API documentation

<< Back to UI

Table of Contents

- Authentication
- JSON object definitions
  - Pipeline
  - Input Stream
  - Input Video
  - Input Audios
  - Input Captions
  - Input Metadata
  - Output Stream
  - Output Video
  - Output Audios
  - Output Captions
  - Output Metadata
  - Uploaded Logos
  - Pipeline State
  - Input Stream State
  - Elementary Streams
  - Output Stream State
- Configuration of All Pipelines
  - List All Existing Pipelines
  - Replace All Pipelines
- Configuration of a Single Pipeline
  - Create a New Pipeline
  - Get an Existing Pipeline
  - Modify an Existing Pipeline
  - Start an Existing Pipeline
  - Stop an Existing Pipeline
  - Remove an Existing Pipeline
- Retrieving Pipeline State
  - Get Pipeline Runtime State
- System
  - Version info
  - Get list of available network interfaces
  - Get list of available SDI devices
  - Get list of available NDI source devices
  - Perform reinstallation of the current version of the device
  - Retrieve state of reinstallation process
  - Perform factory reset of the device
  - Retrieve state of factory reset process
  - Perform restart of main service
  - Perform reboot of the machine
  - Perform shutdown of the machine
- Errors
  - Error response

Authentication

Each API request must contain authorization header for basic access authentication.

Name	Description
`Authorization`	Basic authentication credentials in format `Basic <base64>` where `<base64>` is Base64 encoded string `<username>:<password>`. Example: `Basic YWRtaW46cGFzc3dvcmQ=` (`admin:password`)

JSON object definitions

Pipeline

JSON object containing configuration for single pipeline.

Path	Type	Description
`version`	`String`	Version of pipeline JSON schema. Current version is `"0.16"`.
`id`	`Number`	Unique identifier of this pipeline.
`name`	`String`	Name of this pipeline displayed in UI.
`runningState`	`String`	Policy to restrict automatic pipeline loading after restart. Allowed values are `"INACTIVE"` or `"ACTIVE"`.
`inputs`	`Array of Input Streams`	List of configured input streams. See Input Stream.
`outputs`	`Array of Output Streams`	List of configured output streams. See Output Streams.
`uploadedLogos`	`Array of Uploaded Logos`	Configuration of input logos. See Uploaded Logos

Relation	Description
`self`	Link to this pipeline.
`state`	Link to current state of this pipeline.

Pipeline example:

```
{
  "version" : "0.16",
  "id" : 0,
  "name" : "Pipeline 1",
  "runningState" : "INACTIVE",
  "inputs" : [ {
    "id" : 0,
    "source" : {
      "streamType" : "MPEG_TS",
      "address" : {
        "url" : "srt://127.0.0.1:5001:lo",
        "srt" : {
          "mode" : "CALLER",
          "latency" : 50,
          "streamID" : "#!::u=admin,t=file,m=publish,r=results.csv",
          "passphrase" : "",
          "redundancy" : [ "srt://127.0.0.1:5002:lo" ]
        }
      },
      "complianceWith" : "AUTO",
      "addressPCR" : {
        "url" : "srt://0.0.0.0:5000:lo",
        "srt" : {
          "mode" : "LISTENER",
          "latency" : 50,
          "passphrase" : ""
        }
      }
    },
    "video" : {
      "format" : "J2K",
      "width" : 1920,
      "height" : 1080,
      "sampling" : "4:2:2",
      "depth" : 10,
      "scanRate" : "50",
      "scan" : "INTERLACED",
      "fieldOrder" : "TOP_FIELD_FIRST",
      "filters" : {
        "yOffset" : 16,
        "yGain" : 0.2,
        "rOffset" : 1,
        "rGain" : 0.01,
        "gOffset" : 2,
        "gGain" : 0.02,
        "bOffset" : 3,
        "bGain" : 0.03,
        "rgbClip" : true,
        "hue" : 4.5,
        "saturation" : 0.5
      }
    },
    "audio" : [ {
      "id" : 0,
      "pid" : 661,
      "format" : "AES3",
      "rate" : 48000,
      "channels" : 2,
      "filters" : {
        "volumeGain" : 2.5
      },
      "passthrough" : false
    } ],
    "captions" : [ {
      "id" : 0,
      "pid" : 662,
      "format" : "CEA_708_SMPTE_2038"
    } ],
    "metadata" : [ {
```

```
        "id" : 0,
        "pid" : 663,
        "format" : "UNKNOWN"
      } ]
    } ],
    "outputs" : [ {
      "streamType" : "MPEG_TS",
      "id" : 0,
      "name" : "Output Stream 1",
      "sink" : {
        "address" : {
          "url" : "srt://0.0.0.0:6001:lo",
          "srt" : {
            "mode" : "LISTENER",
            "overhead" : 20,
            "encryption" : "AES_256",
            "passphrase" : "0123456789",
            "redundancy" : [ "srt://127.0.0.1:5002:lo" ]
          }
        }
      },
      "ts" : {
        "service" : {
          "name" : "SERVICE",
          "providerName" : "PROVIDER"
        },
        "pcrStreamLocation" : "SEPARATE",
        "ebpLength" : 5,
        "ebpLengthUnit" : "SECONDS",
        "bitrate" : 11000000,
        "outputComplianceWith" : "STANDARD"
      },
      "video" : {
        "pid" : 670,
        "format" : "H264",
        "width" : 720,
        "height" : 576,
        "sampling" : "4:2:0",
        "depth" : 8,
        "scanRateType" : "SAME_AS_INPUT",
        "scan" : "INTERLACED",
        "fieldOrder" : "TOP_FIELD_FIRST",
        "h264" : {
          "coding" : "CABAC",
          "profile" : "HIGH",
          "gopSize" : 120,
          "bCount" : 0
        },
        "bitrate" : 8000000,
        "captions" : {
          "reference" : {
            "inputId" : 0,
            "id" : 0
          }
        },
        "crop" : {
          "fit" : "PAD",
          "from" : [ 0, 10 ],
          "to" : [ -1, 1070 ]
        },
        "logo" : {
          "id" : 0,
          "horizontalPosition" : 5,
          "verticalPosition" : 10,
          "horizontalScaling" : 150.0,
          "verticalScaling" : 200.0
        }
      },
      "audio" : [ {
        "id" : 0,
```

```
      "pid" : 671,
      "format" : "AAC_ADTS",
      "rate" : 48000,
      "name" : "Output audio 1",
      "bitrate" : 1000000,
      "channels" : 2,
      "references" : [ {
        "inputId" : 0,
        "id" : 0,
        "channel" : 0
      } ]
    } ],
    "captions" : [ {
      "id" : 0,
      "pid" : 672,
      "format" : "CEA_708_SMPTE_2038",
      "reference" : {
        "inputId" : 0,
        "id" : 0
      }
    } ],
    "metadata" : [ {
      "id" : 0,
      "pid" : 673,
      "format" : "UNKNOWN",
      "reference" : {
        "inputId" : 0,
        "id" : 0
      }
    } ]
  } ],
  "uploadedLogos" : [ {
    "id" : 0,
    "name" : "logo.png",
    "codestream" : "iVBORw0KGgoAAAANSUhEUgAAAAEAAAABCAIAAACQd1PeAAAAD0lEQVQIHQEEAPv/AP///wX+Av4DfRnGAAAAAElFTkSuQmCC"
  } ],
  "_links" : {
    "self" : {
      "href" : "https://127.0.0.1/api/pipeline/0"
    },
    "state" : {
      "href" : "https://127.0.0.1/api/pipeline/0/state"
    }
  }
}
```


Input Stream

JSON object containing configuration of input stream.

Path	Type	Description
`id`	`Number`	Unique identifier of the input stream, referenced in output definition.
`source.address`	`Object`	Input stream source address.

Path	Type	Description
source.address.url	String	URL string value. It should be in format "rtp://[<mc-src-ip-address>@]<ip-address>:<port>:<interface-id>" for RTP/UDP stream, "udp://[<mc-src-ip-address>@]<ip-address>:<port>:<interface-id>" for UDP stream, "srt://<ip-address>:<port>:<interface-id>" for SRT stream, "hls://<http|https url>" for HLS stream, "sdi://<sdi-device-id>:<port-index>" for SDI stream, "ndi://<ndi-name>" for NDI stream or "file://<file>" for transcoder system local file. The <interface-id> should be the identifier of one of the available network interfaces. The <sdi-device-id> should be the identifier of one of the available SDI devices. Examples: "rtp://127.0.0.1:5000:lo", "udp://127.0.0.1:5001:lo", "sdi://aja-sdi-0:0" or "file:///tmp/input.ts".
source.address.srt	Object	Specific information for SRT protocol.
source.address.srt.mode	String	Mode of SRT connection. Allowed values are "CALLER" or "LISTENER".
source.address.srt.latency	Integer	Buffer time to cover delivering and possible retransmitting of packets in msec. Allowed values [10, 2000].
source.address.srt.streamID	String	Stream ID is type of information that can be interchanged when a connection is being established. Can be used in a caller-listener connection layout. Allowed string has printable character with length: [0-512] set on the caller side. To disable this functionality leave this field empty.
source.address.srt.passphrase	String	Passphrase for encrypted SRT connection. Allowed phrase has printable character with length: [10-79]. To disable encryption leave this field empty.
source.address.srt.redundancy	Array	Optional list of URL string values representing redundant paths for SRT caller.
source.address.ndi	Object	Specific information for NDI protocol.
source.address.ndi.ndiGroup	String	Optional group name of NDI.
source.address.hls	Object	Specific information for HLS protocol.
source.address.hls.bufferSize	Integer	Optional size of buffer for HLS segments (in seconds). Decoding starts when the buffer is filled. Possible values are [0-999]. For input address only.

Path	Type	Description
source.streamType	String	Type of input stream. Allowed values are "MPEG_TS", "SDI", "NDI" or "HLS".
source.complianceWith	String	Shall be present when streamType is "MPEG_TS". Allowed values are "AUTO" (to allow any auto-detectable stream) or "EVERTZ" (to allow only Evertz stream) or "MEDIALINKS" (to allow only MediaLinks stream).
source.addressPCR	Object	Optionally can be present when streamType is "MPEG_TS". Input stream to receive PCR from.
source.addressPCR.url	String	It should be in format "rtp://[<mc-src-ip-address>@]<ip-address>:<port>:<interface-id>" for RTP/UDP stream, "udp://[<mc-src-ip-address>@]<ip-address>:<port>:<interface-id>" for UDP stream, "srt://<ip-address>:<port>:<interface-id>" for SRT stream. The <interface-id> should be the identifier of one of the available network interfaces. Examples: "rtp://127.0.0.1:5000:lo" or "udp://127.0.0.1:5001:lo".
source.addressPCR.srt	Object	Specific information for SRT protocol.
source.addressPCR.srt.mode	String	Mode of SRT connection. Allowed values are "CALLER" or "LISTENER".
source.addressPCR.srt.latency	Integer	Buffer time to cover delivering and possible retransmitting of packets in msec. Allowed values [10, 2000].
source.addressPCR.srt.passphrase	String	Passphrase for encrypted SRT connection. Allowed phrase has printable character with length: [10-79]. To disable encryption leave this field empty.
video	Input Video	Configuration of the input video. See Input Video.
audio	Array of Input Audios	Configuration of the input audio. See Input Audio.
captions	Array of Input Captions	Configuration of the input captions. See Input Captions.
metadata	Array of Input Metadata	Configuration of the input metadata. See Input Metadata.

Input stream example:

```
{
  "id" : 0,
  "source" : {
    "streamType" : "MPEG_TS",
    "address" : {
      "url" : "srt://127.0.0.1:5001:lo",
      "srt" : {
        "mode" : "CALLER",
        "latency" : 50,
        "streamID" : "#!::u=admin,t=file,m=publish,r=results.csv",
        "passphrase" : "",
        "redundancy" : [ "srt://127.0.0.1:5002:lo" ]
      }
    },
    "complianceWith" : "AUTO",
    "addressPCR" : {
      "url" : "srt://0.0.0.0:5000:lo",
      "srt" : {
        "mode" : "LISTENER",
        "latency" : 50,
        "passphrase" : ""
      }
    }
  },
  "video" : {
    "format" : "J2K",
    "width" : 1920,
    "height" : 1080,
    "sampling" : "4:2:2",
    "depth" : 10,
    "scanRate" : "50",
    "scan" : "INTERLACED",
    "fieldOrder" : "TOP_FIELD_FIRST",
    "filters" : {
      "yOffset" : 16,
      "yGain" : 0.2,
      "rOffset" : 1,
      "rGain" : 0.01,
      "gOffset" : 2,
      "gGain" : 0.02,
      "bOffset" : 3,
      "bGain" : 0.03,
      "rgbClip" : true,
      "hue" : 4.5,
      "saturation" : 0.5
    }
  },
  "audio" : [ {
    "id" : 0,
    "pid" : 661,
    "format" : "AES3",
    "rate" : 48000,
    "channels" : 2,
    "filters" : {
      "volumeGain" : 2.5
    },
    "passthrough" : false
  } ],
  "captions" : [ {
    "id" : 0,
    "pid" : 662,
    "format" : "CEA_708_SMPTE_2038"
  } ],
  "metadata" : [ {
    "id" : 0,
    "pid" : 663,
    "format" : "UNKNOWN"
  } ]
}
```

Input Video

JSON object containing configuration of input video.

