API documentation

Version Unspecified

<< Back to UI (https://172.22.10.159/ui/dashboard)

Table of Contents

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 **`"0.29"`**.
`id`	`Integer`	Unique identifier of this pipeline.
`name`	`String`	Name of this pipeline displayed in UI.
`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.
`runningState`	`String`	Policy to restrict automatic pipeline loading after restart. Allowed values are **`"INACTIVE"`** or **`"ACTIVE"`**.
`inputs`	`Array`	Array of Input Stream objects each describing particular input stream.
`outputs`	`Array`	Array of Output Streams objects each describing particular output stream.
`uploadedLogos`	`Array`	Array of Uploaded Logo objects containing configuration of logos.

Pipeline example:

```JSON
{
  "version" : "0.29",
  "id" : 0,
  "uniqueId" : "5efb5342-1cc1-4f5d-9c0b-3568ba5a3954",
  "name" : "Pipeline 1",
  "runningState" : "INACTIVE",
  "inputs" : [ {
    "id" : 0,
    "uniqueId" : "8cb3ab3d-8f32-4965-b085-616b40e3f19e",
    "source" : {
      "address" : {
        "url" : "srt://127.0.0.1:5001:lo",
        "srt" : {
          "mode" : "CALLER",
          "latency" : 50,
          "passphrase" : "",
          "streamID" : "#!::u=admin,t=file,m=publish,r=results.csv",
          "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" : "",
          "redundancy" : [ ]
        }
      }
    },
    "videoConfigurationMode" : "MANUAL_AFTER_DETECTION",
    "video" : {
      "uniqueId" : "75fc3294-1760-4375-95c8-f395d5f0df38",
      "format" : "J2K",
      "width" : 1920,
      "height" : 1080,
      "sampling" : "4:2:2",
      "depth" : 10,
      "scanRate" : "50",
      "scan" : "INTERLACED",
      "fieldOrder" : "TOP_FIELD_FIRST",
      "pid" : 511,
      "filters" : {
        "yOffset" : 16,
        "rOffset" : 1,
        "gOffset" : 2,
        "bOffset" : 3,
        "yGain" : 0.2,
        "rGain" : 0.01,
        "gGain" : 0.02,
        "bGain" : 0.03,
        "rgbClip" : true,
        "hue" : 4.5,
        "saturation" : 0.5
      }
    },
    "audioConfigurationMode" : "MANUAL_AFTER_DETECTION",
    "audio" : [ {
```

```
      "id" : 0,
      "format" : "AES3",
      "rate" : 48000,
      "pid" : 661,
      "channels" : 2,
      "filters" : {
        "volumeGain" : 2.5
      },
      "passthrough" : false
    } ],
    "captions" : [ {
      "id" : 0,
      "format" : "CEA_708_SMPTE_2038",
      "pid" : 662
    } ],
    "metadata" : [ {
      "id" : 0,
      "format" : "UNKNOWN",
      "pid" : 663
    } ]
  } ],
  "outputs" : [ {
    "id" : 0,
    "uniqueId" : "2f6fe0fb-4374-4970-817f-56a5e509794e",
    "name" : "Output Stream 1",
    "streamType" : "MPEG_TS",
    "sink" : {
      "addresses" : [ {
        "uniqueId" : "f78915d5-0373-496c-9fc1-9d4c899f4e8f",
        "url" : "srt://0.0.0.0:6001:lo",
        "srt" : {
          "mode" : "LISTENER",
          "latency" : 120,
          "overhead" : 25,
          "encryption" : "AES_256",
          "passphrase" : "0123456789",
          "redundancy" : [ ]
        }
      } ]
    },
    "ts" : {
      "service" : {
        "transportStreamId" : null,
        "programNumber" : null,
        "name" : "SERVICE",
        "providerName" : "PROVIDER"
      },
      "pcrStreamLocation" : "SEPARATE",
      "ebpLength" : 5,
      "ebpLengthUnit" : "SECONDS",
      "bitrate" : 11000000,
      "outputComplianceWith" : "STANDARD"
    },
    "video" : {
      "uniqueId" : "610eea9b-d2a0-44fe-8b8c-c160d82c71dc",
      "format" : "H264",
      "width" : 720,
      "height" : 576,
      "sampling" : "4:2:0",
      "depth" : 8,
      "scan" : "INTERLACED",
      "fieldOrder" : "TOP_FIELD_FIRST",
      "pid" : 670,
      "scanRateType" : "SAME_AS_INPUT",
      "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
      },
      "h264" : {
        "coding" : "CABAC",
        "profile" : "HIGH",
        "gopSize" : 120,
        "bCount" : 0
      }
    },
    "audio" : [ {
      "id" : 0,
      "format" : "AAC_ADTS",
      "rate" : 48000,
      "pid" : 671,
      "name" : "Output audio 1",
      "references" : [ {
        "inputId" : 0,
        "id" : 0,
        "channel" : 0
      } ],
```

```
      "channels" : 2,
      "bitrate" : 1000000
    } ],
    "captions" : [ {
      "id" : 0,
      "format" : "CEA_708_SMPTE_2038",
      "pid" : 672,
      "reference" : {
        "inputId" : 0,
        "id" : 0
      }
    } ],
    "metadata" : [ {
      "id" : 0,
      "format" : "UNKNOWN",
      "pid" : 673,
      "reference" : {
        "inputId" : 0,
        "id" : 0
      }
    } ]
  } ],
  "uploadedLogos" : [ {
    "id" : 0,
    "name" : "logo.png",
    "codestream" : "iVBORw0KGgoAAAANSUhEUgAAAAEAAAABCAIAAACQd1PeAAAAD0lEQVQIHQEEAPv/AP///wX+Av4DfRnGAAAAAElFTkSuQmCC"
  } ]
}
```


Input Stream

JSON object containing configuration of input stream.

