurred.

id - Returns the ID of the job that was killed.

Example Output

```
error: none
id: 3146
```

Downloading and Uploading Clips

You can download and upload video clips that are in .mov format to your Ki Pro device using the cURL command line utility.

Before you can download and upload clips, the Ki Pro device first needs to be in Data Transfer mode (Data – LAN mode), which can be done with the **`eParamID_MediaState`** parameter. This parameter defines whether the Ki Pro device is in normal Record-Play mode or Data Transfer mode. If a transfer is in progress when this parameter is set back to Record-Play mode, the transfer will be aborted.

eParamID_MediaState


```
[
  {
     "param_type": "enum",
     "descriptor_type": "enum",
     "param_id": "eParamID_MediaState",
     "param_name": "Media State",
     "persistence_type": "ephemeral",
     "register_type": "included",
     "factory_reset_type": "warm",
     "relations": {},
     "class_names": [],
     "string_attributes": [
        {
           "name": "description",
           "value": "Define whether the Ki Pro device is in
                     normal Record-Play or Data Transfer mode."
        },
        {
           "name": "menu_number",
           "value": "12.1"
        }
     ],
     "integer_attributes": [
        {
           "name": "ParamGroup",
           "value": 3
        }
     ],
     "enum_values": [
        {
           "value": 0,
           "text": "Record-Play",
           "short_text": "Record-Play"
        },
        {
           "value": 1,
           "text": "Data - LAN",
           "short_text": "Data - LAN"
        }
     ],
     "min_value": 0,
     "max_value": 3,
     "default_value": 0,
     "adjust_by": 1,
     "scale_by": 1,
     "offset_by": 0,
     "display_precision": 0,
     "units": ""
  }
]
```


Data - LAN mode Command

To put the Ki Pro devide into Data - LAN mode, set the **eParamID_MediaState** parameter to the value of "1".

```
http://192.168.0.2/config?action=set&paramid=eParamID_
MediaState&value=1
```

eParamID_StoragePath

The **eParamID_StoragePath** stores the path to the location on the filesystem to record or play clips (for example, '/mnt/S1/AJA').

NOTE: The path '/media' can be used instead of '/mnt/S1/AJA' because '/media' points to the currently active storage path.


```
[
  {
     "param_type": "string",
     "descriptor_type": "string",
     "param_id": "eParamID_StoragePath",
     "param_name": "Storage Path",
     "persistence_type": "ephemeral",
     "register_type": "excluded",
     "factory_reset_type": "never",
     "relations": {},
     "class_names": [
        "readonly"
     ],
     "string_attributes": [
        {
           "name": "description",
           "value": "Path to location on filesystem to record or
                     play clips."
        },
        {
           "name": "menu_number",
           "value": "123.1"
        }
     ],
     "integer_attributes": [],
     "min_length": 0,
     "max_length": 20,
     "default_value": ""
  }
]
```


To retrieve the current storage path for your Ki Pro device, use this **get** command:

```
http://192.168.0.2/config?action=get&paramid=eParamID_
StoragePath
```

An example of the REST API response to the above command is:


```
{"paramid":"2063663370","name":"eParamID_
StoragePath","value":"/mnt/S1/AJA","value_name":""}
```


Download Command

To download a .mov file from a Ki Pro device, use the cURL command line utility with a terminal prompt.

In the following command example, **--output** instructs the server to create a new file rather than sending the download to your terminal window.

example-filename-downloaded.mov refers to the file name of the file after it has been downloaded. You can use any file name you like. It doesn't need to match the file name of the file you are downloading from the Ki Pro device. **example-filename-source.mov** refers to the file name of the .mov file that is residing on your Ki Pro device.

```
curl -v0 --output example-filename-downloaded.mov
http://192.168.0.2/media/example-filename-source.mov
```

Substitute the example IP address **192.168.0.2** with your actual Ki Pro device URL address.

Here is an example of the HTTP download request that AJA's browser-based javascript download initiates:

```
GET /media/example-filename-source.mov HTTP/1.1
Host: 192.168.0.2
Connection: keep-alive
```

This will initiate the download of a single file/clip from the currently selected media.

Here is an example of the response that the client will get:

```
HTTP/1.0 200 OK
Content-Type: video/quicktime
Content-disposition: attachment; filename=example-filename-source.mov
Content-Length: 359354368
```

Upload Command

To upload a .mov file from your computer to your Ki Pro device, use the following cURL command as an example:

```
curl -v0 -H "X-File-Size: 98521216" -H "X-File-Name: file-name-after-upload.mov" --url http://192.168.0.2/media --upload-file ./example-file-name-on-local-system.mov
```

- The **`X-File-Size`** header takes the size in bytes of the clip to be uploaded.
- The file will be given the name specified in the **`X-File-Name`** header. This will be the filename on the Ki Pro after the upload and has no relationship to the actual local file name.
- **`--upload-file`** argument: This filename is provided to locate the file on your local system.