Path	Type	Description
`format`	`String`	Format of input video. Allowed values are `"J2K"`, `"H262"`, `"H264"` or `"H265"`.
`width`	`Number`	Video frame width.
`height`	`Number`	Video frame height.
`sampling`	`String`	Video frame subsampling. Allowed values are `"4:2:2"`, `"4:2:0"` or `"4:4:4"`.
`depth`	`Number`	Video frame samples bit-depth.
`scanRate`	`String`	Frame rate for progressive scan video (frames per second) or field rate for interlaced scan video (fields per second). String value should be in format `"<value>/<factor>"` or `"<value>"`. Examples: `"24000/1001"` or `"24"`.
`scan`	`String`	Allowed values are `"PROGRESSIVE"` (for progressive scan video) or `"INTERLACED"` (for interlaced scan video).
`fieldOrder`	`String`	Allowed values are `"TOP_FIELD_FIRST"` or `"BOTTOM_FIELD_FIRST"`.
`filters`	`Object`	Video filters to be applied to decoded video. Modifications in this object can be applied at runtime and thus it will not force the pipeline to be restarted.
`filters.yOffset`	`Number`	Y sample value to be added to all decoded sample values. Examples: `16` to add `16` to all samples in Y component, `-8` to subtract `8` from all samples in Y component.
`filters.yGain`	`Number`	Ratio for Y color component to modify all decoded sample values. Examples: `0.2` to increase all samples in Y component by `20%`, `-0.1` to decrease all samples in Y component by `10%`.
`filters.rOffset`	`Number`	R sample value to be added to all decoded sample values. Examples: `1` to add `1` to all samples in R component, `-2` to subtract `2` from all samples in R component.

Path	Type	Description
`filters.rGain`	`Number`	Ratio for R color component to modify all decoded sample values. Examples: `0.01` to increase all samples in R component by `1%`, `-0.2` to decrease all samples in R component by `20%`.
`filters.gOffset`	`Number`	G sample value to be added to all decoded sample values. Examples: `2` to add `2` to all samples in G component, `-4` to subtract `4` from all samples in G component.
`filters.gGain`	`Number`	Ratio for G color component to modify all decoded sample values. Examples: `0.02` to increase all samples in G component by `2%`, `-0.3` to decrease all samples in G component by `30%`.
`filters.bOffset`	`Number`	B sample value to be added to all decoded sample values. Examples: `3` to add `3` to all samples in B component, `-8` to subtract `8` from all samples in B component.
`filters.bGain`	`Number`	Ratio for B color component to modify all decoded sample values. Examples: `0.03` to increase all samples in B component by `3%`, `-0.4` to decrease all samples in B component by `40%`.
`filters.rgbClip`	`Boolean`	Specifies whether RGB clipping should take place. Disabled by default.
`filters.hue`	`Number`	Number of degrees to change the color hue. Examples: `4.5` to increase color hue by `4.5` degrees, `-2.5` to decrease color hue by `2.5` degrees.
`filters.saturation`	`Number`	Ratio to change the color saturation. Examples: `0.5` to increase color saturation by `50%`, `-0.25` to decrease color saturation by `25%`.

Input video example:

```
{
  "format" : "J2K",
  "width" : 1920,
  "height" : 1080,
  "sampling" : "4:2:2",
  "depth" : 10,
  "scanRate" : "50",
  "scan" : "INTERLACED",
  "fieldOrder" : "TOP_FIELD_FIRST",
  "filters" : {
    "yOffset" : 16,
    "yGain" : 0.2,
    "rOffset" : 1,
    "rGain" : 0.01,
    "gOffset" : 2,
    "gGain" : 0.02,
    "bOffset" : 3,
    "bGain" : 0.03,
    "rgbClip" : true,
    "hue" : 4.5,
    "saturation" : 0.5
  }
}
```

Input Audios

JSON array of input audios.

Path	Type	Description
`[].id`	`Number`	Unique identifier of input audio.
`[].pid`	`Number`	Elementary stream identifier.
`[].format`	`String`	Format of input audio. Allowed values are `"AC3"`, `"EAC3"`, `"AES3"`, `"AAC_ADTS"`, `"AAC_LATM"`, `"MPEG2"` or `"RAW_FLTP"`.
`[].rate`	`Number`	Audio sampling rate in Hz.
`[].channels`	`Number`	Number of audio channels.
`[].filters`	`Object`	Audio filters to be applied to decoded audio. Modifications in this object can be applied at runtime and thus it will not force the pipeline to be restarted.
`[].filters.volumeGain`	`Number`	Ratio to change the audio volume. Examples: `0.5` to increase audio volume by `50%`, `-0.25` to decrease audio volume by `25%`.
`[].passthrough`	`Boolean`	Specifies that audio is sent pass-through (if possible).

Input audios example:

```
[ {
  "id" : 0,
  "pid" : 661,
  "format" : "AES3",
  "rate" : 48000,
  "channels" : 2,
  "filters" : {
    "volumeGain" : 2.5
  },
  "passthrough" : false
} ]
```

Input Captions

JSON array of input captions.

Path	Type	Description
`[].id`	`Number`	Unique identifier of input captions.
`[].pid`	`Number`	Elementary stream identifier.
`[].format`	`String`	Format of input captions. Allowed values are `"CEA_608_SMPTE_2038"`, `"CEA_708_SMPTE_2038"`, `"CEA_608_SDI"` and `"CEA_708_SDI"`.

Input captions example:

```
[ {
  "id" : 0,
  "pid" : 662,
  "format" : "CEA_708_SMPTE_2038"
} ]
```

Input Metadata

JSON array of input metadata.

Path	Type	Description
`[].id`	`Number`	Unique identifier of input metadata.
`[].pid`	`Number`	Elementary stream identifier.
`[].format`	`String`	Format of input metadata. Allowed values are `"UNKNOWN"`.

Input metadata example:

```
[ {
  "id" : 0,
  "pid" : 663,
  "format" : "UNKNOWN"
} ]
```

Output Stream

JSON object containing configuration of output stream.

Path	Type	Description
`id`	`Number`	Unique identifier of output stream.
`name`	`String`	Name of this output stream displayed in UI.
`streamType`	`String`	Type of output stream. Allowed values are `"MPEG_TS"` or `"RTMP"`.
`sink.address`	`Array|Object`	Output stream destination address(es).
`sink.address.srt`	`Object`	Specific information for SRT protocol.
`sink.address.url`	`String`	URL string value. It should be in format `"rtp://<ip-address>:<port>:<interface-id>"` for RTP/UDP stream, `"udp://<ip-address>:<port>:<interface-id>"` for UDP stream, `"srt://<ip-address-or-web-url>:<port>:<interface-id>"` for SRT stream, `"rtmp(s)://<domain-name/ip-address>(:<port>)/path/streaming-key"` for RTMP(S) stream, `"sdi://<sdi-device-id>:<port-index>"` for SDI stream , `\"ndi://<ndi-name>\"` for NDI stream or `"file://<file>"` for transcoder system local file. The `<interface-id>` should be the identifier of one of the available network interfaces. The `<sdi-device-id>` should be the identifier of one of the available SDI devices. Examples: `"rtp://127.0.0.1:5000:lo"`, `"udp://127.0.0.1:5001:lo"`, `"rtmp[s]://[user:password@]example.com:1935/live/abc-123"`, `"sdi://aja-sdi-0:0"` or `"file:///tmp/input.ts"`.
`sink.address.srt.mode`	`String`	Mode of SRT connection. Allowed values are `"CALLER"` or `"LISTENER"`.
`sink.address.srt.overhead`	`Integer`	Bandwidth overhead above output rate in percents. Allowed values: [5, 100].
`sink.address.srt.encryption`	`String`	Settings of key length for AES encryption. Allowed values are `"AES_DEFAULT"`, `"AES_128"`, `"AES_192"` or `"AES_256"`.
`sink.address.srt.passphrase`	`String`	Passphrase for encrypted SRT connection. Allowed phrase has printable character with length: [10-79]. To disable encryption leave this field empty.
`sink.address.srt.redundancy`	`Array`	Optional list of URL string values representing redundant paths for SRT caller.
`sink.address.ndi`	`Object`	Specific information for NDI protocol.

Path	Type	Description
`sink.address.ndi.ndiGroup`	`String`	Optional group name of NDI.
`sink.address.hls`	`Object`	Specific information for HLS protocol.
`sink.address.hls.segmentLength`	`Integer`	Specify duration of the HLS segments (in seconds). Possible values are [1-300]. For output address only. Default is 6.
`sink.address.hls.segmentCount`	`Integer`	Specify how many HLS segments are playable in history. Possible values are [3-600]. For output address only. Default is 10.
`sink.address.hls.deleteThreshold`	`Integer`	Specify how many HLS segments are available after playlist dereference. Possible values are [1-300]. For output address only. Default is 1 (nothing available).
`ts`	`Object`	Specific information for MPEG_TS output stream type.
`ts.service`	`Object`	Information about service to be embedded into SDT (Service Description Table) inside MPEG-TS.
`ts.service.name`	`String`	Name of the service to be embedded into SDT (Service Description Table) inside MPEG-TS.
`ts.service.providerName`	`String`	Name of the service provider to be embedded into SDT (Service Description Table) inside MPEG-TS.
`ts.bitrate`	`Number`	Maximum total bitrate of output stream in bits per second.
`ts.pcrStreamLocation`	`String`	Placement of stream with PCR.
`ts.ebpLength`	`Number`	Encoder boundary point in seconds or multiples of GOP.
`ts.ebpLengthUnit`	`String`	Unit of EBP length. Can be in seconds or in multiples of GOP. Allowed values are `"SECONDS"` or `"GOPS"`.
`ts.outputComplianceWith`	`String`	MPEG-TS standard or Evertz compliance. Allowed values are `"STANDARD"` or `"EVERTZ"`.
`video`	`Output Video`	Configuration of output video. See Output Video.
`audio`	`Array of Output Audio`	Configuration of output audio. See Output Audio.
`captions`	`Array of Output Captions`	Configuration of output captions. See Output Captions.
`metadata`	`Array of Output Metadata`	Configuration of output metadata. See Output Metadata.

Output stream example:

```
{
  "streamType" : "MPEG_TS",
  "id" : 0,
  "name" : "Output Stream 1",
  "sink" : {
    "address" : {
      "url" : "srt://0.0.0.0:6001:lo",
      "srt" : {
        "mode" : "LISTENER",
        "overhead" : 20,
        "encryption" : "AES_256",
        "passphrase" : "0123456789",
        "redundancy" : [ "srt://127.0.0.1:5002:lo" ]
      }
    }
  },
  "ts" : {
    "service" : {
      "name" : "SERVICE",
      "providerName" : "PROVIDER"
    },
    "pcrStreamLocation" : "SEPARATE",
    "ebpLength" : 5,
    "ebpLengthUnit" : "SECONDS",
    "bitrate" : 11000000,
    "outputComplianceWith" : "STANDARD"
  },
  "video" : {
    "pid" : 670,
    "format" : "H264",
    "width" : 720,
    "height" : 576,
    "sampling" : "4:2:0",
    "depth" : 8,
    "scanRateType" : "SAME_AS_INPUT",
    "scan" : "INTERLACED",
    "fieldOrder" : "TOP_FIELD_FIRST",
    "h264" : {
      "coding" : "CABAC",
      "profile" : "HIGH",
      "gopSize" : 120,
      "bCount" : 0
    },
    "bitrate" : 8000000,
    "captions" : {
      "reference" : {
        "inputId" : 0,
        "id" : 0
      }
    },
    "crop" : {
      "fit" : "PAD",
      "from" : [ 0, 10 ],
      "to" : [ -1, 1070 ]
    },
    "logo" : {
      "id" : 0,
      "horizontalPosition" : 5,
      "verticalPosition" : 10,
      "horizontalScaling" : 150.0,
      "verticalScaling" : 200.0
    }
  },
  "audio" : [ {
    "id" : 0,
    "pid" : 671,
    "format" : "AAC_ADTS",
    "rate" : 48000,
    "name" : "Output audio 1",
```

```
      "bitrate" : 1000000,
      "channels" : 2,
      "references" : [ {
        "inputId" : 0,
        "id" : 0,
        "channel" : 0
      } ]
    } ],
    "captions" : [ {
      "id" : 0,
      "pid" : 672,
      "format" : "CEA_708_SMPTE_2038",
      "reference" : {
        "inputId" : 0,
        "id" : 0
      }
    } ],
    "metadata" : [ {
      "id" : 0,
      "pid" : 673,
      "format" : "UNKNOWN",
      "reference" : {
        "inputId" : 0,
        "id" : 0
      }
    } ]
  }
```

Output Video

JSON object containing configuration of output video.

Path	Type	Description
`pid`	`Number`	Elementary stream identifier.
`format`	`String`	Format of output video. Allowed values are `"J2K"`, `"H262"`, `"H264"` or `"H265"`.
`width`	`Number`	Video frame width.
`height`	`Number`	Video frame height.
`crop`	`Object`	Configuration of cropping and/or padding.