Path	Type	Description
`id`	`Integer`	Unique identifier of the input stream, referenced in output definition.
`uniqueId`	`String`	Unique pipeline input UUID. Randomly generated when a user creates the input, removed on the input removal. Can't be downloaded with pipeline preset. Survives stopping of pipeline and restart of service/machine.
`source.address`	`Source address`	Configuration of the input stream source address. See Input Source Address.
`source.streamType`	`String`	Type of input stream. Allowed values are **`"MPEG_TS"`**, **`"RTMP"`**, **`"SDI"`**, **`"NDI"`**, **`"HLS"`**, **`"SMPTE_2110"`**.
`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`	`Source address`	Optionally can be present when **`streamType`** is **`"MPEG_TS"`**. Input stream to receive PCR from. See Input Source Address but note that only UDP, RTP and SRT stream types are allowed.
`videoConfigurationMode`	`String`	Mode of video configuration. Allowed values are **`"MANUAL_AFTER_DETECTION"`** or **`"AUTORECONFIG"`**.
`video`	`Input Video`	Configuration of the input video. See Input Video.
`audioConfigurationMode`	`String`	Mode of audio configuration. Allowed values are **`"MANUAL_AFTER_DETECTION"`**, **`"MANUAL_FIXED_CHANNELS"`** or **`"AUTORECONFIG"`**.
`audio`	`Array`	Array of Input Audio describing the input audio streams.
`captions`	`Array`	Array of Input Captions describing the input captions.
`metadata`	`Array`	Array of Input Metadata describing the input metadata.

Input stream example:

```json
{
  "id" : 0,
  "uniqueId" : "8cb3ab3d-8f32-4965-b085-616b40e3f19e",
  "source" : {
    "address" : {
      "url" : "srt://127.0.0.1:5001:lo",
      "srt" : {
        "mode" : "CALLER",
        "latency" : 50,
        "passphrase" : "",
        "streamID" : "#!::u=admin,t=file,m=publish,r=results.csv",
        "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" : "",
        "redundancy" : [ ]
      }
    }
  },
  "videoConfigurationMode" : "MANUAL_AFTER_DETECTION",
  "video" : {
    "uniqueId" : "75fc3294-1760-4375-95c8-f395d5f0df38",
    "format" : "J2K",
    "width" : 1920,
    "height" : 1080,
    "sampling" : "4:2:2",
    "depth" : 10,
    "scanRate" : "50",
    "scan" : "INTERLACED",
    "fieldOrder" : "TOP_FIELD_FIRST",
    "pid" : 511,
    "filters" : {
      "yOffset" : 16,
      "rOffset" : 1,
      "gOffset" : 2,
      "bOffset" : 3,
      "yGain" : 0.2,
      "rGain" : 0.01,
      "gGain" : 0.02,
      "bGain" : 0.03,
      "rgbClip" : true,
      "hue" : 4.5,
      "saturation" : 0.5
    }
  },
  "audioConfigurationMode" : "MANUAL_AFTER_DETECTION",
  "audio" : [ {
    "id" : 0,
    "format" : "AES3",
    "rate" : 48000,
    "pid" : 661,
    "channels" : 2,
    "filters" : {
      "volumeGain" : 2.5
    },
    "passthrough" : false
  } ],
  "captions" : [ {
    "id" : 0,
    "format" : "CEA_708_SMPTE_2038",
    "pid" : 662
  } ],
  "metadata" : [ {
    "id" : 0,
    "format" : "UNKNOWN",
    "pid" : 663
  } ]
}
```

Input Source Address

JSON object containing configuration of input stream source address.

Path	Type	Description
`url`	`String`	URL specifying stream type and its connection parameters. See Input Source Address URL
`srt`	`Object`	Specific information for SRT protocol.
`srt.mode`	`String`	Mode of SRT connection. Allowed values are **`"CALLER"`** or **`"LISTENER"`**.
`srt.latency`	`Integer`	This is the buffer time (ms) held at the receiver's side to allow for packet delivery/retransmission. The connection uses the highest latency value proposed by either the sender or receiver. Try starting with a latency roughly four times your network's Round- Trip Time (RTT). Allowed values [0..2000]

		Trip Time (RTT). Allowed values [0, 2000].
`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.
`srt.passphrase`	`String`	Passphrase for encrypted SRT connection. Allowed phrase has printable character with length: [10-79]. To disable encryption leave this field empty.
`srt.redundancy`	`Array`	Optional array of URL string values representing redundant paths for SRT caller.
`ndi`	`Object`	Specific information for NDI protocol.
`ndi.ndiGroup`	`String`	Optional comma separated list of groups. For instance **`"cameras,studio1,10am show"`** would place the output in the three groups named. The default group name is **`"public"`**.
`hls`	`Object`	Specific information for HLS protocol.
`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.