Here is an example of an HTTP request that AJA's browser-based javascript upload makes:

```
PUT /media HTTP/1.1
Host: 192.168.0.2
Connection: keep-alive
Content-Length: 98521216
X-File-Name: file-name-after-upload.mov
X-File-Size: 98521216
Content-Type: video/quicktime
Accept: application/json, text/javascript, */*; q=0.01
X-File-Type: video/quicktime
```

This will upload the file to the currently selected media. The following is an example response the client will get from the server:

```
HTTP/1.1 200 OK
Server: Seminole/2.71 (Linux; E21)
Date: Tue, 12 May 2020 22:46:34 GMT
Connection: keep-alive
Cache-Control: no-cache
Content-Type: text/json
```

No body is included in the response.

Chapter 6 – Common Ki Pro Ultra Plus and Ki Pro Ultra 12G Commands

To access this chapter online, please see:

https://gitlab.aja.com/pub/rest_api/-/blob/master/KiProUltraPlus_KiProUltra12G/06_KPUP_KPU12G_Commands.md

Universal Transport: Start, Stop, Record, Fast Forward

The Ki Pro Ultra Plus and Ki Pro Ultra 12G subsystem that controls video transport is known as Transport. You can control Universal Transport commands such as Start, Stop, Record, and Fast Forward independently with the **`eParamID_TransportCommand`** parameter.

eParamID_TransportCommand Parameter Descriptor


```
[
  {
    "param_type": "enum",
    "descriptor_type": "enum",
    "param_id": "eParamID_TransportCommand",
    "param_name": "Transport Command",
    "persistence_type": "ephemeral",
    "register_type": "excluded",
    "factory_reset_type": "never",
    "relations": {},
    "class_names": [
      "volatile"
    ],
    "string_attributes": [
      {
        "name": "description",
        "value": "Transport command."
      },
      {
      "name": "menu_number",
      "value": "125.1"
      }
   ],
    "integer_attributes": [],
    "enum_values": [
      {
        "value": 0,
        "text": "No Command",
        "short_text": "No Command"
      },
      {
        "value": 1,
        "text": "Play Command",
        "short_text": "Play Command"
      },
      {
        "value": 3,
        "text": "Record Command",
        "short_text": "Record Command"
      },
```

```
{
  "value": 4,
  "text": "Stop Command",
  "short_text": "Stop Command"
},
{
  "value": 5,
  "text": "Fast Forward",
  "short_text": "Fast Forward"
},
{
  "value": 6,
  "text": "Fast Reverse",
  "short_text": "Fast Reverse"
},
{
  "value": 7,
  "text": "Single Step Forward",
  "short_text": "Single Step Forward"
},
{
  "value": 8,
  "text": "Single Step Reverse",
  "short_text": "Single Step Reverse"
},
{
  "value": 9,
  "text": "Next Clip",
  "short_text": "Next Clip"
},
{
  "value": 10,
  "text": "Previous Clip",
  "short_text": "Previous Clip"
},
{
  "value": 11,
  "text": "Variable Speed Play",
  "short_text": "Variable Speed Play"
},
{
  "value": 12,
  "text": "Preroll",
  "short_text": "Preroll"
},
{
  "value": 13,
  "text": "Assemble Edit",
  "short_text": "Assemble Edit"
},
{
  "value": 14,
  "text": "Cue",
  "short_text": "Cue"
},
{
  "value": 15,
  "text": "Shutdown",
  "short_text": "Shutdown"
},
```

```
        {
          "value": 16,
          "text": "Play At System Time",
          "short_text": "Play At System Time"
        },
        {
          "value": 17,
          "text": "Record At System Time",
          "short_text": "Record At System Time"
        },
        {
          "value": 18,
          "text": "Go To Idle",
          "short_text": "Go To Idle"
        }
      ],
      "min_value": 0,
      "max_value": 18,
      "default_value": 0,
      "adjust_by": 1,
      "scale_by": 1,
      "offset_by": 0,
      "display_precision": 0,
      "units": ""
    }
  ]
```


Play Command

To initiate the play command, set the **eParamID_TransportCommand** to the value of "1".

```
http://192.168.0.2/config?action=set&paramid=eParamID_
TransportCommand&value=1
```

Stop Command (once to pause, twice to stop)

The first time the stop command is made, the unit will pause. The second time the stop command is made, the unit will stop (idle).