Path	Type	Description
`crop.fit`	`String`	Allowed values are `"SCALE"`, `"CROP"` or `"PAD"`. Use `"SCALE"` to resize configured input frame size to configured output frame size and change aspect ratio when it differs. Use `"CROP"` to resize configured input frame size to configured output frame size but keep the input frame aspect ratio. The input will be automatically horizontally or vertically cropped. Use `"PAD"` to resize configured input frame size to configured output frame size but keep the input frame aspect ratio. The input will be automatically expanded with horizontal or vertical black bars.
`crop.from`	`Array`	Array with X and Y offset in input video frame which specifies the first left/top most input pixel to be included in output frame. It can point outside of input frame area to create black bars. Examples: `[10,20]` to crop `10` pixels from left and `20` pixels from top, `[-10,-20]` to create `10` pixels wide black bar at left and `20` pixels high black bar at top in input video frame.
`crop.to`	`Array`	Array with X and Y offset in input video frame which specifies the first right/bottom most input pixel to not be included in output frame. It can point outside of input frame area to create black bars. Examples: `[1910,1060]` to crop `10` pixels from right and `20` pixels from bottom in `1920x1080` input video frame, `[1930,1100]` to create `10` pixels wide black bar at right and `20` pixels high black bar at bottom in `1920x1080` input video frame.
`logo`	`Object`	Configuration of logo placement.
`logo.id`	`Number`	ID of the logo present in **uploadedLogos** section.
`logo.horizontalPosition`	`Number`	Horizontal position of the logo in number of pixels from left side.
`logo.verticalPosition`	`Number`	Vertical position of the logo in number of pixels from top side.
`logo.horizontalScaling`	`Number`	Horizontal scaling of the logo in percents of original logo width.
`logo.verticalScaling`	`Number`	Vertical scaling of the logo in percents of original logo height.

Path	Type	Description
`sampling`	`String`	Video frame subsampling. Allowed values are `"4:2:2"`, `"4:2:0"` or `"4:4:4"`.
`depth`	`Number`	Video frame samples bit-depth.
`scanRateType`	`String`	Scan rate type can be set to SAME_AS_INPUT, HALF_THE_INPUT and CONVERT_TO. I takes the input scan rate and sets the output scan rate based on this option.
`scan`	`String`	Allowed values are `"PROGRESSIVE"` (for progressive scan video) or `"INTERLACED"` (for interlaced scan video).
`fieldOrder`	`String`	Allowed values are `"TOP_FIELD_FIRST"` or `"BOTTOM_FIELD_FIRST"`.
`bitrate`	`Number`	Maximum bitrate of output video in bits per second. Value `"0"` enables video passthrough (then output video parameters have to exactly match input video parameters).
`captions`	`Object`	Identifier of input captions to be wrapped inside output video.
`captions.reference`	`Object`	Referenced input captions.
`captions.reference.inputId`	`Number`	Identifier of referenced input stream with captions.
`captions.reference.id`	`Number`	Identifier of referenced input captions.
`h262`	`Object`	Encoding settings for H.262 video. Valid only when `format` is `"H262"`.
`h262.profile`	`String`	Allowed values are `"HIGH"`, `"MAIN"` or `"SIMPLE"`.
`h262.gopSize`	`Number`	GOP size in number of frames. Allowed values are `12` - `300`.
`h262.bCount`	`Number`	Number of B frames. Allowed values are `0` - `4`.
`h264`	`Object`	Encoding settings for H.264 video. Valid only when `format` is `"H264"`.
`h264.coding`	`String`	Allowed values are `"CABAC"` or `"CAVLC"`.
`h264.profile`	`String`	Allowed values are `"HIGH"`, `"MAIN"` or `"BASELINE"`.

Path	Type	Description
`h264.gopSize`	`Number`	GOP size in number of frames. Allowed values are `25` - `300`.
`h264.bCount`	`Number`	Number of B frames. Allowed values are `0` - `4`.

Output video example:

```
{
  "pid" : 670,
  "format" : "H264",
  "width" : 720,
  "height" : 576,
  "sampling" : "4:2:0",
  "depth" : 8,
  "scanRateType" : "SAME_AS_INPUT",
  "scan" : "INTERLACED",
  "fieldOrder" : "TOP_FIELD_FIRST",
  "h264" : {
    "coding" : "CABAC",
    "profile" : "HIGH",
    "gopSize" : 120,
    "bCount" : 0
  },
  "bitrate" : 8000000,
  "captions" : {
    "reference" : {
      "inputId" : 0,
      "id" : 0
    }
  },
  "crop" : {
    "fit" : "PAD",
    "from" : [ 0, 10 ],
    "to" : [ -1, 1070 ]
  },
  "logo" : {
    "id" : 0,
    "horizontalPosition" : 5,
    "verticalPosition" : 10,
    "horizontalScaling" : 150.0,
    "verticalScaling" : 200.0
  }
}
```

Output Audios

JSON array of output audios.

Path	Type	Description
`[]`	`Array`	List of output audios.
`[].id`	`Number`	Unique identifier of output audio.
`[].name`	`String`	Name of this output audio displayed in UI.
`[].pid`	`Number`	Elementary stream identifier.

Path	Type	Description
`[].format`	`String`	Format of output audio. Allowed values are `"AC3"`, `"AES3"`, `"AAC_ADTS"`, `"AAC_LATM"`, `"MPEG2"` or `"RAW_FLTP"`.
`[].rate`	`Number`	Audio sampling rate in Hz.
`[].references[]`	`Array`	Array of output channels referencing channels from input audio.
`[].references[].inputId`	`Number`	Identifier of referenced input stream with audio.
`[].references[].id`	`Number`	Identifier of referenced input audio.
`[].references[].channel`	`Number`	Index of channel from referenced input audio.
`[].channels`	`Number`	Number of output audio channels which shall be down-mixed/up-mixed from the referenced input channels. Omit the field when down-mixing/up-mixing shall not be performed or use `2` for stereo, `6` for 5.1 and `8` for 7.1.
`[].bitrate`	`Number`	Maximum bitrate of output audio in bits per second.

Output audios example:

```
[ {
  "id" : 0,
  "pid" : 671,
  "format" : "AAC_ADTS",
  "rate" : 48000,
  "name" : "Output audio 1",
  "bitrate" : 1000000,
  "channels" : 2,
  "references" : [ {
    "inputId" : 0,
    "id" : 0,
    "channel" : 0
  } ]
} ]
```

Output Captions

JSON array of output captions.

Path	Type	Description
`[]`	`Array`	List of output captions.
`[].id`	`Number`	Unique identifier of output captions.
`[].pid`	`Number`	Elementary stream identifier.
`[].format`	`String`	Format of output captions. Allowed values are `"CEA_608_SMPTE_2038"`, `"CEA_708_SMPTE_2038"`, `"CEA_608_SDI"` and `"CEA_708_SDI"`.

Path	Type	Description
`[].reference`	`Object`	Referenced input captions.
`[].reference.inputId`	`Number`	Identifier of referenced input stream with captions.
`[].reference.id`	`Number`	Identifier of referenced input captions.

Output captions example:

```
[ {
  "id" : 0,
  "pid" : 672,
  "format" : "CEA_708_SMPTE_2038",
  "reference" : {
    "inputId" : 0,
    "id" : 0
  }
} ]
```

Output Metadata

JSON array of output metadata.

Path	Type	Description
`[]`	`Array`	List of output metadata.
`[].id`	`Number`	Unique identifier of output metadata.
`[].pid`	`Number`	Elementary stream identifier.
`[].format`	`String`	Format of output metadata. Allowed values are `"UNKNOWN"`.
`[].reference`	`Object`	Referenced input metadata.
`[].reference.inputId`	`Number`	Identifier of referenced input stream with metadata.
`[].reference.id`	`Number`	Identifier of referenced input metadata.

Output metadata example:

```
[ {
  "id" : 0,
  "pid" : 673,
  "format" : "UNKNOWN",
  "reference" : {
    "inputId" : 0,
    "id" : 0
  }
} ]
```

Uploaded Logos

JSON array of uploaded logos.

Path	Type	Description
`id`	`Number`	Unique identifier of uploaded logo.
`name`	`String`	Original file name of the logo.
`codestream`	`String`	PNG file content in Base64

Uploaded logos example:


```
{
  "id" : 0,
  "name" : "logo.png",
  "codestream" : "iVBORw0KGgoAAAANSUhEUgAAAAEAAAABCAIAAACQd1PeAAAAD0lEQVQIHQEEAPv/AP///wX+Av4DfRnGAAAAAElFTkSuQmCC"
}
```


Pipeline State

JSON object containing runtime state of a running pipeline.

Path	Type	Description
`id`	`Number`	Unique identifier of this pipeline.
`status`	`String`	Allowed values are `"ERROR"`, `"WARNING"`, `"OK"` or `"UNKNOWN"`. Values describes worst state of input stream part. In case of `"UNKNOWN"` It is equal to operationState: starting, stopping, restarting, not started. To distinguish state on different input stream see inputs.
`operationState`	`String`	Allowed values are `"NOT_STARTED"`, `"STARTED"`, `"FAILED"` (pipeline crashed), `"STOPPING"` (pipeline is being stopped), `"STARTING"` (pipeline is being started) or `"RESTARTING"` (pipeline is being restarted).
`licenseStatus`	`String`	Allowed values are `"BEING_CHECKED"`, `"BEING_RETURNED"`, `"GRANTED"`, `"DENIED_BLOCKED"`, `"DENIED_INVALID"`, `"EMERGENCY_LICENSE_SERVER"` (license servers connection problems), `"EMERGENCY_TRANSCODER"` (connection to license server broken) or `"UNKNOWN"`.
`licenseExpirationDate`	`String`	ISO 8601 date when license for pipeline expires on license server.
`licenseRemainingTime`	`String`	ISO 8601 time before licenseStatus changes automatically on transcoder.
`name`	`String`	Name of this pipeline.
`uniqueId`	`String`	Unique pipeline UUID. Randomly generated when a user creates the pipeline, removed with the pipeline. Can't be downloaded with pipeline preset. Survives stopping of pipeline and restart of service/machine.

Path	Type	Description
`inputs`	`Array of Input Stream States`	Array of Input Stream States. See Input Stream State
`outputs`	`Array of Output Stream States`	Array of Output Stream States. See Output Stream State

Relation	Description
`self`	Link to current state of this pipeline.
`pipeline`	Link to this pipeline.

Pipeline state example:

```
{
  "id" : 0,
  "status" : "ERROR",
  "operationState" : "STARTED",
  "licenseStatus" : "GRANTED",
  "licenseExpirationDate" : "1970-01-01T00:00:00Z",
  "licenseRemainingTime" : "00:00:00",
  "name" : "Pipeline 1",
  "uniqueId" : "577b54da-5dfe-4bea-add6-b689e8cd7eea",
  "inputs" : [ {
    "id" : 0,
    "status" : "ERROR",
    "name" : "rtp://0.0.0.0:5000:lo",
    "description" : "JPEG2000 1920x1080 4:4:4 10-bit",
    "elementaryStreams" : [ {
      "id" : 0,
      "pid" : 501,
      "type" : "UNKNOWN",
      "state" : "NEW",
      "elementaryStream" : {
        "messages" : [ ],
        "state" : "NEW",
        "pid" : 501,
        "id" : 0,
        "streamDescription" : "Unknown Stream (pid: 501)"
      }
    }, {
      "id" : 0,
      "pid" : 502,
      "type" : "VIDEO",
      "state" : "NEW",
      "format" : "J2K",
      "width" : 1920,
      "height" : 1080,
      "depth" : 10,
      "sampling" : "4:2:2",
      "scanRate" : "60000/1001",
      "scan" : "INTERLACED",
      "fieldOrder" : "BOTTOM_FIELD_FIRST",
      "elementaryStream" : {
        "messages" : [ ],
        "state" : "NEW",
        "pid" : 502,
        "id" : 0,
        "format" : "J2K",
        "width" : 1920,
        "height" : 1080,
        "depth" : 10,
        "sampling" : "S422",
        "scanRate" : "60000/1001",
        "scan" : "INTERLACED",
        "fieldOrder" : "BOTTOM_FIELD_FIRST",
        "streamDescription" : "Video Stream (pid: 502, JPEG2000 1920x1080i (bottom-first) @59.94Hz 4:2:2 10-bit)"
      },
      "messages" : [ ]
    }, {
      "id" : 0,
      "pid" : 503,
      "type" : "AUDIO",
      "state" : "NEW",
      "format" : "AES3",
      "channelCount" : 2,
      "rate" : 48000,
      "bitrate" : 235000,
      "elementaryStream" : {
        "messages" : [ {
          "severity" : "ERROR",
          "message" : "${stream} wasn't detected in source stream."
        } ],
```

```
      "state" : "NEW",
      "pid" : 503,
      "id" : 0,
      "format" : "AES3",
      "channelCount" : 2,
      "rate" : 48000,
      "bitrate" : 235000,
      "bitsPerSample" : 0,
      "streamDescription" : "Audio Stream (pid: 503, AES3 2-channels 48000 Hz 235 kb/s)"
    },
    "messages" : [ {
      "severity" : "ERROR",
      "message" : "This audio stream wasn't detected in source stream."
    } ]
  }, {
    "id" : 1,
    "pid" : 499,
    "type" : "AUDIO",
    "state" : "NEW",
    "format" : "AES3_DOLBY_E",
    "channelCount" : 2,
    "rate" : 48000,
    "bitrate" : 128000,
    "elementaryStream" : {
      "messages" : [ {
        "severity" : "ERROR",
        "message" : "${stream} wasn't detected in source stream."
      } ],
      "state" : "NEW",
      "pid" : 499,
      "id" : 1,
      "format" : "AES3_DOLBY_E",
      "channelCount" : 2,
      "rate" : 48000,
      "bitrate" : 128000,
      "bitsPerSample" : 0,
      "streamDescription" : "Audio Stream (pid: 499, AES3_DOLBY_E 2-channels 48000 Hz 128 kb/s)"
    },
    "messages" : [ {
      "severity" : "ERROR",
      "message" : "This audio stream wasn't detected in source stream."
    } ]
  }, {
    "id" : 0,
    "pid" : 504,
    "type" : "CAPTIONS",
    "state" : "NEW",
    "format" : "CEA_708_SMPTE_2038",
    "elementaryStream" : {
      "messages" : [ ],
      "state" : "NEW",
      "pid" : 504,
      "id" : 0,
      "format" : "CEA_708_SMPTE_2038",
      "streamDescription" : "Captions Stream (pid: 504, CEA-708 in SMPTE-2038)"
    },
    "messages" : [ ]
  }, {
    "id" : 0,
    "pid" : 505,
    "type" : "METADATA",
    "state" : "NEW",
    "format" : "UNKNOWN",
    "info" : "INFO1",
    "elementaryStream" : {
      "messages" : [ ],
      "state" : "NEW",
      "pid" : 505,
      "id" : 0,
      "format" : "UNKNOWN",
```

```
        "info" : "INFO1",
        "streamDescription" : "Metadata Stream (pid: 505, Unknown, info: INFO1)"
      },
      "messages" : [ ]
    }, {
      "id" : 1,
      "pid" : 506,
      "type" : "METADATA",
      "state" : "NEW",
      "format" : "SMPTE_2038",
      "info" : "BLABLA",
      "elementaryStream" : {
        "messages" : [ ],
        "state" : "NEW",
        "pid" : 506,
        "id" : 1,
        "format" : "SMPTE_2038",
        "info" : "BLABLA",
        "streamDescription" : "Metadata Stream (pid: 506, SMPTE-2038, info: BLABLA)"
      },
      "messages" : [ ]
    } ],
    "decodedFps" : 0.0,
    "decodingLatency" : 0.0,
    "invalidFps" : 0.0,
    "messages" : [ {
      "severity" : "ERROR",
      "message" : "rtp://0.0.0.0:5000:lo: input stream #0 error."
    } ]
  } ],
  "outputs" : [ {
    "id" : 0,
    "uniqueId" : "ddc2b880-d32c-4bae-afd1-b4c5b2b75d19",
    "status" : "ERROR",
    "name" : "Output Stream 1",
    "sink" : "rtp://127.0.0.1:50001",
    "description" : "H.264 1920x1080 4:2:0 8-bit",
    "encodingLatency" : 0.0,
    "transcodedFps" : 0.0,
    "transcodingLatency" : 0.0,
    "droppedFps" : 0.0,
    "messages" : [ {
      "severity" : "ERROR",
      "message" : "Output Stream 1: Not producing any output."
    } ],
    "latencyAddition" : 0
  } ],
  "_links" : {
    "self" : {
      "href" : "https://127.0.0.1/api/pipeline/0/state"
    },
    "pipeline" : {
      "href" : "https://127.0.0.1/api/pipeline/0"
    }
  }
}
```