Input source address example:

```JSON
{
  "url" : "srt://127.0.0.1:5001:lo",
  "srt" : {
    "mode" : "CALLER",
    "latency" : 50,
    "passphrase" : "",
    "streamID" : "#!::u=admin,t=file,m=publish,r=results.csv",
    "redundancy" : [ "srt://127.0.0.1:5002:lo" ]
  }
}
```

Input Source Address URL

It should be in one of the following formats according to the stream type:

Stream Type	**URL**
RTP/UDP	**`"rtp://[<mc-src-ip-address>@]<ip-address>:<port>[:<interface-id>]"`**
UDP	**`"udp://[<mc-src-ip-address>@]<ip-address>:<port>[:<interface-id>]"`**
SRT	**`"srt://<ip-address>:<port>[:<interface-id>]"`**
RTMP	**`"rtmp://<stream-key>"`**
HLS	**`"hls://<http|https url>"`**
SDI	**`"sdi://<sdi-device-id>:<port-index>"`**
NDI	**`"ndi://<ndi-source>"`**
SMPTE 2110	**`"smpte2110://"`**

Where optional parts are enclosed with **`[]`** and

- **`<interface-id>`** should be the identifier of one of the available network interfaces
- **`<sdi-device-id>`** should be the identifier of one of the available SDI devices.
- **`<ndi-source>`** consists of hostname and NDI name formatted like **`"hostname (ndi-name)"`**. E.g. source **`"TRANSCODER 0 (Studio 0)"`** consists of hostname **`"TRANSCODER 0"`** and NDI name **`"Studio 0"`**

Note that no parameters can be provided in the URL for the SMPTE 2110 and they must be set separately through the NMOS REST API.

Examples:

- **`"rtp://127.0.0.1:5000:lo"`**
- **`"udp://127.0.0.1:5001:lo"`**
- **`"srt://0.0.0.0:5000"`**
- **`"rtmp://c4fdd994-0172-4999-8eb5-fc7714b4f488"`**
- **`"hls://http://source.net/hls/test.m3u8"`**
- **`"sdi://aja-sdi-0:0"`**
- **`"ndi://TRANSCODER 0 (Studio 0)"`**

Input Video

JSON object containing configuration of input video.

Path	Type	Description
`uniqueId`	`String`	Unique id of the video stream. Randomly generated when a user creates the input, removed on the input removal. Can't be downloaded with pipeline preset. Survives stopping of pipeline and restart of service/machine.
`pid`	`Integer`	Elementary stream identifier.
`format`	`String`	Format of input video. Allowed values are `"RAW"`, `"H262"`, `"H264"`, `"H265"`, `"J2K"`, `"JXS"`.
`width`	`Integer`	Video frame width.
`height`	`Integer`	Video frame height.
`sampling`	`String`	Video frame subsampling. Allowed values are `"4:2:2"`, `"4:2:0"` or `"4:4:4"`.
`depth`	`Integer`	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.
`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:

```
{
  "uniqueId" : "75fc3294-1760-4375-95c8-f395d5f0df38",
  "format" : "J2K",
  "width" : 1920,
  "height" : 1080,
  "sampling" : "4:2:2",
  "depth" : 10,
  "scanRate" : "50",
  "scan" : "INTERLACED",
  "fieldOrder" : "TOP_FIELD_FIRST",
  "pid" : 511,
  "filters" : {
    "yOffset" : 16,
    "rOffset" : 1,
    "gOffset" : 2,
    "bOffset" : 3,
    "yGain" : 0.2,
    "rGain" : 0.01,
    "gGain" : 0.02,
    "bGain" : 0.03,
    "rgbClip" : true,
    "hue" : 4.5,
    "saturation" : 0.5
  }
}
```

Input Audio

JSON object containing configuration of a single input audio stream.

Path	Type	Description
`id`	`Integer`	Unique identifier of input audio.
`pid`	`Integer`	Elementary stream identifier in case input stream type is `"MPEG_TS"`.
`format`	`String`	Format of input audio. Allowed values are: `"AC3"`, `"EAC3"`, `"AES3"`, `"AAC_ADTS"`, `"AAC_LATM"`, `"AAC_HE_ADTS"`, `"AAC_HE_LATM"`, `"AAC_HE_V2_ADTS"`, `"AAC_HE_V2_LATM"`, `"MPEG2"`, `"RAW_FLTP"`, `"AES3_DOLBY_E"`, `"RAW_S16"`, `"RAW_S24"`
`rate`	`Integer`	Audio sampling rate in Hz.
`channels`	`Integer`	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%`.
`delay`	`Integer`	Delay of the audio in ms.
`bitsPerSample`	`Integer`	How many bits are used per sample.
`passthrough`	`Boolean`	Specifies that audio is sent pass-through (if possible).

Example of a input audio stream configuration:

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

Input Captions

JSON object containing configuration of input captions.