To initiate the stop command, set the **eParamID_TransportCommand** to the value of "4".

```
http://192.168.0.2/config?action=set&paramid=eParamID_
TransportCommand&value=4
```

Record Command

To start recording, set the **eParamID_TransportCommand** to the value of "3".

```
http://192.168.0.2/config?action=set&paramid=eParamID_
TransportCommand&value=3
```

Fast Forward Command

To fast forward, set the **eParamID_TransportCommand** to the value of "5".

```
http://192.168.0.2/config?action=set&paramid=eParamID_
TransportCommand&value=5
```

Setting Playback Speed (v3.1.1 and above)

The Ki Pro Ultra Plus and Ki Pro Ultra 12G subsystem that controls video transport speed is known as TransportRequestedSpeed.

You can control the speed of playing forward and playing in reverse with the **eParamID_TransportRequestedSpeed** parameter. Set the parameter value to whatever forward or reverse speed you need by using a floating-point value representing your desired playback speed. The value can be any positive or negative floating-point value. For example:

- 1 = normal playback speed
- 16 = 16x playback speed
- -1.5 = 1.5x reverse speed
- 0 = pause
- 0.5 = 0.5x playback speed

eParamID_TransportRequestedSpeed Parameter Descriptor


```
[
  {
    "param_type": "string",
    "descriptor_type": "string",
    "param_id": "eParamID_TransportRequestedSpeed",
    "param_name": "Requested Speed",
    "persistence_type": "ephemeral",
    "register_type": "excluded",
    "factory_reset_type": "never",
    "relations": {},
    "class_names": [],
    "string_attributes": [
      {
        "name": "description",
        "value": "Speed requested for variable speed playback.
                  Only used with the eTCVarPlay transport
                  command."
      },
      {
        "name": "menu_number",
        "value": "125.1"
      }
    ],
    "integer_attributes": [],
    "min_length": 0,
    "max_length": 20,
    "default_value": "0.0"
  }
]
```


Normal Playback Speed Command

To initiate normal playback speed, set the **eParamID_TransportRequestedSpeed** to the value of "1".

```
http://192.168.0.2/config?action=set&paramid=eParamID_
TransportRequestedSpeed&value=1
```

16x Playback Speed Command

To initiate 16x playback speed, set the **eParamID_TransportRequestedSpeed** to the value of "16".

```
http://192.168.0.2/config?action=set&paramid=eParamID_
TransportRequestedSpeed&value=16
```

-1.5x Playback Speed Command

To initiate -1.5x playback speed, set the **eParamID_ TransportRequestedSpeed** to the value of "-1.5".

```
http://192.168.0.2/config?action=set&paramid=eParamID_
TransportRequestedSpeed&value=-1.5
```

Pause Command

To pause playback, set the **eParamID_TransportRequestedSpeed** to the value of "0".

```
http://192.168.0.2/config?action=set&paramid=eParamID_
TransportRequestedSpeed&value=0
```

0.5 Speed Command

To initiate half speed playback, set the **eParamID_TransportRequestedSpeed** to the value of "0.5".

```
http://192.168.0.2/config?action=set&paramid=eParamID_
TransportRequestedSpeed&value=0.5
```

Formatting Media

Using both the **eParamID_FileSystemFormat** parameter and the **eParamID_ StorageCommand** parameter, you can format media as either HFS+ (also known as Mac OS Extended), or ExFAT.

The **eParamID_FileSystemFormat** defines what type of disk formatting will be done when the **eParamID_StorageCommand** param is set.

Use the **eParamID_StorageCommand** to erase and format the media.

eParamID_FileSystemFormat Parameter Descriptor


```
[
  {
     "param_type": "enum",
     "descriptor_type": "enum",
     "param_id": "eParamID_FileSystemFormat",
     "param_name": "File System Formatting",
     "persistence_type": "persistent",
     "register_type": "excluded",
     "factory_reset_type": "never",
     "relations": {},
     "class_names": [],
     "string_attributes": [
        {
           "name": "description",
           "value": "Defines what file system format will be used
                     when formatting media via 6.2 Format Media."
        },
        {
           "name": "menu_number",
           "value": "16.0"
        }
     ],
     "integer_attributes": [
        {
           "name": "ParamGroup",
           "value": 3
        }
     ],
     "enum_values": [
        {
           "value": 0,
           "text": "HFS+",
           "short_text": "HFS+"
        },
        {
           "value": 1,
           "text": "ExFAT",
           "short_text": "ExFAT"
        }
     ],
     "min_value": 0,
     "max_value": 1,
     "default_value": 0,
     "adjust_by": 1,
     "scale_by": 1,
     "offset_by": 0,
     "display_precision": 0,
     "units": ""
  }
]
```


Specify Media Format as HFS+ Command

To indicate that media formatting should be done as HFS+, set the **eParamID_FileSystemFormat** to the value of "0".

```
http://192.168.0.2/config?action=set&paramid=eParamID_
FileSystemFormat&value=0
```