Input Stream State

JSON object containing input stream state.

Path	Type	Description
`id`	`Number`	Unique identifier of this input stream.
`status`	`String`	Allowed values are **`"ERROR"`**, **`"WARNING"`**, **`"OK"`** or **`"UNKNOWN"`**.

Path	Type	Description
`name`	`String`	Name of this input stream.
`description`	`String`	Input stream description.
`elementaryStreams`	`Array of Elementary Streams`	Array of detected/configured elementary streams. See Elementary Stream.
`decodedFps`	`Number`	Number of decoded progressive frames/interlaced fields per second for this input stream (average over the last few seconds).
`decodingLatency`	`Number`	Decoding latency of progressive frame/interlaced field in milliseconds for this input stream (average over the last few seconds).
`invalidFps`	`Number`	Number of dropped progressive frames/interlaced fields per second for this input stream (average over the last few seconds).
`messages`	`Array`	Messages for this input stream.

Input Stream State example:

```
{
  "id" : 0,
  "status" : "ERROR",
  "name" : "rtp://0.0.0.0:5000:lo",
  "description" : "JPEG2000 1920x1080 4:4:4 10-bit",
  "elementaryStreams" : [ {
    "id" : 0,
    "pid" : 501,
    "type" : "UNKNOWN",
    "state" : "NEW",
    "elementaryStream" : {
      "messages" : [ ],
      "state" : "NEW",
      "pid" : 501,
      "id" : 0,
      "streamDescription" : "Unknown Stream (pid: 501)"
    }
  }, {
    "id" : 0,
    "pid" : 502,
    "type" : "VIDEO",
    "state" : "NEW",
    "format" : "J2K",
    "width" : 1920,
    "height" : 1080,
    "depth" : 10,
    "sampling" : "4:2:2",
    "scanRate" : "60000/1001",
    "scan" : "INTERLACED",
    "fieldOrder" : "BOTTOM_FIELD_FIRST",
    "elementaryStream" : {
      "messages" : [ ],
      "state" : "NEW",
      "pid" : 502,
      "id" : 0,
      "format" : "J2K",
      "width" : 1920,
      "height" : 1080,
      "depth" : 10,
      "sampling" : "S422",
      "scanRate" : "60000/1001",
      "scan" : "INTERLACED",
      "fieldOrder" : "BOTTOM_FIELD_FIRST",
      "streamDescription" : "Video Stream (pid: 502, JPEG2000 1920x1080i (bottom-first) @59.94Hz 4:2:2 10-bit)"
    },
    "messages" : [ ]
  }, {
    "id" : 0,
    "pid" : 503,
    "type" : "AUDIO",
    "state" : "NEW",
    "format" : "AES3",
    "channelCount" : 2,
    "rate" : 48000,
    "bitrate" : 235000,
    "elementaryStream" : {
      "messages" : [ {
        "severity" : "ERROR",
        "message" : "${stream} wasn't detected in source stream."
      } ],
      "state" : "NEW",
      "pid" : 503,
      "id" : 0,
      "format" : "AES3",
      "channelCount" : 2,
      "rate" : 48000,
      "bitrate" : 235000,
      "bitsPerSample" : 0,
      "streamDescription" : "Audio Stream (pid: 503, AES3 2-channels 48000 Hz 235 kb/s)"
```

```
    },
    "messages" : [ {
      "severity" : "ERROR",
      "message" : "This audio stream wasn't detected in source stream."
    } ]
  }, {
    "id" : 1,
    "pid" : 499,
    "type" : "AUDIO",
    "state" : "NEW",
    "format" : "AES3_DOLBY_E",
    "channelCount" : 2,
    "rate" : 48000,
    "bitrate" : 128000,
    "elementaryStream" : {
      "messages" : [ {
        "severity" : "ERROR",
        "message" : "${stream} wasn't detected in source stream."
      } ],
      "state" : "NEW",
      "pid" : 499,
      "id" : 1,
      "format" : "AES3_DOLBY_E",
      "channelCount" : 2,
      "rate" : 48000,
      "bitrate" : 128000,
      "bitsPerSample" : 0,
      "streamDescription" : "Audio Stream (pid: 499, AES3_DOLBY_E 2-channels 48000 Hz 128 kb/s)"
    },
    "messages" : [ {
      "severity" : "ERROR",
      "message" : "This audio stream wasn't detected in source stream."
    } ]
  }, {
    "id" : 0,
    "pid" : 504,
    "type" : "CAPTIONS",
    "state" : "NEW",
    "format" : "CEA_708_SMPTE_2038",
    "elementaryStream" : {
      "messages" : [ ],
      "state" : "NEW",
      "pid" : 504,
      "id" : 0,
      "format" : "CEA_708_SMPTE_2038",
      "streamDescription" : "Captions Stream (pid: 504, CEA-708 in SMPTE-2038)"
    },
    "messages" : [ ]
  }, {
    "id" : 0,
    "pid" : 505,
    "type" : "METADATA",
    "state" : "NEW",
    "format" : "UNKNOWN",
    "info" : "INFO1",
    "elementaryStream" : {
      "messages" : [ ],
      "state" : "NEW",
      "pid" : 505,
      "id" : 0,
      "format" : "UNKNOWN",
      "info" : "INFO1",
      "streamDescription" : "Metadata Stream (pid: 505, Unknown, info: INFO1)"
    },
    "messages" : [ ]
  }, {
    "id" : 1,
    "pid" : 506,
    "type" : "METADATA",
    "state" : "NEW",
```

```
      "format" : "SMPTE_2038",
      "info" : "BLABLA",
      "elementaryStream" : {
        "messages" : [ ],
        "state" : "NEW",
        "pid" : 506,
        "id" : 1,
        "format" : "SMPTE_2038",
        "info" : "BLABLA",
        "streamDescription" : "Metadata Stream (pid: 506, SMPTE-2038, info: BLABLA)"
      },
      "messages" : [ ]
    } ],
    "decodedFps" : 0.0,
    "decodingLatency" : 0.0,
    "invalidFps" : 0.0,
    "messages" : [ {
      "severity" : "ERROR",
      "message" : "rtp://0.0.0.0:5000:lo: input stream #0 error."
    } ]
  }
```


Elementary Streams

JSON array of elementary streams.

Path	Type	Description
`[].id`	`Number`	Unique identifier of this elementary stream in a set of elementary streams with the same **`type`**.
`[].type`	`String`	Type of elementary stream. Allowed values are **`"VIDEO"`**, **`"AUDIO"`**, **`"CAPTIONS"`**, **`"METADATA"`** or **`"UNKNOWN"`**.
`[].state`	`String`	Current state of elementary stream. Allowed values are **`"NEW"`** (newly detected elementary stream), **`"MATCHED"`** (configured elementary stream which was found) or **`"ERROR"`** (configured elementary stream which wasn't found).
`[].pid`	`Number`	Elementary stream identifier.
`[].format`	`String`	Format of video. Allowed values are **`"J2K"`**, **`"H262"`**, **`"H264"`** or **`"H265"`**.
`[].width`	`Number`	Video frame width.
`[].height`	`Number`	Video frame height.
`[].depth`	`Number`	Video frame samples bit-depth.
`[].sampling`	`String`	Video frame subsampling. Allowed values are **`"4:2:2"`**, **`"4:2:0"`** or **`"4:4:4"`**.

Path	Type	Description
`[].scanRate`	`String`	Frame rate for progressive scan video (frames per second) or field rate for interlaced scan video (fields per second). String value should be in format `"<value>/<factor>"` or `"<value>"`. Examples: `"24000/1001"` or `"24"`.
`[].scan`	`String`	Allowed values are `"PROGRESSIVE"` (for progressive scan video) or `"INTERLACED"` (for interlaced scan video).
`[].fieldOrder`	`String`	Allowed values are `"TOP_FIELD_FIRST"` or `"BOTTOM_FIELD_FIRST"`.
`[].channelCount`	`Number`	Number of audio channels.
`[].rate`	`Number`	Audio sampling rate in Hz.
`[].bitrate`	`Number`	Bitrate of audio in bits per second.
`[].info`	`String`	Metadata description.
`messages`	`Array`	Messages for this elementary stream.

Elementary streams of particular types (VIDEO, AUDIO, CAPTIONS, METADATA) have additional fields.