Path	Type	Description
`id`	`Integer`	Unique identifier of input captions.
`pid`	`Integer`	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,
  "format" : "CEA_708_SMPTE_2038",
  "pid" : 662
}
```

Input Metadata

JSON object containing configuration of input metadata.

Path	Type	Description
`id`	`Integer`	Unique identifier of input metadata.
`pid`	`Integer`	Elementary stream identifier.
`format`	`String`	Format of input metadata. Allowed values are `"SCTE_104"`, `"SCTE_35"`, `"SDI_ANC"`, `"SMPTE_2038"`, `"DVB_EIT"`, `"AIT"`, `"EBU_TELETEXT"`, `"UNKNOWN"`.

Input metadata example:

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

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.
`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.
`streamType`	`String`	Type of output stream. Allowed values are `"MPEG_TS"`, `"RTMP"`, `"SDI"`, `"NDI"`, `"HLS"`, `"SMPTE_2110"`, `"MPEG_TS_PCR_ONLY"`.
`sink.addresses`	`Array`	Configuration of the output stream destination addresses. See Output Destination

		Configuration of the output stream destination addresses. See Output Destination Address.
`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.service.transportStreamId`	`Integer`	A unique 16-bit identifier for this Transport Stream. Used in PAT and SDT tables to distinguish streams within a network. If omitted a default unspecified ID will be used.
`ts.service.programNumber`	`Integer`	A 16-bit identifier for the service within the Transport Stream. Called 'service_id' in SDT/PMT and 'program_number' in PAT. If omitted the value is passed through from the input (if MPEG-TS), or a default unspecified value is used for non-MPEG-TS inputs.
`ts.bitrate`	`Integer`	Maximum total bitrate of output stream in bits per second.
`ts.pcrStreamLocation`	`String`	Placement of the PCR. Allowed values are **`"SEPARATE"`**, **`"VIDEO"`**. If omitted **`"SEPARATE"`** is used as default.
`ts.ebpLength`	`Integer`	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.rcMode`	`String`	Rate control mode. Allowed values are **`"CBR"`**, **`"VBR"`** or **`"CVBR"`**. If omitted **`"CBR"`** is used as default.
`ts.outputComplianceWith`	`String`	MPEG-TS standard or Evertz compliance. Allowed values are **`"STANDARD"`** or **`"EVERTZ"`**.
`smpte2110Output`	`Object`	Information specific to the SMPTE 2110 output streams.
`smpte2110Output.flowType`	`String`	Type of the 2110 stream flow. Allowed values are **`"VIDEO"`**, **`"AUDIO"`** and **`"METADATA"`**.
`smpte2110Output.audio`	`Object`	Information specific to the SMPTE 2110 audio output streams.
`smpte2110Output.audio.rate`	`Integer`	Audio sample rate. Allowed values are **`48000`**, **`96000`**.
`smpte2110Output.audio.ptime`	`Integer`	Duration of the audio contained in each packet in [µs]. Allowed values are **`125`**, **`1000`**.
`smpte2110Output.audio.bitDepth`	`Integer`	Audio sample bit depth. Allowed values are **`16`**, **`24`**.
`video`	`Output Video`	Configuration of output video. See Output Video.
`audio`	`Array`	Array of Output Audio describing the configuration of output audio streams.
`captions`	`Array`	Array of Output Captions describing the configuration of output captions.
`metadata`	`Array`	Configuration of output metadata. See Output Metadata.

Output stream example:

```JSON
{
  "id" : 0,
  "uniqueId" : "2f6fe0fb-4374-4970-817f-56a5e509794e",
  "name" : "Output Stream 1",
```