Specify Media Format as ExFAT Command

To indicate that media formatting should be done as ExFAT, set the **`eParamID_FileSystemFormat`** to the value of "1".

```
http://192.168.0.2/config?action=set&paramid=eParamID_
FileSystemFormat&value=1
```

eParamID_StorageCommand Parameter Descriptor


```
[
  {
     "param_type": "enum",
     "descriptor_type": "enum",
     "param_id": "eParamID_StorageCommand",
     "param_name": "Storage Command",
     "persistence_type": "ephemeral",
     "register_type": "excluded",
     "factory_reset_type": "never",
     "relations": {},
     "class_names": [
        "panel_hidden",
        "volatile"
     ],
     "string_attributes": [
        {
           "name": "description",
           "value": "Internal parameter for controlling use of
                     media."
        },
        {
           "name": "menu_number",
           "value": "123.1"
        }
     ],
     "integer_attributes": [],
     "enum_values": [
        {
           "value": 0,
           "text": "None",
           "short_text": "None"
        },
        {
           "value": 1,
           "text": "Acquire",
           "short_text": "Acquire"
        },
        {
           "value": 2,
           "text": "Release",
           "short_text": "Release"
        },
        {
           "value": 3,
           "text": "Rollover",
           "short_text": "Rollover"
        },
        {
           "value": 4,
           "text": "Erase",
           "short_text": "Erase"
        }
     ],
```

```
        "min_value": 0,
        "max_value": 4,
        "default_value": 0,
        "adjust_by": 1,
        "scale_by": 1,
        "offset_by": 0,
        "display_precision": 0,
        "units": ""
    }
]
```


Format Media Command

To erase and format media, set the **eParamID_StorageCommand** parameter to the value of "4" (Erase).

```
http://192.168.0.2/config?action=set&paramid=eParamID_
StorageCommand&value=4
```

Creating Sub-Clips on PAK Media

You can create sub-clips from larger clips on the PAK media, isolating a section of the clip to create a separate sub-clip. The Ki Pro Ultra Plus and Ki Pro Ultra 12G subsystem that controls creating sub-clips is known as **mediaedit**. You can use the REST API to send a **mediaedit** command that will initiate a separate process to create the new clip.

NOTE: The 'mediaedit' command is separate from the typical 'config' commands frequently referenced elsewhere in the AJA REST API.

The **mediaedit** REST API supports the following actions:

- Subclip: Starts a sub-clip job to create a new sub-clip
- Status: Returns the status of a job
- Kill: Kills a running job

Requirements

1. The original clip and the subclip MUST be on the same drive or the process will hang. We do not support writing to the second drive.
2. There must be room remaining on the drive or the process will error out.
3. The TC (timecode) values (either start or duration) must be limited to a single clip. It will not span clips or create blank frames in the new clip.
4. Creating sub-clips is supported only with MOV files.

Start a Sub-Clip Job

Example Sub-Clip Command

```
http://192.168.0.2/mediaedit?action=subclip&arg1=/mnt/S1/AJA/
clip04.mov&arg2=/mnt/S1/AJA/subclip04.mov&arg3=10:00:20:00&a
rg4=00:01:00:00
```

Return Fields

error - Contains any error that occurred upon execution. It will be set to 'none' when no error has occurred and the job has launched.

id - Will be present if the job executed. It contains the job ID for this job and will be dynamic for each job. You'll need this ID to perform future actions on this job.

Example Sub-Clip Output

```
error: none
id: 3146
```

Command Syntax

To build a sub-clip command, use the following syntax:

After the protocol (for example, **`http://`**) and domain name (for example, **`192.168.0.2`**), the media edit API is exposed at the path **`/mediaedit`**.

```
http://192.168.0.2/mediaedit
```

Following the path, a question mark is used as a separator between the URL path and the action command:

```
http://192.168.0.2/mediaedit?
```

The **`action=subclip`** command comes next:

```
http://192.168.0.2/mediaedit?action=subclip
```

Next, add an ampersand '&', followed by the four arguments, each separated by ampersands '&': 'arg1&arg2&arg3&arg4'.

Example of Four Arguments

1. arg1 - path to the input clip

   ```
   arg1=/mnt/S1/AJA/clip04.mov
   ```

2. arg2 - path to the output clip

   ```
   arg2=/mnt/S1/AJA/subclip04.mov
   ```

3. arg3 - timecode inside input clip to start sub-clip

   ```
   arg3=01:27:33:03
   ```

4. arg4 - timecode duration of sub-clip

   ```
   arg4=00:01:00:00
   ```

Use complete paths for 'arg1' and 'arg2'. For instance, if your PAK media is mounted in slot one of the device, the path to the media will be:

```
/mnt/S1/AJA
```

If your PAK media is mounted in slot two, the path will be:

```
/mnt/S2/AJA
```

For names of clips and sub-clip files, include the **`.mov`** extension.