Elementary streams example:

```
[ {
  "id" : 0,
  "pid" : 501,
  "type" : "UNKNOWN",
  "state" : "NEW",
  "elementaryStream" : {
    "messages" : [ ],
    "state" : "NEW",
    "pid" : 501,
    "id" : 0,
    "streamDescription" : "Unknown Stream (pid: 501)"
  }
}, {
  "id" : 0,
  "pid" : 502,
  "type" : "VIDEO",
  "state" : "NEW",
  "format" : "J2K",
  "width" : 1920,
  "height" : 1080,
  "depth" : 10,
  "sampling" : "4:2:2",
  "scanRate" : "60000/1001",
  "scan" : "INTERLACED",
  "fieldOrder" : "BOTTOM_FIELD_FIRST",
  "elementaryStream" : {
    "messages" : [ ],
    "state" : "NEW",
    "pid" : 502,
    "id" : 0,
    "format" : "J2K",
    "width" : 1920,
    "height" : 1080,
    "depth" : 10,
    "sampling" : "S422",
    "scanRate" : "60000/1001",
    "scan" : "INTERLACED",
    "fieldOrder" : "BOTTOM_FIELD_FIRST",
    "streamDescription" : "Video Stream (pid: 502, JPEG2000 1920x1080i (bottom-first) @59.94Hz 4:2:2 10-bit)"
  },
  "messages" : [ ]
}, {
  "id" : 0,
  "pid" : 503,
  "type" : "AUDIO",
  "state" : "NEW",
  "format" : "AES3",
  "channelCount" : 2,
  "rate" : 48000,
  "bitrate" : 235000,
  "elementaryStream" : {
    "messages" : [ {
      "severity" : "ERROR",
      "message" : "${stream} wasn't detected in source stream."
    } ],
    "state" : "NEW",
    "pid" : 503,
    "id" : 0,
    "format" : "AES3",
    "channelCount" : 2,
    "rate" : 48000,
    "bitrate" : 235000,
    "bitsPerSample" : 0,
    "streamDescription" : "Audio Stream (pid: 503, AES3 2-channels 48000 Hz 235 kb/s)"
  },
  "messages" : [ {
    "severity" : "ERROR",
    "message" : "This audio stream wasn't detected in source stream."
  } ]
```

```
  }, {
    "id" : 1,
    "pid" : 499,
    "type" : "AUDIO",
    "state" : "NEW",
    "format" : "AES3_DOLBY_E",
    "channelCount" : 2,
    "rate" : 48000,
    "bitrate" : 128000,
    "elementaryStream" : {
      "messages" : [ {
        "severity" : "ERROR",
        "message" : "${stream} wasn't detected in source stream."
      } ],
      "state" : "NEW",
      "pid" : 499,
      "id" : 1,
      "format" : "AES3_DOLBY_E",
      "channelCount" : 2,
      "rate" : 48000,
      "bitrate" : 128000,
      "bitsPerSample" : 0,
      "streamDescription" : "Audio Stream (pid: 499, AES3_DOLBY_E 2-channels 48000 Hz 128 kb/s)"
    },
    "messages" : [ {
      "severity" : "ERROR",
      "message" : "This audio stream wasn't detected in source stream."
    } ]
  }, {
    "id" : 0,
    "pid" : 504,
    "type" : "CAPTIONS",
    "state" : "NEW",
    "format" : "CEA_708_SMPTE_2038",
    "elementaryStream" : {
      "messages" : [ ],
      "state" : "NEW",
      "pid" : 504,
      "id" : 0,
      "format" : "CEA_708_SMPTE_2038",
      "streamDescription" : "Captions Stream (pid: 504, CEA-708 in SMPTE-2038)"
    },
    "messages" : [ ]
  }, {
    "id" : 0,
    "pid" : 505,
    "type" : "METADATA",
    "state" : "NEW",
    "format" : "UNKNOWN",
    "info" : "INFO1",
    "elementaryStream" : {
      "messages" : [ ],
      "state" : "NEW",
      "pid" : 505,
      "id" : 0,
      "format" : "UNKNOWN",
      "info" : "INFO1",
      "streamDescription" : "Metadata Stream (pid: 505, Unknown, info: INFO1)"
    },
    "messages" : [ ]
  }, {
    "id" : 1,
    "pid" : 506,
    "type" : "METADATA",
    "state" : "NEW",
    "format" : "SMPTE_2038",
    "info" : "BLABLA",
    "elementaryStream" : {
      "messages" : [ ],
      "state" : "NEW",
```

```
    "pid" : 506,
    "id" : 1,
    "format" : "SMPTE_2038",
    "info" : "BLABLA",
    "streamDescription" : "Metadata Stream (pid: 506, SMPTE-2038, info: BLABLA)"
  },
  "messages" : [ ]
} ]
```

Output Stream State

JSON object containing output stream state.

Path	Type	Description
`id`	`Number`	Unique identifier of this output stream.
`uniqueId`	`String`	Unique pipeline output UUID. Randomly generated when a user creates the output, removed on the output removal. Can't be downloaded with pipeline preset. Survives stopping of pipeline and restart of service/machine.
`status`	`String`	Allowed values are `"ERROR"`, `"WARNING"`, `"OK"` or `"UNKNOWN"`.
`name`	`String`	Name of this output stream.
`sink`	`String`	URL string value. It should be in format `"rtp://<ip-address>:<port>:<interface-id>"` for RTP/UDP stream, `"udp://<ip-address>:<port>:<interface-id>"` for UDP stream, `"srt://<ip-address-or-web-url>:<port>:<interface-id>"` for SRT stream, `"sdi://<sdi-device-id>:<port-index>"` for SDI stream , `\"ndi://<ndi-name>\"` for NDI stream or `"file://<file>"` for transcoder system local file. The `<interface-id>` should be the identifier of one of the available network interfaces. The `<sdi-device-id>` should be the identifier of one of the available SDI devices. Examples: `"rtp://127.0.0.1:5000:lo"`, `"udp://127.0.0.1:5001:lo"`, `"sdi://aja-sdi-0:0"` or `"file:///tmp/input.ts"`.
`description`	`String`	Output stream description.
`encodingLatency`	`Number`	Encoding latency of progressive frame/interlaced field in milliseconds for this output stream (average over the last few seconds).
`transcodedFps`	`Number`	Number of transcoded progressive frames/interlaced fields per second for this output stream (average over the last few seconds).
`transcodingLatency`	`Number`	Transcoding latency of progressive frame/interlaced field in milliseconds for this output stream (average over the last few seconds).
`latencyAddition`	`Number`	Latency as difference between PTS of corresponding input and output frame in msec. Negative value means undefined.

Path	Type	Description
`droppedFps`	`Number`	Number of dropped progressive frames/interlaced fields per second for this output stream (average over the last few seconds).
`messages`	`Array`	Messages for this output stream.

Output Stream State example:


```
{
  "id" : 0,
  "uniqueId" : "ddc2b880-d32c-4bae-afd1-b4c5b2b75d19",
  "status" : "ERROR",
  "name" : "Output Stream 1",
  "sink" : "rtp://127.0.0.1:50001",
  "description" : "H.264 1920x1080 4:2:0 8-bit",
  "encodingLatency" : 0.0,
  "transcodedFps" : 0.0,
  "transcodingLatency" : 0.0,
  "droppedFps" : 0.0,
  "messages" : [ {
    "severity" : "ERROR",
    "message" : "Output Stream 1: Not producing any output."
  } ],
  "latencyAddition" : 0
}
```

Configuration of All Pipelines

Allows you to list and replace configuration of all pipelines at once.

List All Existing Pipelines

You may list all existing pipelines by using this action. It returns JSON array containing all existing pipelines.

HTTP request

HTTP

```
GET /api/pipelines HTTP/1.1
Authorization: Basic YWRtaW46cGFzc3dvcmQ=
Accept: application/json
Host: 127.0.0.1
```

Curl request

BASH

```
$ curl 'https://127.0.0.1/api/pipelines' -i -u 'admin:password' -X GET \
    -H 'Accept: application/json'
```

Response data

Path	Type	Description
`[]`	`Array of Pipelines`	An array of all existing pipelines. See Pipeline.

HTTP response

HTTP

```
HTTP/1.1 200 OK
Vary: Origin
Vary: Access-Control-Request-Method
Vary: Access-Control-Request-Headers
Content-Type: application/json
Content-Length: 4775

[ {
  "version" : "0.16",
  "id" : 0,
  "name" : "Pipeline 1",
  "runningState" : "INACTIVE",
  "inputs" : [ {
    "id" : 0,
    "source" : {
      "streamType" : "MPEG_TS",
      "address" : {
        "url" : "srt://127.0.0.1:5001:lo",
        "srt" : {
          "mode" : "CALLER",
          "latency" : 50,
          "streamID" : "#!::u=admin,t=file,m=publish,r=results.csv",
          "passphrase" : "",
          "redundancy" : [ "srt://127.0.0.1:5002:lo" ]
        }
      },
      "complianceWith" : "AUTO",
      "addressPCR" : {
        "url" : "srt://0.0.0.0:5000:lo",
        "srt" : {
          "mode" : "LISTENER",
          "latency" : 50,
          "passphrase" : ""
        }
      }
    },
    "video" : {
      "format" : "J2K",
      "width" : 1920,
      "height" : 1080,
      "sampling" : "4:2:2",
      "depth" : 10,
      "scanRate" : "50",
      "scan" : "INTERLACED",
      "fieldOrder" : "TOP_FIELD_FIRST",
      "filters" : {
        "yOffset" : 16,
        "yGain" : 0.2,
        "rOffset" : 1,
        "rGain" : 0.01,
        "gOffset" : 2,
        "gGain" : 0.02,
        "bOffset" : 3,
        "bGain" : 0.03,
        "rgbClip" : true,
        "hue" : 4.5,
        "saturation" : 0.5
      }
    },
    "audio" : [ {
      "id" : 0,
      "pid" : 661,
      "format" : "AES3",
      "rate" : 48000,
      "channels" : 2,
      "filters" : {
        "volumeGain" : 2.5
      },
      "passthrough" : false
```

```
    } ],
    "captions" : [ {
      "id" : 0,
      "pid" : 662,
      "format" : "CEA_708_SMPTE_2038"
    } ],
    "metadata" : [ {
      "id" : 0,
      "pid" : 663,
      "format" : "UNKNOWN"
    } ]
  } ],
  "outputs" : [ {
    "streamType" : "MPEG_TS",
    "id" : 0,
    "name" : "Output Stream 1",
    "sink" : {
      "address" : {
        "url" : "srt://0.0.0.0:6001:lo",
        "srt" : {
          "mode" : "LISTENER",
          "overhead" : 20,
          "encryption" : "AES_256",
          "passphrase" : "0123456789",
          "redundancy" : [ "srt://127.0.0.1:5002:lo" ]
        }
      }
    },
    "ts" : {
      "service" : {
        "name" : "SERVICE",
        "providerName" : "PROVIDER"
      },
      "pcrStreamLocation" : "SEPARATE",
      "ebpLength" : 5,
      "ebpLengthUnit" : "SECONDS",
      "bitrate" : 11000000,
      "outputComplianceWith" : "STANDARD"
    },
    "video" : {
      "pid" : 670,
      "format" : "H264",
      "width" : 720,
      "height" : 576,
      "sampling" : "4:2:0",
      "depth" : 8,
      "scanRateType" : "SAME_AS_INPUT",
      "scan" : "INTERLACED",
      "fieldOrder" : "TOP_FIELD_FIRST",
      "h264" : {
        "coding" : "CABAC",
        "profile" : "HIGH",
        "gopSize" : 120,
        "bCount" : 0
      },
      "bitrate" : 8000000,
      "captions" : {
        "reference" : {
          "inputId" : 0,
          "id" : 0
        }
      },
      "crop" : {
        "fit" : "PAD",
        "from" : [ 0, 10 ],
        "to" : [ -1, 1070 ]
      },
      "logo" : {
        "id" : 0,
        "horizontalPosition" : 5,
```

```
          "verticalPosition" : 10,
          "horizontalScaling" : 150.0,
          "verticalScaling" : 200.0
        }
      },
      "audio" : [ {
        "id" : 0,
        "pid" : 671,
        "format" : "AAC_ADTS",
        "rate" : 48000,
        "name" : "Output audio 1",
        "bitrate" : 1000000,
        "channels" : 2,
        "references" : [ {
          "inputId" : 0,
          "id" : 0,
          "channel" : 0
        } ]
      } ],
      "captions" : [ {
        "id" : 0,
        "pid" : 672,
        "format" : "CEA_708_SMPTE_2038",
        "reference" : {
          "inputId" : 0,
          "id" : 0
        }
      } ],
      "metadata" : [ {
        "id" : 0,
        "pid" : 673,
        "format" : "UNKNOWN",
        "reference" : {
          "inputId" : 0,
          "id" : 0
        }
      } ]
    } ],
    "uploadedLogos" : [ {
      "id" : 0,
      "name" : "logo.png",
      "codestream" : "iVBORw0KGgoAAAANSUhEUgAAAAEAAAABCAIAAACQd1PeAAAAD0lEQVQIHQEEAPv/AP///wX+Av4DfRnGAAAAAElFTkSuQmCC"
    } ],
    "links" : [ {
      "rel" : "self",
      "href" : "https://127.0.0.1/api/pipeline/0"
    }, {
      "rel" : "state",
      "href" : "https://127.0.0.1/api/pipeline/0/state"
    } ]
  }, {
    "version" : "0.16",
    "id" : 1,
    "name" : "Pipeline 2",
    "inputs" : [ {
      "id" : 0,
      "source" : {
        "streamType" : "MPEG_TS",
        "address" : {
          "url" : "rtp://0.0.0.0:5002:lo"
        },
        "complianceWith" : "AUTO"
      },
      "video" : {
        "format" : "J2K",
        "width" : 1920,
        "height" : 1080,
        "sampling" : "4:2:2",
        "depth" : 10,
        "scanRate" : "60000/1001",
```

```
      "scan" : "PROGRESSIVE"
    }
  } ],
  "links" : [ {
    "rel" : "self",
    "href" : "https://127.0.0.1/api/pipeline/1"
  }, {
    "rel" : "state",
    "href" : "https://127.0.0.1/api/pipeline/1/state"
  } ]
} ]
```

Replace All Pipelines

You may replace all existing pipelines by using this action. It removes all existing pipelines and creates all pipelines contained in given JSON array.

You can use the response JSON array from listing all existing pipelines as input for this action.

Request data

Path	Type	Description
`[]`	`Array`	An array of pipelines to be replaced. See Pipeline.