```
  "streamType" : "MPEG_TS",
  "sink" : {
    "addresses" : [ {
      "uniqueId" : "f78915d5-0373-496c-9fc1-9d4c899f4e8f",
      "url" : "srt://0.0.0.0:6001:lo",
      "srt" : {
        "mode" : "LISTENER",
        "latency" : 120,
        "overhead" : 25,
        "encryption" : "AES_256",
        "passphrase" : "0123456789",
        "redundancy" : [ ]
      }
    } ]
  },
  "ts" : {
    "service" : {
      "transportStreamId" : null,
      "programNumber" : null,
      "name" : "SERVICE",
      "providerName" : "PROVIDER"
    },
    "pcrStreamLocation" : "SEPARATE",
    "ebpLength" : 5,
    "ebpLengthUnit" : "SECONDS",
    "bitrate" : 11000000,
    "outputComplianceWith" : "STANDARD"
  },
  "video" : {
    "uniqueId" : "610eea9b-d2a0-44fe-8b8c-c160d82c71dc",
    "format" : "H264",
    "width" : 720,
    "height" : 576,
    "sampling" : "4:2:0",
    "depth" : 8,
    "scan" : "INTERLACED",
    "fieldOrder" : "TOP_FIELD_FIRST",
    "pid" : 670,
    "scanRateType" : "SAME_AS_INPUT",
    "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
    },
    "h264" : {
      "coding" : "CABAC",
      "profile" : "HIGH",
      "gopSize" : 120,
      "bCount" : 0
    }
  },
  "audio" : [ {
    "id" : 0,
    "format" : "AAC_ADTS",
    "rate" : 48000,
    "pid" : 671,
    "name" : "Output audio 1",
    "references" : [ {
      "inputId" : 0,
      "id" : 0,
      "channel" : 0
    } ],
    "channels" : 2,
    "bitrate" : 1000000
  } ],
  "captions" : [ {
    "id" : 0,
    "format" : "CEA_708_SMPTE_2038",
    "pid" : 672,
    "reference" : {
      "inputId" : 0,
      "id" : 0
    }
  } ],
  "metadata" : [ {
    "id" : 0,
    "format" : "UNKNOWN",
    "pid" : 673,
    "reference" : {
      "inputId" : 0,
      "id" : 0
    }
  } ]
}
```


Output Destination Address

JSON object containing configuration of output stream destination address.

Path	Type	Description
`url`	`String`	URL specifying stream type and its connection parameters. See Output Destination Address URL.
`uniqueId`	`String`	Unique destination address UUID. Randomly generated when a user creates the destination, removed when destination is deleted. Survives stopping of pipeline and restart of service/machine.
`srt`	`Object`	Specific information for SRT protocol.
`srt.mode`	`String`	Mode of SRT connection. Allowed values are **`"CALLER"`** or **`"LISTENER"`**.
`srt.overhead`	`Integer`	Bandwidth overhead above output rate in percents. Allowed values: [5, 100].
`srt.latency`	`Integer`	This is the buffer time (ms) held at the receiver's side to allow for packet delivery/retransmission. The connection uses the highest latency value proposed by either the sender or receiver. Try starting with a latency roughly four times your network's Round-Trip Time (RTT). Allowed values [0, 2000].
`srt.streamID`	`String`	Stream ID for SRT CALLER mode.
`srt.encryption`	`String`	Settings of key length for AES encryption. Allowed values are **`"AES_DEFAULT"`**, **`"AES_128"`**, **`"AES_192"`** or **`"AES_256"`**.
`srt.passphrase`	`String`	Passphrase for encrypted SRT connection. Allowed phrase has printable character with length: [10, 79]. To disable encryption leave this field empty.
`srt.redundancy`	`Array`	Optional list of URL string values representing redundant paths for SRT caller.
`ndi`	`Object`	Specific information for NDI protocol.
`ndi.ndiGroup`	`String`	Optional group name of NDI.
`hls`	`Object`	Specific information for HLS protocol.
`hls.segmentLength`	`Integer`	Specify duration of the HLS segments (s). Possible values are [1, 300], default is 6.
`hls.segmentCount`	`Integer`	Specify how many HLS segments are playable in history. Possible values are [3, 600], default is 10.
`hls.deleteThreshold`	`Integer`	Specify how many HLS segments are available after playlist dereference. Possible values are [1, 300], default is 1 (nothing available).
`smpte2110`	`Object`	Specific information for SMPTE 2110 protocol.
`smpte2110.primary`	`Object`	Configuration of the primary leg.
`smpte2110.primary.rtpEnabled`	`Boolean`	RTP transmission active/inactive.
`smpte2110.primary.sourcePort`	`Integer`	Source port for RTP packets.
`smpte2110.redundant`	`Array`	Configuration of redundant leg(s). Currently just 1 redundant leg is supported.
`smpte2110.redundant[].url`	`String`	URL specifying connection configuration of the leg. Use the **`smpte2110://…`** format
`smpte2110.redundant[].rtpEnabled`	`Boolean`	RTP transmission active/inactive.
`smpte2110.redundant[].sourcePort`	`Integer`	Source port for RTP packets.

Output destination address example:

```
{
  "uniqueId" : "f78915d5-0373-496c-9fc1-9d4c899f4e8f",
  "url" : "srt://0.0.0.0:6001:lo",
  "srt" : {
    "mode" : "LISTENER",
    "latency" : 120,
    "overhead" : 25,
    "encryption" : "AES_256",
    "passphrase" : "0123456789",
    "redundancy" : [ ]
  }
}
```

Output Destination Address URL

It should be in one of the following formats according to the stream type:

Stream Type	URL
RTP	**"rtp://<ip-address>:<port>:<interface-id>"**
UDP	**"udp://<ip-address>:<port>:<interface-id>"**
SRT	**"srt://<ip-address-or-web-url>:<port>:<interface-id>"**
RTMP(S)	**"rtmp[s]://[user:password@]<domain-name/ip-address>[:<port>]/path/streaming-key"**
HLS	**"hls://<playlist-name>"**
SDI	**"sdi://<sdi-device-id>:<port-index>"**
NDI	**"ndi://<ndi-name>"**
SMPTE 2110	**"smpte2110://<ip-address>:<port>:<interface-id>"**

Where optional parts are enclosed with **[]** and

- **<interface-id>** should be the identifier of one of the available network interfaces.
- **<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"**
- **"srt://0.0.0.0:5050"**
- **"rtmp://user:password@example.com:1935/live/abc-123"**
- **"hls://stream-playlist"**
- **"sdi://aja-sdi-0:0"**
- **"ndi://TRANSCODER 0 (Studio 0)"**
- **"smpte2110://239.0.0.1:5004:enp3s0f0np0"**

Output Video

JSON object containing configuration of output video.

Path	Type	Description
uniqueId	String	Unique id of the video stream. 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.
pid	Number	Elementary stream identifier.
format	String	Format of output video. Allowed values are **"RAW"**, **"H262"**, **"H264"**, **"H265"**, **"J2K"**, **"JXS"**.
width	Integer	Video frame width.
height	Integer	Video frame height.
crop	Object	Configuration of cropping and/or padding.
crop.fit	String	Allowed values are **"SCALE"**, **"CROP"** or **"PAD"**. Use **"SCALE"** to resize input frame size to configured output frame size and

		Use **"SCALE"** to resize input frame size to configured output frame size and change aspect ratio when it differs. Use **"CROP"** to resize 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 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 the 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 the 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 the **`uploadedLogos`** section of the pipeline configuration.
`logo.horizontalPosition`	`Integer`	Horizontal position of the logo in number of pixels from the left side.
`logo.verticalPosition`	`Integer`	Vertical position of the logo in number of pixels from the top side.
`logo.horizontalScaling`	`Number`	Horizontal scaling of the logo in percents of the original logo width.
`logo.verticalScaling`	`Number`	Vertical scaling of the logo in percents of the original logo height.
`sampling`	`String`	Video frame subsampling. Allowed values are **`"4:2:2"`**, **`"4:2:0"`** or **`"4:4:4"`**.
`depth`	`Integer`	Video frame samples bit depth.
`scanRateType`	`String`	Scan rate type can be set to **`"SAME_AS_INPUT"`**, **`"HALF_THE_INPUT"`** and **`"CONVERT_TO"`**. It takes the input scan rate and sets the output scan rate based on this option.
`scanRate`	`String`	Frame rate for progressive scan video (frames per second) or field rate for interlaced scan video (fields per second). Only in case that `scanRateType` is set to **`"CONVERT_TO"`**.
`scan`	`String`	Video scan type. Allowed values are **`"PROGRESSIVE"`** (for progressive scan video) or **`"INTERLACED"`** (for interlaced scan video).
`fieldOrder`	`String`	Field order of an interlaced video. Allowed values are **`"TOP_FIELD_FIRST"`** or **`"BOTTOM_FIELD_FIRST"`**.
`bitrate`	`Integer`	Maximum bitrate of the output video in bits per second. Value **`0`** enables video passthrough (in such case output video parameters have to match input video parameters exactly).
`captions`	`Object`	Captions to be wrapped inside the output video.
`captions.reference`	`Object`	Referenced of the input captions.

captions.reference.inputId	Integer	Identifier of the input stream containing the captions.
captions.reference.id	Integer	Identifier of referenced input captions.
j2k	Object	Encoding settings for J2K video. Valid only when `format` is `"J2K"`.
jxs	Object	Encoding settings for JXS video. Valid only when `format` is `"JXS"`.
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	Integer	GOP size in number of frames. Allowed values are [1, 300].
h262.bCount	Integer	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"`.
h264.gopSize	Integer	GOP size in number of frames. Allowed values are [1, 300].
h264.bCount	Integer	Number of B frames. Allowed values are [0, 4].
h265	Object	Encoding settings for H.265 video. Valid only when `format` is `"H265"`.
h265.profile	String	Allowed values are `"MAIN"` or `"MAIN10"`.
h265.gopSize	Integer	GOP size in number of frames. Allowed values are [1, 300].
h265.bCount	Integer	Number of B frames. Allowed values are [0, 4].

Output video example:

```json
{
  "uniqueId" : "610eea9b-d2a0-44fe-8b8c-c160d82c71dc",
  "format" : "H264",
  "width" : 720,
  "height" : 576,
  "sampling" : "4:2:0",
  "depth" : 8,
  "scan" : "INTERLACED",
  "fieldOrder" : "TOP_FIELD_FIRST",
  "pid" : 670,
  "scanRateType" : "SAME_AS_INPUT",
  "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
  },
  "h264" : {
    "coding" : "CABAC",
    "profile" : "HIGH",
    "gopSize" : 120,
    "bCount" : 0
  }
}
```

Output Audio

JSON object containing configuration of output audio stream.

Path	Type	Description
id	Integer	Unique identifier of output audio.
name	String	Name of this output audio displayed in UI.
pid	Integer	Elementary stream identifier in case output stream is `"MPEG_TS"`. Allowed values are [1

		Elementary stream identifier in case output stream is `"MPEG_TS"`. Allowed values are [1, 8190].
`format`	`String`	Format of output audio. Allowed values are `"AC3"`, `"EAC3"`, `"AES3"`, `"AAC_ADTS"`, `"AAC_LATM"`, `"AAC_HE_ADTS"`, `"AAC_HE_LATM"`, `"AAC_HE_V2_ADTS"`, `"AAC_HE_V2_LATM"`, `"MPEG2"`, `"RAW_FLTP"`, `"AES3_DOLBY_E"`, `"RAW_S16"`, `"RAW_S24"`.
`rate`	`Integer`	Audio sampling rate in Hz.
`references[]`	`Array`	Array of references to input audio channels mixed in this audio stream.
`references[].inputId`	`Integer`	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`	`Integer`	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. Allowed values are [1,16].
`bitrate`	`Integer`	Maximum bitrate of output audio in bits per second.
`bitDepth`	`Integer`	Audio sample bit depth.

Output audio example:

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

Output Captions

JSON object containing configuration of output captions.

Path	Type	Description
`id`	`Integer`	Unique identifier of the output captions.
`pid`	`Integer`	Elementary stream identifier. Allowed values are [16, 8190].
`format`	`String`	Format of output captions. Allowed values are `"CEA_608_SMPTE_2038"`, `"CEA_708_SMPTE_2038"`, `"CEA_608_SDI"` and `"CEA_708_SDI"`.
`reference`	`Object`	Referenced input captions.
`reference.inputId`	`Integer`	Identifier of input stream containing the referenced captions.
`reference.id`	`Integer`	Identifier of referenced input captions.

Output captions example:

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

Output Metadata

JSON object containing configuration of output metadata.

Path	Type	Description
`id`	`Integer`	Unique identifier of output metadata.
`pid`	`Integer`	Elementary stream identifier. Allowed values are [16, 8190].
`format`	`String`	Format of output metadata. Allowed values are `"SCTE_104"`, `"SCTE_35"`, `"SDI_ANC"`, `"SMPTE_2038"`, `"DVB_EIT"`, `"AIT"`, `"EBU_TELETEXT"`, `"UNKNOWN"`.
`reference`	`Object`	Referenced input metadata.
`reference.inputId`	`Integer`	Identifier of the input stream containing the referenced metadata.
`reference.id`	`Integer`	Identifier of referenced input metadata.

Output metadata example:

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

Uploaded Logo

JSON object containing uploaded logo.

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 logo 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

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.
inputs	Array	Array of Input Stream States.
outputs	Array	Array of Output Stream States.

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" : "5efb5342-1cc1-4f5d-9c0b-3568ba5a3954",
  "inputs" : [ {
    "id" : 0,
    "uniqueId" : "8cb3ab3d-8f32-4965-b085-616b40e3f19e",
    "status" : "ERROR",
    "name" : "rtp://0.0.0.0:5000:lo",
    "description" : "JPEG 2000 1920x1080 4:4:4 10-bit",
    "elementaryStreams" : [ {
      "type" : "UNKNOWN",
      "id" : 0,
      "pid" : 501,
      "state" : "NEW",
      "messages" : [ ]
    }, {
      "type" : "VIDEO",
      "id" : 0,
      "pid" : 502,
      "state" : "NEW",
      "messages" : [ ],
      "uniqueId" : null,
      "format" : "J2K",
      "width" : 1920,
      "height" : 1080,
      "depth" : 10,
      "sampling" : "4:2:2",
      "scanRate" : "60000/1001",
      "scan" : "INTERLACED",
      "fieldOrder" : "BOTTOM_FIELD_FIRST"
    }, {
      "type" : "AUDIO",
      "id" : 0,
      "pid" : 503,
      "state" : "NEW",
      "messages" : [ {
        "severity" : "ERROR",
        "message" : "This audio stream wasn't detected in source stream."
      } ],
      "format" : "AES3",
      "channelCount" : 2,
      "realChannelCount" : null,
      "rate" : 48000,
      "bitrate" : 235000
    }, {
      "type" : "AUDIO",
      "id" : 1,
      "pid" : 499,
      "state" : "NEW",
      "messages" : [ {
        "severity" : "ERROR",
        "message" : "This audio stream wasn't detected in source stream."
      } ],
      "format" : "AES3_DOLBY_E",
      "channelCount" : 2,
      "realChannelCount" : null,
      "rate" : 48000,
      "bitrate" : 128000
    }, {
      "type" : "CAPTIONS",
      "id" : 0,
      "pid" : 504,
      "state" : "NEW",
      "messages" : [ ],
      "format" : "CEA_708_SMPTE_2038"
    }, {
      "type" : "METADATA",
      "id" : 0,
      "pid" : 505,
      "state" : "NEW",
      "messages" : [ ],
      "format" : "UNKNOWN",
      "info" : "INFO1"
    }, {
      "type" : "METADATA",
      "id" : 1,
      "pid" : 506,
```

```
      "state" : "NEW",
      "messages" : [ ],
      "format" : "SMPTE_2038",
      "info" : "BLABLA"
    } ],
    "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" : "2f6fe0fb-4374-4970-817f-56a5e509794e",
    "destinationUniqueId" : null,
    "status" : "ERROR",
    "messages" : [ {
      "severity" : "ERROR",
      "message" : "Output Stream 1: Not producing any output."
    } ],
    "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,
    "latencyAddition" : 0,
    "droppedFps" : 0.0
  } ]
}
```


Input Stream State

JSON object containing input stream state.

Path	Type	Description
`id`	`Number`	Unique identifier of this input stream.
`uniqueId`	`String`	Unique pipeline input UUID. Randomly generated when a user creates the input, removed on the input 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 input stream.
`description`	`String`	Input stream description.
`elementaryStreams`	`Array`	Array of detected/configured Elementary Streams.
`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`	Array of Stream State Messages for this input stream.