Return the Status of a Job

To find out the status of a job, you'll need to pass the ID of the job in question.

Example Status Command

```
http://192.168.1.44/mediaedit?action=status&id=3146
```

Return Fields

error - Returns any error that has occurred. This will return 'none' until the job is no longer running. Once the job is no longer running, if the job completed successfully, the error will continue to return 'none'.

id - Returns the ID of the job.

status - Returns the current status of the job. This is job specific, but is usually something like a percent completed or reason for failure.

running - "yes" if the job is still running, "no" if the job has finished.

time_now - Time stamp of the request (all time stamps are in Unix time)

time_start - Time stamp when job was started

time_stop - Time stamp when job ended (will be 0 when the job is still running)

Example Output

```
error: none
id: 3146
status: 44.75% complete
running: yes
time_now: 1472148646
time_start: 1472148607
time_stop: 0
```

Kill a Running Job

To kill a running job, you'll need to pass the ID of the job in question.

Example Kill Job Command

```
http://192.168.1.44/mediaedit?action=kill&id=3146
```

Return Fields

error - Contains any error that occurred when attempting to kill the job. It will be set to 'none' when no error has occurred.

id - Returns the ID of the job that was killed.

Example Output

```
error: none
id: 3146
```

Downloading and Uploading Clips

You can download and upload video clips that are in .mov format to your Ki Pro device using the cURL command line utility.

Before you can download and upload clips, the Ki Pro device first needs to be in Data Transfer mode (Data – LAN mode), which can be done with the **`eParamID_MediaState`** parameter. This parameter defines whether the Ki Pro device is in normal Record-Play mode or Data Transfer mode. If a transfer is in progress when this parameter is set back to Record-Play mode, the transfer will be aborted.

eParamID_MediaState


```
[
  {
     "param_type": "enum",
     "descriptor_type": "enum",
     "param_id": "eParamID_MediaState",
     "param_name": "Media State",
     "persistence_type": "ephemeral",
     "register_type": "included",
     "factory_reset_type": "warm",
     "relations": {},
     "class_names": [],
     "string_attributes": [
        {
           "name": "description",
           "value": "Define whether the Ki Pro device is in
                     normal Record-Play or Data Transfer mode."
        },
        {
           "name": "menu_number",
           "value": "12.1"
        }
     ],
     "integer_attributes": [
        {
           "name": "ParamGroup",
           "value": 3
        }
     ],
     "enum_values": [
        {
           "value": 0,
           "text": "Record-Play",
           "short_text": "Record-Play"
        },
        {
           "value": 1,
           "text": "Data - LAN",
           "short_text": "Data - LAN"
        }
     ],
     "min_value": 0,
     "max_value": 3,
     "default_value": 0,
     "adjust_by": 1,
     "scale_by": 1,
     "offset_by": 0,
     "display_precision": 0,
     "units": ""
  }
]
```

Data - LAN mode Command

To put the Ki Pro devide into Data - LAN mode, set the **eParamID_MediaState** parameter to the value of "1".

```
http://192.168.0.2/config?action=set&paramid=eParamID_
MediaState&value=1
```

eParamID_StoragePath

The **eParamID_StoragePath** stores the path to the location on the filesystem to record or play clips (for example, '/mnt/S1/AJA').

NOTE: The path '/media' can be used instead of '/mnt/S1/AJA' because '/media' points to the currently active storage path.


```
[
  {
     "param_type": "string",
     "descriptor_type": "string",
     "param_id": "eParamID_StoragePath",
     "param_name": "Storage Path",
     "persistence_type": "ephemeral",
     "register_type": "excluded",
     "factory_reset_type": "never",
     "relations": {},
     "class_names": [
        "readonly"
     ],
     "string_attributes": [
        {
           "name": "description",
           "value": "Path to location on filesystem to record or
                     play clips."
        },
        {
           "name": "menu_number",
           "value": "123.1"
        }
     ],
     "integer_attributes": [],
     "min_length": 0,
     "max_length": 20,
     "default_value": ""
  }
]
```


To retrieve the current storage path for your Ki Pro device, use this **get** command:

```
http://192.168.0.2/config?action=get&paramid=eParamID_
StoragePath
```

An example of the REST API response to the above command is:


```
{"paramid":"2063663370","name":"eParamID_
StoragePath","value":"/mnt/S1/AJA","value_name":""}
```


Download Command

To download a .mov file from a Ki Pro device, use the cURL command line utility with a terminal prompt.