HTTP request

HTTP

```
POST /api/pipelines HTTP/1.1
Content-Type: application/json
Authorization: Basic YWRtaW46cGFzc3dvcmQ=
Content-Length: 2649
Host: 127.0.0.1

[{"id":0,"name":"Pipeline 1","runningState":"INACTIVE","inputs":[{"id":0, "source":{"address":{"url":"srt://127.0.0.1
, {"id":1,"name":"Pipeline 2","inputs":[{"id":0, "source":{"address":{"url":"rtp://0.0.0.0:5002:lo"},"streamType":"MP
```

Curl request

BASH

```
$ curl 'https://127.0.0.1/api/pipelines' -i -u 'admin:password' -X POST \
    -H 'Content-Type: application/json' \
    -d '[{"id":0,"name":"Pipeline 1","runningState":"INACTIVE","inputs":[{"id":0, "source":{"address":
{"url":"srt://127.0.0.1:5001:lo","srt":
{"mode":"CALLER","latency":50,"streamID":"#!::u=admin,t=file,m=publish,r=results.csv","passphrase":"",
"redundancy":["srt://127.0.0.1:5002:lo"]}},"streamType":"MPEG_TS","complianceWith":"AUTO","addressPCR":
{"url":"srt://0.0.0.0:5000:lo","srt":{"mode":"LISTENER","latency":50,"passphrase":""}}},"video":
{"format":"J2K","width":1920,"height":1080,"sampling":"4:2:2","depth":10,"scanRate":"50","scan":"INTERLACED","field
Order":"TOP_FIELD_FIRST","filters":
{"yOffset":16,"yGain":0.2,"rOffset":1,"rGain":0.01,"gOffset":2,"gGain":0.02,"bOffset":3,"bGain":0.03,"rgbClip":true
,"hue":4.5,"saturation":0.5}},"audio":[{"id":0,"pid":661,"format":"AES3","rate":48000,"channels":2,"filters":
{"volumeGain":2.5},"passthrough":false}],"captions":[{"id":0,"pid":662,"format":"CEA_708_SMPTE_2038"}],"metadata":
[{"id":0,"pid":663,"format":"UNKNOWN"}]}],"outputs":[{"id":0,"name":"Output Stream
1","streamType":"MPEG_TS","sink":{"address":{"url":"srt://0.0.0.0:6001:lo","srt":
{"mode":"LISTENER","overhead":20,"encryption":"AES_256","passphrase":"0123456789", "redundancy":
["srt://127.0.0.1:5002:lo"]}}},"ts":{"service":
{"name":"SERVICE","providerName":"PROVIDER"},"pcrStreamLocation":"SEPARATE","ebpLength":5,"ebpLengthUnit":"SECONDS"
,"bitrate":11000000,"outputComplianceWith":"STANDARD"},"video":
{"pid":670,"format":"H264","width":720,"height":576,"crop":{"fit":"PAD","from":[0,10],"to":[-1,1070]},"logo":
{"id":0,"horizontalPosition":5,"verticalPosition":10,"horizontalScaling":150.0,"verticalScaling":200.0},"sampling":
"4:2:0","depth":8,"scanRateType":"SAME_AS_INPUT","scan":"INTERLACED","fieldOrder":"TOP_FIELD_FIRST","captions":
{"reference":{"inputId":0,"id":0}},"bitrate":8000000,"h264":
{"coding":"CABAC","profile":"HIGH","gopSize":120,"bCount":0}},"audio":
[{"id":0,"pid":671,"format":"AAC_ADTS","rate":48000,"name":"Output audio 1","bitrate":1000000,"references":
[{"inputId":0,"id":0,"channel":0}],"channels":2}],"captions":
[{"id":0,"pid":672,"format":"CEA_708_SMPTE_2038","reference":{"inputId":0,"id":0}}],"metadata":
[{"id":0,"pid":673,"format":"UNKNOWN","reference":{"inputId":0,"id":0}}]}],"uploadedLogos":
[{"id":0,"name":"logo.png","codestream":"iVBORw0KGgoAAAANSUhEUgAAAAEAAAABCAIAAACQd1PeAAAAD0lEQVQIHQEEAPv/AP///wX+Av
4DfRnGAAAAAElFTkSuQmCC"}]}
, {"id":1,"name":"Pipeline 2","inputs":[{"id":0, "source":{"address":
{"url":"rtp://0.0.0.0:5002:lo"},"streamType":"MPEG_TS","complianceWith":"AUTO"},"video":
{"format":"J2K","width":1920,"height":1080,"sampling":"4:2:2","depth":10,"scanRate":"60000/1001","scan":"PROGRESSIV
E"}}]}]
'
```

HTTP response

HTTP

```
HTTP/1.1 200 OK
Vary: Origin
Vary: Access-Control-Request-Method
Vary: Access-Control-Request-Headers
```

Configuration of a Single Pipeline

Allows you to create a new pipeline or get, modify or delete some existing pipeline.

Create a New Pipeline

You may create a new pipeline by using this action. It takes a JSON object containing the new pipeline configuration.

Request data

Path	Type	Description
{}	Pipeline	Configuration of new pipeline. See Pipeline.

HTTP request

HTTP

```
POST /api/pipeline HTTP/1.1
Content-Type: application/json
Authorization: Basic YWRtaW46cGFzc3dvcmQ=
Content-Length: 261
Host: 127.0.0.1

{"name":"Pipeline 3","inputs":[{"id":0, "source":{"address":{"url":"rtp://0.0.0.0:5003:lo"},"streamType":"MPEG_TS","c
```

Curl request

BASH

```
$ curl 'https://127.0.0.1/api/pipeline' -i -u 'admin:password' -X POST \
    -H 'Content-Type: application/json' \
    -d '{"name":"Pipeline 3","inputs":[{"id":0, "source":{"address":
{"url":"rtp://0.0.0.0:5003:lo"},"streamType":"MPEG_TS","complianceWith":"AUTO"},"video":
{"format":"J2K","width":1920,"height":1080,"sampling":"4:2:2","depth":10,"scanRate":"24","scan":"PROGRESSIVE"}}]}'
```

HTTP response

HTTP

```
HTTP/1.1 201 Created
Vary: Origin
Vary: Access-Control-Request-Method
Vary: Access-Control-Request-Headers
Location: https://127.0.0.1/api/pipeline/2
```

Get an Existing Pipeline

You may get an existing pipeline by using this action. It takes pipeline identifier as a parameter and it returns JSON object with pipeline configuration.

HTTP request

HTTP

```
GET /api/pipeline/0 HTTP/1.1
Authorization: Basic YWRtaW46cGFzc3dvcmQ=
Accept: application/json
Host: 127.0.0.1
```

Curl request

BASH

```
$ curl 'https://127.0.0.1/api/pipeline/0' -i -u 'admin:password' -X GET \
    -H 'Accept: application/json'
```

Response data

Path	Type	Description
`{}`	`Pipeline`	Configuration of existing pipeline. See Pipeline.

HTTP response

HTTP

```
HTTP/1.1 200 OK
Vary: Origin
Vary: Access-Control-Request-Method
Vary: Access-Control-Request-Headers
Content-Type: application/json
Content-Length: 4136

{
  "version" : "0.16",
  "id" : 0,
  "name" : "Pipeline 1",
  "runningState" : "INACTIVE",
  "inputs" : [ {
    "id" : 0,
    "source" : {
      "streamType" : "MPEG_TS",
      "address" : {
        "url" : "srt://127.0.0.1:5001:lo",
        "srt" : {
          "mode" : "CALLER",
          "latency" : 50,
          "streamID" : "#!::u=admin,t=file,m=publish,r=results.csv",
          "passphrase" : "",
          "redundancy" : [ "srt://127.0.0.1:5002:lo" ]
        }
      },
      "complianceWith" : "AUTO",
      "addressPCR" : {
        "url" : "srt://0.0.0.0:5000:lo",
        "srt" : {
          "mode" : "LISTENER",
          "latency" : 50,
          "passphrase" : ""
        }
      }
    },
    "video" : {
      "format" : "J2K",
      "width" : 1920,
      "height" : 1080,
      "sampling" : "4:2:2",
      "depth" : 10,
      "scanRate" : "50",
      "scan" : "INTERLACED",
      "fieldOrder" : "TOP_FIELD_FIRST",
      "filters" : {
        "yOffset" : 16,
        "yGain" : 0.2,
        "rOffset" : 1,
        "rGain" : 0.01,
        "gOffset" : 2,
        "gGain" : 0.02,
        "bOffset" : 3,
        "bGain" : 0.03,
        "rgbClip" : true,
        "hue" : 4.5,
        "saturation" : 0.5
      }
    },
    "audio" : [ {
      "id" : 0,
      "pid" : 661,
      "format" : "AES3",
      "rate" : 48000,
      "channels" : 2,
      "filters" : {
        "volumeGain" : 2.5
      },
      "passthrough" : false
```

```
    } ],
    "captions" : [ {
      "id" : 0,
      "pid" : 662,
      "format" : "CEA_708_SMPTE_2038"
    } ],
    "metadata" : [ {
      "id" : 0,
      "pid" : 663,
      "format" : "UNKNOWN"
    } ]
  } ],
  "outputs" : [ {
    "streamType" : "MPEG_TS",
    "id" : 0,
    "name" : "Output Stream 1",
    "sink" : {
      "address" : {
        "url" : "srt://0.0.0.0:6001:lo",
        "srt" : {
          "mode" : "LISTENER",
          "overhead" : 20,
          "encryption" : "AES_256",
          "passphrase" : "0123456789",
          "redundancy" : [ "srt://127.0.0.1:5002:lo" ]
        }
      }
    },
    "ts" : {
      "service" : {
        "name" : "SERVICE",
        "providerName" : "PROVIDER"
      },
      "pcrStreamLocation" : "SEPARATE",
      "ebpLength" : 5,
      "ebpLengthUnit" : "SECONDS",
      "bitrate" : 11000000,
      "outputComplianceWith" : "STANDARD"
    },
    "video" : {
      "pid" : 670,
      "format" : "H264",
      "width" : 720,
      "height" : 576,
      "sampling" : "4:2:0",
      "depth" : 8,
      "scanRateType" : "SAME_AS_INPUT",
      "scan" : "INTERLACED",
      "fieldOrder" : "TOP_FIELD_FIRST",
      "h264" : {
        "coding" : "CABAC",
        "profile" : "HIGH",
        "gopSize" : 120,
        "bCount" : 0
      },
      "bitrate" : 8000000,
      "captions" : {
        "reference" : {
          "inputId" : 0,
          "id" : 0
        }
      },
      "crop" : {
        "fit" : "PAD",
        "from" : [ 0, 10 ],
        "to" : [ -1, 1070 ]
      },
      "logo" : {
        "id" : 0,
        "horizontalPosition" : 5,
```

```
        "verticalPosition" : 10,
        "horizontalScaling" : 150.0,
        "verticalScaling" : 200.0
      }
    },
    "audio" : [ {
      "id" : 0,
      "pid" : 671,
      "format" : "AAC_ADTS",
      "rate" : 48000,
      "name" : "Output audio 1",
      "bitrate" : 1000000,
      "channels" : 2,
      "references" : [ {
        "inputId" : 0,
        "id" : 0,
        "channel" : 0
      } ]
    } ],
    "captions" : [ {
      "id" : 0,
      "pid" : 672,
      "format" : "CEA_708_SMPTE_2038",
      "reference" : {
        "inputId" : 0,
        "id" : 0
      }
    } ],
    "metadata" : [ {
      "id" : 0,
      "pid" : 673,
      "format" : "UNKNOWN",
      "reference" : {
        "inputId" : 0,
        "id" : 0
      }
    } ]
  } ],
  "uploadedLogos" : [ {
    "id" : 0,
    "name" : "logo.png",
    "codestream" : "iVBORw0KGgoAAAANSUhEUgAAAAEAAAABCAIAAACQd1PeAAAAD0lEQVQIHQEEAPv/AP///wX+Av4DfRnGAAAAAElFTkSuQmCC"
  } ],
  "_links" : {
    "self" : {
      "href" : "https://127.0.0.1/api/pipeline/0"
    },
    "state" : {
      "href" : "https://127.0.0.1/api/pipeline/0/state"
    }
  }
}
```


Modify an Existing Pipeline

You may modify an existing pipeline configuration by using this action. It takes pipeline identifier and JSON object with new pipeline configuration as parameters.

Request data

Path	Type	Description
`{}`	`Pipeline`	Configuration of new pipeline. See Pipeline.

HTTP request

```HTTP
PUT /api/pipeline/1 HTTP/1.1
Content-Type: application/json
Authorization: Basic YWRtaW46cGFzc3dvcmQ=
Content-Length: 269
Host: 127.0.0.1

{"name":"Pipeline 2 Changed","inputs":[{"id":0, "source":{"address":{"url":"rtp://0.0.0.0:5002:lo"},"streamType":"MPE
```

Curl request

```BASH
$ curl 'https://127.0.0.1/api/pipeline/1' -i -u 'admin:password' -X PUT \
    -H 'Content-Type: application/json' \
    -d '{"name":"Pipeline 2 Changed","inputs":[{"id":0, "source":{"address":
{"url":"rtp://0.0.0.0:5002:lo"},"streamType":"MPEG_TS","complianceWith":"AUTO"},"video":
{"format":"J2K","width":1920,"height":1080,"sampling":"4:2:2","depth":10,"scanRate":"24","scan":"PROGRESSIVE"}}]}'
```

HTTP response

```HTTP
HTTP/1.1 200 OK
Vary: Origin
Vary: Access-Control-Request-Method
Vary: Access-Control-Request-Headers
Location: https://127.0.0.1/api/pipeline/1
```

Start an Existing Pipeline

You may start an existing pipeline by using this action. It takes pipeline identifier as a parameter. Starting an already started pipeline or a pipeline that's in the process of starting/stopping returns `503`.

HTTP request

```HTTP
PUT /api/pipeline/1/start HTTP/1.1
Content-Type: application/json
Authorization: Basic YWRtaW46cGFzc3dvcmQ=
Host: 127.0.0.1
```

Curl request

```BASH
$ curl 'https://127.0.0.1/api/pipeline/1/start' -i -u 'admin:password' -X PUT \
    -H 'Content-Type: application/json'
```

HTTP response

```HTTP
HTTP/1.1 503 Service Unavailable
Vary: Origin
Vary: Access-Control-Request-Method
Vary: Access-Control-Request-Headers
Content-Type: application/json
Content-Length: 46

Failed to start the pipeline, try again later.
```

Stop an Existing Pipeline

You may stop an existing pipeline by using this action. It takes pipeline identifier as a parameter. Stopping an already stopped pipeline or a pipeline that's in the process of starting/stopping returns `503`.