In the following command example, **--output** instructs the server to create a new file rather than sending the download to your terminal window.

example-filename-downloaded.mov refers to the file name of the file after it has been downloaded. You can use any file name you like. It doesn't need to match the file name of the file you are downloading from the Ki Pro device. **example-filename-source.mov** refers to the file name of the .mov file that is residing on your Ki Pro device.

```
curl -v0 --output example-filename-downloaded.mov
http://192.168.0.2/media/example-filename-source.mov
```

Substitute the example IP address **192.168.0.2** with your actual Ki Pro device URL address.

Here is an example of the HTTP download request that AJA's browser-based javascript download initiates:

```
GET /media/example-filename-source.mov HTTP/1.1
Host: 192.168.0.2
Connection: keep-alive
```

This will initiate the download of a single file/clip from the currently selected media.

Here is an example of the response that the client will get:

```
HTTP/1.0 200 OK
Content-Type: video/quicktime
Content-disposition: attachment; filename=example-
filename-source.mov
Content-Length: 359354368
```

Upload Command

To upload a .mov file from your computer to your Ki Pro device, use the following cURL command as an example:

```
curl -v0 -H "X-File-Size: 98521216" -H "X-File-Name: file-
name-after-upload.mov" --url http://192.168.0.2/media
--upload-file ./example-file-name-on-local-system.mov
```

- The **X-File-Size** header takes the size in bytes of the clip to be uploaded.
- The file will be given the name specified in the **X-File-Name** header. This will be the filename on the Ki Pro after the upload and has no relationship to the actual local file name.
- **--upload-file** argument: This filename is provided to locate the file on your local system.

Here is an example of an HTTP request that AJA's browser-based javascript upload makes:

```
PUT /media HTTP/1.1
Host: 192.168.0.2
Connection: keep-alive
Content-Length: 98521216
X-File-Name: file-name-after-upload.mov
X-File-Size: 98521216
Content-Type: video/quicktime
Accept: application/json, text/javascript, */*; q=0.01
X-File-Type: video/quicktime
```

This will upload the file to the currently selected media. The following is an example response the client will get from the server:

```
HTTP/1.1 200 OK
Server: Seminole/2.71 (Linux; E21)
Date: Tue, 12 May 2020 22:46:34 GMT
Connection: keep-alive
Cache-Control: no-cache
Content-Type: text/json
```

No body is included in the response.

Selecting the Channels for Headphone Audio

With the **`eParamID_AudioChannelsFocus`** parameter, you can select one of eight audio pairs to monitor through the headphone port. This setting also affects the rear RCA audio monitor jacks and the front panel VU meters. The available options are:

- Channels 1 & 2
- Channels 3 & 4
- Channels 5 & 6
- Channels 7 & 8
- Channels 9 & 10
- Channels 11 & 12
- Channels 13 & 14
- Channels 15 & 16

eParamID_AudioChannelsFocus Parameter Descriptor


```
[
  {
     "param_type": "enum",
     "descriptor_type": "enum",
     "param_id": "eParamID_AudioChannelsFocus",
     "param_name": "Headphone Audio",
     "persistence_type": "persistent",
     "register_type": "included",
     "factory_reset_type": "warm",
     "relations": {},
     "class_names": [],
     "string_attributes": [
        {
           "name": "description",
           "value": "Selects the audio channels output to the
                     headphones and shown in the VU Meters."
        },
        {
           "name": "menu_number",
           "value": "2.4"
        }
     ],
     "integer_attributes": [
        {
           "name": "ParamGroup",
           "value": 2
        }
     ],
     "enum_values": [
        {
           "value": 0,
           "text": "Channels 1-2",
           "short_text": "Channels 1-2"
        },
        {
           "value": 1,
           "text": "Channels 3-4",
           "short_text": "Channels 3-4"
        },
        {
           "value": 2,
           "text": "Channels 5-6",
           "short_text": "Channels 5-6"
        },
        {
           "value": 3,
           "text": "Channels 7-8",
           "short_text": "Channels 7-8"
        },
        {
           "value": 4,
           "text": "Channels 9-10",
           "short_text": "Channels 9-10"
        },
        {
           "value": 5,
           "text": "Channels 11-12",
           "short_text": "Channels 11-12"
        },
```

```
            {
                "value": 6,
                "text": "Channels 13-14",
                "short_text": "Channels 13-14"
            },
            {
                "value": 7,
                "text": "Channels 15-16",
                "short_text": "Channels 15-16"
            }
        ],
        "min_value": 0,
        "max_value": 7,
        "default_value": 0,
        "adjust_by": 1,
        "scale_by": 1,
        "offset_by": 0,
        "display_precision": 0,
        "units": ""
    }
]
```


To select Channels 1 & 2, set the **eParamID_AudioChannelsFocus** to the value of "0".

```
http://192.168.0.2/config?action=set&paramid=eParamID_
AudioChannelsFocus&value=0
```

To select Channels 3 & 4, set the **eParamID_AudioChannelsFocus** to the value of "1".

```
http://192.168.0.2/config?action=set&paramid=eParamID_
AudioChannelsFocus&value=1
```

To select Channels 5 & 6, set the **eParamID_AudioChannelsFocus** to the value of "2".

```
http://192.168.0.2/config?action=set&paramid=eParamID_
AudioChannelsFocus&value=2
```

To select Channels 7 & 8, set the **eParamID_AudioChannelsFocus** to the value of "3".

```
http://192.168.0.2/config?action=set&paramid=eParamID_
AudioChannelsFocus&value=3
```

To select Channels 9 & 10, set the **eParamID_AudioChannelsFocus** to the value of "4".

```
http://192.168.0.2/config?action=set&paramid=eParamID_
AudioChannelsFocus&value=4
```

To select Channels 11 & 12, set the **eParamID_AudioChannelsFocus** to the value of "5".

```
http://192.168.0.2/config?action=set&paramid=eParamID_
AudioChannelsFocus&value=5
```

To select Channels 13 & 14, set the **eParamID_AudioChannelsFocus** to the value of "6".

```
http://192.168.0.2/config?action=set&paramid=eParamID_
AudioChannelsFocus&value=6
```

To select Channels 15 & 16, set the **eParamID_AudioChannelsFocus** to the value of "7".

```
http://192.168.0.2/config?action=set&paramid=eParamID_
AudioChannelsFocus&value=7
```

Selecting the Headphone Audio Channel for Multi Channel Mode

(Multi Channel Mode only.) With the **eParamID_AudioEncodeChannelFocus** parameter, you can select which channel's audio is routed to the headphone output. The available options are:

- Channel 1 (default)
- Channel 2
- Channel 3
- Channel 4

eParamID_AudioEncodeChannelFocus Parameter Descriptor


```
[
  {
     "param_type": "enum",
     "descriptor_type": "enum",
     "param_id": "eParamID_AudioEncodeChannelFocus",
     "param_name": "Headphone Audio Channel",
     "persistence_type": "persistent",
     "register_type": "included",
     "factory_reset_type": "warm",
     "relations": {
        "disabled_by": [
           {
              "paramid": "eParamID_EncodeChannels",
              "value": 1
           }
        ]
     },
     "class_names": [],
     "string_attributes": [
        {
           "name": "description",
           "value": "Selects the Encode Channel that will feed
                     the headphones and VU Meters"
        },
        {
           "name": "menu_number",
           "value": "2.7"
        }
     ],
     "integer_attributes": [
        {
           "name": "ParamGroup",
           "value": 2
        }
     ],
     "enum_values": [
        {
           "value": 0,
           "text": "Channel 1",
           "short_text": "Channel 1"
        },
        {
           "value": 1,
           "text": "Channel 2",
           "short_text": "Channel 2"
        },
```

```
            {
                "value": 2,
                "text": "Channel 3",
                "short_text": "Channel 3"
            },
            {
                "value": 3,
                "text": "Channel 4",
                "short_text": "Channel 4"
            }
        ],
        "min_value": 0,
        "max_value": 4,
        "default_value": 0,
        "adjust_by": 1,
        "scale_by": 1,
        "offset_by": 0,
        "display_precision": 0,
        "units": ""
    }
]
```


To select Channel 1, set the **eParamID_AudioEncodeChannelFocus** to the value of "0".

```
http://192.168.0.2/config?action=set&paramid=eParamID_
AudioEncodeChannelFocus&value=0
```

To select Channel 2, set the **eParamID_AudioEncodeChannelFocus** to the value of "1".

```
http://192.168.0.2/config?action=set&paramid=eParamID_
AudioEncodeChannelFocus&value=1
```

To select Channel 3, set the **eParamID_AudioEncodeChannelFocus** to the value of "2".

```
http://192.168.0.2/config?action=set&paramid=eParamID_
AudioEncodeChannelFocus&value=2
```

To select Channel 4, set the **eParamID_AudioEncodeChannelFocus** to the value of "3".

```
http://192.168.0.2/config?action=set&paramid=eParamID_
AudioEncodeChannelFocus&value=3
```

Selecting SDI Monitor Channel

(Multi Channel Mode only.) You can select which channels are displayed by the SDI Monitor output with the **eParamID_SDIMonitorChannel** parameter.