HTTP request

```HTTP
PUT /api/pipeline/1/stop HTTP/1.1
Content-Type: application/json
Authorization: Basic YWRtaW46cGFzc3dvcmQ=
Host: 127.0.0.1
```

Curl request

```BASH
$ curl 'https://127.0.0.1/api/pipeline/1/stop' -i -u 'admin:password' -X PUT \
    -H 'Content-Type: application/json'
```

HTTP response

```HTTP
HTTP/1.1 503 Service Unavailable
Vary: Origin
Vary: Access-Control-Request-Method
Vary: Access-Control-Request-Headers
Content-Type: application/json
Content-Length: 45

Failed to stop the pipeline, try again later.
```

Remove an Existing Pipeline

You may delete an existing pipeline by using this action. It takes pipeline identifier as a parameter.

HTTP request

```HTTP
DELETE /api/pipeline/1 HTTP/1.1
Authorization: Basic YWRtaW46cGFzc3dvcmQ=
Host: 127.0.0.1
```

Curl request

```BASH
$ curl 'https://127.0.0.1/api/pipeline/1' -i -u 'admin:password' -X DELETE
```

HTTP response

```HTTP
HTTP/1.1 200 OK
Vary: Origin
Vary: Access-Control-Request-Method
Vary: Access-Control-Request-Headers
```

Retrieving Pipeline State

Allows you to retrieve current state of all existing pipelines.

Get Pipeline Runtime State

You may retrieve the runtime state of a running pipeline by using this action. It takes pipeline identifier as a parameter and it returns JSON object containing the pipeline runtime state.

HTTP request

HTTP

```
GET /api/pipeline/0/state HTTP/1.1
Authorization: Basic YWRtaW46cGFzc3dvcmQ=
Accept: application/json
Host: 127.0.0.1
```

Curl request

BASH

```
$ curl 'https://127.0.0.1/api/pipeline/0/state' -i -u 'admin:password' -X GET \
    -H 'Accept: application/json'
```

Response data

Path	Type	Description
`{}`	`Pipeline State`	Current state of existing pipeline. See Pipeline State.

HTTP response

HTTP

```
HTTP/1.1 200 OK
Vary: Origin
Vary: Access-Control-Request-Method
Vary: Access-Control-Request-Headers
Content-Type: application/json
Content-Length: 5320

{
  "id" : 0,
  "status" : "ERROR",
  "operationState" : "STARTED",
  "licenseStatus" : "GRANTED",
  "licenseExpirationDate" : "1970-01-01T00:00:00Z",
  "licenseRemainingTime" : "00:00:00",
  "name" : "Pipeline 1",
  "uniqueId" : "577b54da-5dfe-4bea-add6-b689e8cd7eea",
  "inputs" : [ {
    "id" : 0,
    "status" : "ERROR",
    "name" : "rtp://0.0.0.0:5000:lo",
    "description" : "JPEG2000 1920x1080 4:4:4 10-bit",
    "elementaryStreams" : [ {
      "id" : 0,
      "pid" : 501,
      "type" : "UNKNOWN",
      "state" : "NEW",
      "elementaryStream" : {
        "messages" : [ ],
        "state" : "NEW",
        "pid" : 501,
        "id" : 0,
        "streamDescription" : "Unknown Stream (pid: 501)"
      }
    }, {
      "id" : 0,
      "pid" : 502,
      "type" : "VIDEO",
      "state" : "NEW",
      "format" : "J2K",
      "width" : 1920,
      "height" : 1080,
      "depth" : 10,
      "sampling" : "4:2:2",
      "scanRate" : "60000/1001",
      "scan" : "INTERLACED",
      "fieldOrder" : "BOTTOM_FIELD_FIRST",
      "elementaryStream" : {
        "messages" : [ ],
        "state" : "NEW",
        "pid" : 502,
        "id" : 0,
        "format" : "J2K",
        "width" : 1920,
        "height" : 1080,
        "depth" : 10,
        "sampling" : "S422",
        "scanRate" : "60000/1001",
        "scan" : "INTERLACED",
        "fieldOrder" : "BOTTOM_FIELD_FIRST",
        "streamDescription" : "Video Stream (pid: 502, JPEG2000 1920x1080i (bottom-first) @59.94Hz 4:2:2 10-bit)"
      },
      "messages" : [ ]
    }, {
      "id" : 0,
      "pid" : 503,
      "type" : "AUDIO",
      "state" : "NEW",
      "format" : "AES3",
      "channelCount" : 2,
```

```
      "rate" : 48000,
      "bitrate" : 235000,
      "elementaryStream" : {
        "messages" : [ {
          "severity" : "ERROR",
          "message" : "${stream} wasn't detected in source stream."
        } ],
        "state" : "NEW",
        "pid" : 503,
        "id" : 0,
        "format" : "AES3",
        "channelCount" : 2,
        "rate" : 48000,
        "bitrate" : 235000,
        "bitsPerSample" : 0,
        "streamDescription" : "Audio Stream (pid: 503, AES3 2-channels 48000 Hz 235 kb/s)"
      },
      "messages" : [ {
        "severity" : "ERROR",
        "message" : "This audio stream wasn't detected in source stream."
      } ]
    }, {
      "id" : 1,
      "pid" : 499,
      "type" : "AUDIO",
      "state" : "NEW",
      "format" : "AES3_DOLBY_E",
      "channelCount" : 2,
      "rate" : 48000,
      "bitrate" : 128000,
      "elementaryStream" : {
        "messages" : [ {
          "severity" : "ERROR",
          "message" : "${stream} wasn't detected in source stream."
        } ],
        "state" : "NEW",
        "pid" : 499,
        "id" : 1,
        "format" : "AES3_DOLBY_E",
        "channelCount" : 2,
        "rate" : 48000,
        "bitrate" : 128000,
        "bitsPerSample" : 0,
        "streamDescription" : "Audio Stream (pid: 499, AES3_DOLBY_E 2-channels 48000 Hz 128 kb/s)"
      },
      "messages" : [ {
        "severity" : "ERROR",
        "message" : "This audio stream wasn't detected in source stream."
      } ]
    }, {
      "id" : 0,
      "pid" : 504,
      "type" : "CAPTIONS",
      "state" : "NEW",
      "format" : "CEA_708_SMPTE_2038",
      "elementaryStream" : {
        "messages" : [ ],
        "state" : "NEW",
        "pid" : 504,
        "id" : 0,
        "format" : "CEA_708_SMPTE_2038",
        "streamDescription" : "Captions Stream (pid: 504, CEA-708 in SMPTE-2038)"
      },
      "messages" : [ ]
    }, {
      "id" : 0,
      "pid" : 505,
      "type" : "METADATA",
      "state" : "NEW",
      "format" : "UNKNOWN",
```

```
        "info" : "INFO1",
        "elementaryStream" : {
          "messages" : [ ],
          "state" : "NEW",
          "pid" : 505,
          "id" : 0,
          "format" : "UNKNOWN",
          "info" : "INFO1",
          "streamDescription" : "Metadata Stream (pid: 505, Unknown, info: INFO1)"
        },
        "messages" : [ ]
      }, {
        "id" : 1,
        "pid" : 506,
        "type" : "METADATA",
        "state" : "NEW",
        "format" : "SMPTE_2038",
        "info" : "BLABLA",
        "elementaryStream" : {
          "messages" : [ ],
          "state" : "NEW",
          "pid" : 506,
          "id" : 1,
          "format" : "SMPTE_2038",
          "info" : "BLABLA",
          "streamDescription" : "Metadata Stream (pid: 506, SMPTE-2038, info: BLABLA)"
        },
        "messages" : [ ]
      } ],
      "decodedFps" : 0.0,
      "decodingLatency" : 0.0,
      "invalidFps" : 0.0,
      "messages" : [ {
        "severity" : "ERROR",
        "message" : "rtp://0.0.0.0:5000:lo: input stream #0 error."
      } ]
    } ],
    "outputs" : [ {
      "id" : 0,
      "uniqueId" : "ddc2b880-d32c-4bae-afd1-b4c5b2b75d19",
      "status" : "ERROR",
      "name" : "Output Stream 1",
      "sink" : "rtp://127.0.0.1:50001",
      "description" : "H.264 1920x1080 4:2:0 8-bit",
      "encodingLatency" : 0.0,
      "transcodedFps" : 0.0,
      "transcodingLatency" : 0.0,
      "droppedFps" : 0.0,
      "messages" : [ {
        "severity" : "ERROR",
        "message" : "Output Stream 1: Not producing any output."
      } ],
      "latencyAddition" : 0
    } ],
    "_links" : {
      "self" : {
        "href" : "https://127.0.0.1/api/pipeline/0/state"
      },
      "pipeline" : {
        "href" : "https://127.0.0.1/api/pipeline/0"
      }
    }
  }
```

System

Version info

Allows you to retrieve information about current running version of transcoder.

HTTP request

```http
GET /api/system/version HTTP/1.1
Authorization: Basic YWRtaW46cGFzc3dvcmQ=
Accept: application/json
Host: 127.0.0.1
```

Curl request

```bash
$ curl 'https://127.0.0.1/api/system/version' -i -u 'admin:password' -X GET \
    -H 'Accept: application/json'
```

Response data

Path	Type	Description
`transcoder`	`Object`	Transcoder version info.
`transcoder.version`	`String`	Version in format `<major>.<minor>.<maintenance>`.
`transcoder.stage`	`String`	Version stage (e.g., `release` or `beta`).
`transcoder.date`	`String`	Release date in format `<year>-<month>-<day>`.
`transcoder.hash`	`String`	Version hash (unique version identifier).
`transcoder.deploymentMode`	`String`	Deployment mode (`host` or `docker`)
`transcoder.fullVersionName`	`String`	Full version string.
`j2kDec`	`Object`	JPEG2000 SDK decoder version info.
`j2kDec.version`	`String`	Version in format `<major>.<minor>.<maintenance>`.
`j2kDec.date`	`String`	Release date in format `<year>-<month>-<day>`.
`j2kDec.hash`	`String`	Version hash (unique version identifier).
`j2kDec.stage`	`String`	Version stage (e.g., `release`).
`j2kDec.type`	`String`	Version type (e.g., `full` or `evaluation`).
`j2kDec.technology`	`Array`	Array of supported technologies (`cpu`, `cuda` or `opencl`).
`j2kEnc`	`Object`	JPEG2000 SDK encoder version info.
`j2kEnc.version`	`String`	Version in format `<major>.<minor>.<maintenance>`.
`j2kEnc.date`	`String`	Release date in format `<year>-<month>-<day>`.
`j2kEnc.hash`	`String`	Version hash (unique version identifier).

Path	Type	Description
`j2kEnc.stage`	`String`	Version stage (e.g., **`release`**).
`j2kEnc.type`	`String`	Version type (e.g., **`full`** or **`evaluation`**).
`j2kEnc.technology`	`Array`	Array of supported technologies (**`cpu`**, **`cuda`** or **`opencl`**).
`j2kTrc`	`Object`	JPEG2000 SDK transcoder version info.
`j2kTrc.version`	`String`	Version in format **`<major>.<minor>.<maintenance>`**.
`j2kTrc.date`	`String`	Release date in format **`<year>-<month>-<day>`**.
`j2kTrc.hash`	`String`	Version hash (unique version identifier).
`j2kTrc.stage`	`String`	Version stage (e.g., **`release`**).
`j2kTrc.type`	`String`	Version type (e.g., **`full`** or **`evaluation`**).
`j2kTrc.technology`	`Array`	Array of supported technologies (**`cpu`**, **`cuda`** or **`opencl`**).
`nvidiaDriver`	`String`	NVIDIA driver version in format **`<major>.<minor>`** or **`<major>.<minor>.<maintenance>`**.
`cudaDriver`	`String`	CUDA driver version in format **`<major>.<minor>`**.
`cudaRuntime`	`String`	CUDA runtime version in format **`<major>.<minor>`**.

HTTP response

HTTP

```
HTTP/1.1 200 OK
Vary: Origin
Vary: Access-Control-Request-Method
Vary: Access-Control-Request-Headers
Content-Type: application/json
Content-Length: 1016

{
  "transcoder" : {
    "version" : "1.0.0",
    "stage" : "pipelines/38649",
    "date" : "2024-09-04",
    "hash" : "7814d7e05c38d0e5a50ba4f31850ece62938bba1",
    "fullVersionName" : "1.0.0-pipelines/38649 2024-09-04 7814d7e05c38d0e5a50ba4f31850ece62938bba1",
    "deploymentMode" : "host"
  },
  "j2kDec" : {
    "version" : "2.5.10",
    "stage" : "release",
    "date" : "2018-03-27",
    "hash" : "a22b87f11d83cd7a610a91e38b7ec6c8cef33664",
    "type" : "full",
    "technology" : [ "cuda", "cpu" ]
  },
  "j2kEnc" : {
    "version" : "2.5.10",
    "stage" : "release",
    "date" : "2018-03-27",
    "hash" : "a22b87f11d83cd7a610a91e38b7ec6c8cef33664",
    "type" : "full",
    "technology" : [ "cuda", "cpu" ]
  },
  "j2kTrc" : {
    "version" : "2.5.10",
    "stage" : "release",
    "date" : "2018-03-27",
    "hash" : "a22b87f11d83cd7a610a91e38b7ec6c8cef33664",
    "type" : "full",
    "technology" : [ "cuda", "cpu" ]
  },
  "nvidiaDriver" : "384.98",
  "cudaDriver" : "9.0",
  "cudaRuntime" : "9.0"
}
```

Get list of available network interfaces

Allows you to retrieve list of all available network interfaces whose identifiers can be used in pipeline source or output stream sink.

HTTP request

HTTP

```
GET /api/system/network-interfaces HTTP/1.1
Authorization: Basic YWRtaW46cGFzc3dvcmQ=
Accept: application/json
Host: 127.0.0.1
```

Curl request

BASH

```
$ curl 'https://127.0.0.1/api/system/network-interfaces' -i -u 'admin:password' -X GET \
    -H 'Accept: application/json'
```

Response data

Path	Type	Description
`[]`	`Array`	Array of available network interfaces.
`[].id`	`String`	Identifier of the network interface.
`[].name`	`String`	Display name of the network interface.
`[].connected`	`Boolean`	Indicates whether network interface is connected.

HTTP response

HTTP

```
HTTP/1.1 200 OK
Vary: Origin
Vary: Access-Control-Request-Method
Vary: Access-Control-Request-Headers
Content-Type: application/json
Content-Length: 507

[ {
  "id" : "docker0",
  "name" : "Bridge 1 (docker0)",
  "connected" : false
}, {
  "id" : "enp4s0f0",
  "name" : "Ethernet card 1 (enp4s0f0, 1 Gb/s link)",
  "connected" : true
}, {
  "id" : "enp4s0f1",
  "name" : "Ethernet card 2 (enp4s0f1)",
  "connected" : false
}, {
  "id" : "ens5",
  "name" : "Ethernet card 3 (ens5)",
  "connected" : false
}, {
  "id" : "ens5d1",
  "name" : "Ethernet card 4 (ens5d1)",
  "connected" : false
}, {
  "id" : "lo",
  "name" : "Loopback (lo)",
  "connected" : true
} ]
```

Get list of available SDI devices

Allows you to retrieve a list of all available SDI devices whose identifiers can be used in pipeline source or output stream sink.

HTTP request

HTTP

```
GET /api/system/sdi-devices HTTP/1.1
Authorization: Basic YWRtaW46cGFzc3dvcmQ=
Accept: application/json
Host: 127.0.0.1
```

Curl request

```bash
$ curl 'https://127.0.0.1/api/system/sdi-devices' -i -u 'admin:password' -X GET \
    -H 'Accept: application/json'
```

Response data

Path	Type	Description
`[]`	`Array`	Array of available SDI devices.
`[].id`	`String`	Identifier of the SDI device.
`[].name`	`String`	Name of the SDI device.
`[].ports`	`Array`	Available ports in SDI device.

HTTP response

```http
HTTP/1.1 200 OK
Vary: Origin
Vary: Access-Control-Request-Method
Vary: Access-Control-Request-Headers
Content-Type: application/json
Content-Length: 293

[ {
  "id" : "aja-sdi-0",
  "name" : "AJA SDI #0",
  "ports" : [ "1", "2", "3", "4" ]
}, {
  "id" : "aja-sdi-1",
  "name" : "AJA SDI #1",
  "ports" : [ "1", "2", "3", "4", "5", "6" ]
}, {
  "id" : "aja-sdi-2",
  "name" : "AJA SDI #2",
  "ports" : [ "1", "2", "3", "4", "5", "6", "7", "8" ]
} ]
```

Get list of available NDI source devices

Allows you to retrieve a list of all currently available NDI source devices whose identifiers can be used in pipeline source. Optionally, one or more NDI groups to search in may be specified using parameter `"ndiGroup="` with a comma-separated list of groups. The default group (if not specified) is `"public"`. For instance `"cameras,studio1,10am show"` would search for a source in the three groups named.

Please note that this API call works **asynchronously**, so it may return a growing list of sources if called repeatedly, as those are gradually being discovered on the network. Typically, the first call of this method with a given NDI group(s) initializes the search and returns none or a small subset of available sources, but after two or more seconds, the complete list is retrieved, so that all future calls would return the complete set of sources.

HTTP request

```http
GET /api/system/ndi-devices?ndiGroup=public HTTP/1.1
Authorization: Basic YWRtaW46cGFzc3dvcmQ=
Accept: application/json
Host: 127.0.0.1
```

Curl request

```bash
$ curl 'https://127.0.0.1/api/system/ndi-devices?ndiGroup=public' -i -u 'admin:password' -X GET \
    -H 'Accept: application/json'
```

Response data

Path	Type	Description
`[]`	`Array`	Array of available NDI devices in the group.
`[].name`	`String`	Name of the NDI device.

HTTP response

```http
HTTP/1.1 200 OK
Vary: Origin
Vary: Access-Control-Request-Method
Vary: Access-Control-Request-Headers
Content-Type: application/json
Content-Length: 170

[ {
  "name" : "TRANSCODER 0 (Studio 0)"
}, {
  "name" : "TRANSCODER 0 (Studio 1)"
}, {
  "name" : "TRANSCODER 1 (Studio 2)"
}, {
  "name" : "TRANSCODER 1 (Studio 3)"
} ]
```

Perform reinstallation of the current version of the device

Triggers reinstallation process which automatically reboots machine on success. The call is synchronous. Reinstallation makes clean installation of the currently booted software version of the device. No settings are preserved - only current licenses.

HTTP request

```http
GET /api/system/reinstall HTTP/1.1
Authorization: Basic YWRtaW46cGFzc3dvcmQ=
Accept: application/json
Host: 127.0.0.1
```

Curl request

```bash
$ curl 'https://127.0.0.1/api/system/reinstall' -i -u 'admin:password' -X GET \
    -H 'Accept: application/json'
```

Response data

Path	Type	Description
`state`	`String`	State of the reinstallation after the call is processed on the machine, i.e.: "SUCCESS" or "FAILED". It can return "IN-PROGRESS" if the call is already being processed.

Path	Type	Description
errorMessage	String	Error message in a case of failure.

HTTP response

```HTTP
HTTP/1.1 200 OK
Vary: Origin
Vary: Access-Control-Request-Method
Vary: Access-Control-Request-Headers
Content-Type: application/json
Content-Length: 48

{
  "state" : "SUCCESS",
  "errorMessage" : ""
}
```

Retrieve state of reinstallation process

It returns current state of a reinstallation process.

HTTP request

```HTTP
GET /api/system/reinstall-state HTTP/1.1
Authorization: Basic YWRtaW46cGFzc3dvcmQ=
Accept: application/json
Host: 127.0.0.1
```

Curl request

```BASH
$ curl 'https://127.0.0.1/api/system/reinstall-state' -i -u 'admin:password' -X GET \
    -H 'Accept: application/json'
```

Response data

Path	Type	Description
state	String	Current state of the reinstallation process. Possible values: "NOT-STARTED", "IN-PROGRESS", "FAILED", "SUCCESS".
errorMessage	String	Error message in a case of failed reinstallation process.

HTTP response

```HTTP
HTTP/1.1 200 OK
Vary: Origin
Vary: Access-Control-Request-Method
Vary: Access-Control-Request-Headers
Content-Type: application/json
Content-Length: 48

{
  "state" : "SUCCESS",
  "errorMessage" : ""
}
```

Perform factory reset of the device

Triggers factory reset process which automatically reboots machine on success. The call is synchronous.

HTTP request

```HTTP
GET /api/system/factory-reset HTTP/1.1
Authorization: Basic YWRtaW46cGFzc3dvcmQ=
Accept: application/json
Host: 127.0.0.1
```

Curl request

```BASH
$ curl 'https://127.0.0.1/api/system/factory-reset' -i -u 'admin:password' -X GET \
    -H 'Accept: application/json'
```

Response data

Path	Type	Description
`state`	`String`	State of the factory reset after the call is processed on the machine, i.e.: "SUCCESS" or "FAILED". It can return "IN-PROGRESS" if the call is already being processed.
`errorMessage`	`String`	Error message in a case of failure.

HTTP response

```HTTP
HTTP/1.1 200 OK
Vary: Origin
Vary: Access-Control-Request-Method
Vary: Access-Control-Request-Headers
Content-Type: application/json
Content-Length: 48

{
  "state" : "SUCCESS",
  "errorMessage" : ""
}
```

Retrieve state of factory reset process

It returns current state of a factory reset process.

HTTP request

```HTTP
GET /api/system/factory-reset-state HTTP/1.1
Authorization: Basic YWRtaW46cGFzc3dvcmQ=
Accept: application/json
Host: 127.0.0.1
```

Curl request

```BASH
$ curl 'https://127.0.0.1/api/system/factory-reset-state' -i -u 'admin:password' -X GET \
    -H 'Accept: application/json'
```

Response data

Path	Type	Description
`state`	`String`	Current state of the factory reset process. Possible values: "NOT-STARTED", "IN-PROGRESS", "FAILED", "SUCCESS".
`errorMessage`	`String`	Error message in a case of failed factory reset process.

HTTP response

```HTTP
HTTP/1.1 200 OK
Vary: Origin
Vary: Access-Control-Request-Method
Vary: Access-Control-Request-Headers
Content-Type: application/json
Content-Length: 48

{
  "state" : "SUCCESS",
  "errorMessage" : ""
}
```

Perform restart of main service

HTTP request

```HTTP
GET /api/system/restart-service HTTP/1.1
Authorization: Basic YWRtaW46cGFzc3dvcmQ=
Accept: application/json
Host: 127.0.0.1
```

Curl request

```BASH
$ curl 'https://127.0.0.1/api/system/restart-service' -i -u 'admin:password' -X GET \
    -H 'Accept: application/json'
```

Response data

Path	Type	Description
`message`	`String`	Info message what is going to happen.

HTTP response

```HTTP
HTTP/1.1 200 OK
Vary: Origin
Vary: Access-Control-Request-Method
Vary: Access-Control-Request-Headers
Content-Type: application/json
Content-Length: 70

{
  "message" : "Main program service will be restarted in a while."
}
```

Perform reboot of the machine

HTTP request

```http
GET /api/system/reboot HTTP/1.1
Authorization: Basic YWRtaW46cGFzc3dvcmQ=
Accept: application/json
Host: 127.0.0.1
```

Curl request

```bash
$ curl 'https://127.0.0.1/api/system/reboot' -i -u 'admin:password' -X GET \
    -H 'Accept: application/json'
```

Response data

Path	Type	Description
message	String	Info message what is going to happen.

HTTP response

```http
HTTP/1.1 200 OK
Vary: Origin
Vary: Access-Control-Request-Method
Vary: Access-Control-Request-Headers
Content-Type: application/json
Content-Length: 60

{
  "message" : "The machine will be rebooted in a while."
}
```

Perform shutdown of the machine

HTTP request

```http
GET /api/system/shutdown HTTP/1.1
Authorization: Basic YWRtaW46cGFzc3dvcmQ=
Accept: application/json
Host: 127.0.0.1
```

Curl request

```bash
$ curl 'https://127.0.0.1/api/system/shutdown' -i -u 'admin:password' -X GET \
    -H 'Accept: application/json'
```

Response data

Path	Type	Description
message	String	Info message what is going to happen.

HTTP response

HTTP

```
HTTP/1.1 200 OK
Vary: Origin
Vary: Access-Control-Request-Method
Vary: Access-Control-Request-Headers
Content-Type: application/json
Content-Length: 60

{
  "message" : "The machine will be shutdown in a while."
}
```

Errors

Error response

Path	Type	Description
`status`	`Number`	Error status code.
`error`	`String`	Error message.

Example 1

```HTTP
DELETE /api/pipeline/7 HTTP/1.1
Authorization: Basic YWRtaW46cGFzc3dvcmQ=
Host: 127.0.0.1
```

```HTTP
HTTP/1.1 404 Not Found
Vary: Origin
Vary: Access-Control-Request-Method
Vary: Access-Control-Request-Headers
Content-Type: application/json
Content-Length: 71

{
  "status" : 404,
  "error" : "Pipeline with id = 7 doesn't exist."
}
```

Example 2

```HTTP
POST /api/pipeline HTTP/1.1
Content-Type: application/json
Authorization: Basic YWRtaW46cGFzc3dvcmQ=
Content-Length: 19
Host: 127.0.0.1

{
  "source" : {}
}
```

```HTTP
HTTP/1.1 400 Bad Request
Vary: Origin
Vary: Access-Control-Request-Method
Vary: Access-Control-Request-Headers
Content-Type: application/json
Content-Length: 99

{
  "status" : 400,
  "error" : "Missing required property 'address'. (in: '{}.inputs.0.source')"
}
```

Example 3

HTTP

```
POST /api/pipeline HTTP/1.1
Content-Type: application/json
Authorization: Basic YWRtaW46cGFzc3dvcmQ=
Content-Length: 160
Host: 127.0.0.1

{
  "source" : {
    "address" : {
      "url": "rtp://0.0.0.0:5000:lo"
    },
    "streamType" : "MPEG_TS",
    "complianceWith" : "AUTO"
  },
  "video": {}
}
```

HTTP

```
HTTP/1.1 400 Bad Request
Vary: Origin
Vary: Access-Control-Request-Method
Vary: Access-Control-Request-Headers
Content-Type: application/json
Content-Length: 96

{
  "status" : 400,
  "error" : "Missing required property 'depth'. (in: '{}.inputs.0.video')"
}
```

Last updated 2024-09-04 19:36:56 UTC