eParamID_SDIMonitorChannel Parameter Descriptor


```
[
  {
    "param_type": "enum",
    "descriptor_type": "enum",
    "param_id": "eParamID_SDIMonitorChannel",
    "param_name": "SDI Monitor Channel",
    "persistence_type": "persistent",
    "register_type": "included",
    "factory_reset_type": "warm",
    "relations": {
      "disabled_by": [
        {
          "paramid": "eParamID_EncodeChannels",
          "value": 1
        }
      ]
    },
    "class_names": [],
    "string_attributes": [
      {
        "name": "description",
        "value": "Sets the SDI monitor's channel"
      },
      {
        "name": "menu_number",
        "value": "1.15"
      }
    ],
    "integer_attributes": [
      {
        "name": "ParamGroup",
        "value": 2
      }
    ],
    "enum_values": [
      {
        "value": 0,
        "text": "Channel 1",
        "short_text": "Channel 1"
      },
      {
        "value": 1,
        "text": "Channel 2",
        "short_text": "Channel 2"
      },
      {
        "value": 2,
        "text": "Channel 3",
        "short_text": "Channel 3"
      },
      {
        "value": 3,
        "text": "Channel 4",
        "short_text": "Channel 4"
      },
      {
        "value": 4,
        "text": "All Channels",
        "short_text": "All Channels"
      }
    ],
    "min_value": 0,
    "max_value": 4,
```

```
      "default_value": 4,
      "adjust_by": 1,
      "scale_by": 1,
      "offset_by": 0,
      "display_precision": 0,
      "units": ""
    }
  ]
```


To select Channel 1, set the **eParamID_SDIMonitorChannel** to the value of "0".

```
http://192.168.0.2/config?action=set&eParamID_
SDIMonitorChannel&value=0
```

To select Channel 2, set the **eParamID_SDIMonitorChannel** to the value of "1".

```
http://192.168.0.2/config?action=set&eParamID_
SDIMonitorChannel&value=1
```

To select Channel 3, set the **eParamID_SDIMonitorChannel** to the value of "2".

```
http://192.168.0.2/config?action=set&eParamID_
SDIMonitorChannel&value=2
```

To select Channel 4, set the **eParamID_SDIMonitorChannel** to the value of "3".

```
http://192.168.0.2/config?action=set&eParamID_
SDIMonitorChannel&value=3
```

To select All Channels (displays all four channels in quadrants), set the **eParamID_SDIMonitorChannel** to the value of "4".

```
http://192.168.0.2/config?action=set&eParamID_
SDIMonitorChannel&value=4
```

Selecting HDMI Monitor Output Channel

(Multi Channel Mode only.) You can select which channel(s) are displayed by the HDMI Monitor output with the **eParamID_HDMIOutChannel** parameter.

eParamID_HDMIOutChannel Parameter Descriptor


```
[
  {
     "param_type": "enum",
     "descriptor_type": "enum",
     "param_id": "eParamID_HDMIOutChannel",
     "param_name": "HDMI Out Channel",
     "persistence_type": "persistent",
     "register_type": "included",
     "factory_reset_type": "warm",
     "relations": {
        "disabled_by": [
           {
              "paramid": "eParamID_EncodeChannels",
              "value": 1
           }
        ]
     },
     "class_names": [],
     "string_attributes": [
        {
           "name": "description",
           "value": "Sets the HDMI output channel"
        },
        {
           "name": "menu_number",
           "value": "1.16"
        }
     ],
     "integer_attributes": [
        {
           "name": "ParamGroup",
           "value": 2
        }
     ],
     "enum_values": [
        {
           "value": 0,
           "text": "Channel 1",
           "short_text": "Channel 1"
        },
        {
           "value": 1,
           "text": "Channel 2",
           "short_text": "Channel 2"
        },
        {
           "value": 2,
           "text": "Channel 3",
           "short_text": "Channel 3"
        },
        {
           "value": 3,
           "text": "Channel 4",
           "short_text": "Channel 4"
        },
        {
           "value": 4,
           "text": "All Channels",
           "short_text": "All Channels"
        }
     ],
     "min_value": 0,
     "max_value": 4,
```

```
        "default_value": 4,
        "adjust_by": 1,
        "scale_by": 1,
        "offset_by": 0,
        "display_precision": 0,
        "units": ""
    }
]
```


To select Channel 1, set the **eParamID_HDMIOutChannel** to the value of "0".

```
http://192.168.0.2/config?action=set&paramid=eParamID_
HDMIOutChannel&value=0
```

To select Channel 2, set the **eParamID_HDMIOutChannel** to the value of "1".

```
http://192.168.0.2/config?action=set&paramid=eParamID_
HDMIOutChannel&value=1
```

To select Channel 3, set the **eParamID_HDMIOutChannel** to the value of "2".

```
http://192.168.0.2/config?action=set&paramid=eParamID_
HDMIOutChannel&value=2
```

To select Channel 4, set the **eParamID_HDMIOutChannel** to the value of "3".

```
http://192.168.0.2/config?action=set&paramid=eParamID_
HDMIOutChannel&value=3
```

To select All Channels, set the **eParamID_HDMIOutChannel** to the value of "4".

```
http://192.168.0.2/config?action=set&paramid=eParamID_
HDMIOutChannel&value=4
```