Input Stream State example:

```
{
  "id" : 0,
  "uniqueId" : "8cb3ab3d-8f32-4965-b085-616b40e3f19e",
  "status" : "ERROR",
  "name" : "rtp://0.0.0.0:5000:lo",
  "description" : "JPEG 2000 1920x1080 4:4:4 10-bit",
  "elementaryStreams" : [ {
    "type" : "UNKNOWN",
    "id" : 0,
    "pid" : 501,
    "state" : "NEW",
    "messages" : [ ]
  }, {
    "type" : "VIDEO",
    "id" : 0,
    "pid" : 502,
    "state" : "NEW",
    "messages" : [ ],
    "uniqueId" : null,
    "format" : "J2K",
    "width" : 1920,
    "height" : 1080,
    "depth" : 10,
    "sampling" : "4:2:2",
    "scanRate" : "60000/1001",
    "scan" : "INTERLACED",
    "fieldOrder" : "BOTTOM_FIELD_FIRST"
  }, {
    "type" : "AUDIO",
    "id" : 0,
    "pid" : 503,
    "state" : "NEW",
    "messages" : [ {
      "severity" : "ERROR",
      "message" : "This audio stream wasn't detected in source stream."
    } ],
    "format" : "AES3",
    "channelCount" : 2,
    "realChannelCount" : null,
    "rate" : 48000,
    "bitrate" : 235000
  }, {
    "type" : "AUDIO",
    "id" : 1,
    "pid" : 499,
    "state" : "NEW",
    "messages" : [ {
      "severity" : "ERROR",
      "message" : "This audio stream wasn't detected in source stream."
    } ],
    "format" : "AES3_DOLBY_E",
    "channelCount" : 2,
    "realChannelCount" : null,
    "rate" : 48000,
    "bitrate" : 128000
  }, {
    "type" : "CAPTIONS",
    "id" : 0,
    "pid" : 504,
    "state" : "NEW",
    "messages" : [ ],
    "format" : "CEA_708_SMPTE_2038"
  }, {
    "type" : "METADATA",
    "id" : 0,
    "pid" : 505,
    "state" : "NEW",
    "messages" : [ ],
    "format" : "UNKNOWN",
    "info" : "INFO1"
  }, {
    "type" : "METADATA",
    "id" : 1,
    "pid" : 506,
    "state" : "NEW",
    "messages" : [ ],
    "format" : "SMPTE_2038",
    "info" : "BLABLA"
  } ],
  "decodedFps" : 0.0,
  "decodingLatency" : 0.0,
  "invalidFps" : 0.0,
  "messages" : [ {
    "severity" : "ERROR",
    "message" : "rtp://0.0.0.0:5000:lo: input stream #0 error."
  } ]
}
```


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

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.
destinationUniqueId	Null	Unique destination address UUID. Randomly generated when a user creates the destination, removed when destination is deleted. 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 of the stream destination.
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.
droppedFps	Number	Number of dropped progressive frames/interlaced fields per second for this output stream (average over the last few seconds).
messages	Array	Array of Stream State Messages for this output stream.

Output Stream State example:

```JSON
{
  "id" : 0,
  "uniqueId" : "2f6fe0fb-4374-4970-817f-56a5e509794e",
  "destinationUniqueId" : null,
  "status" : "ERROR",
  "messages" : [ {
    "severity" : "ERROR",
    "message" : "Output Stream 1: Not producing any output."
  } ],
  "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,
  "latencyAddition" : 0,
  "droppedFps" : 0.0
}
```

Elementary Stream

JSON object containing information about elementary stream. All types of elementary streams have the following common fields:

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.
messages	Array	Array of Stream State Messages for this elementary stream.

Depending on the stream type additional fields are available

- VIDEO

- AUDIO
- CAPTIONS
- METADATA

Video Elementary Stream

JSON object containing information about video elementary stream. Except the common elementary stream fields it contains:

Path	Type	Description
`format`	`String`	Format of video. Allowed values are `"RAW"`, `"H262"`, `"H264"`, `"H265"`, `"J2K"`, `"JXS"`.
`width`	`Integer`	Video frame width.
`height`	`Integer`	Video frame height.
`depth`	`Integer`	Video frame samples bit-depth.
`sampling`	`String`	Video frame subsampling. Allowed values are `"4:2:2"`, `"4:2:0"` or `"4:4:4"`.
`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"`.

Example of video elementary stream:

```JSON
{
  "type" : "VIDEO",
  "id" : 0,
  "pid" : 502,
  "state" : "NEW",
  "messages" : [ ],
  "uniqueId" : null,
  "format" : "J2K",
  "width" : 1920,
  "height" : 1080,
  "depth" : 10,
  "sampling" : "4:2:2",
  "scanRate" : "60000/1001",
  "scan" : "INTERLACED",
  "fieldOrder" : "BOTTOM_FIELD_FIRST"
}
```

Audio Elementary Stream

JSON object containing information about audio elementary stream. Except the common elementary stream fields it contains:

Path	Type	Description
`format`	`String`	Format of audio. Allowed values are: `"AC3"`, `"EAC3"`, `"AES3"`, `"AAC_ADTS"`, `"AAC_LATM"`, `"AAC_HE_ADTS"`, `"AAC_HE_LATM"`, `"AAC_HE_V2_ADTS"`, `"AAC_HE_V2_LATM"`, `"MPEG2"`, `"RAW_FLTP"`, `"AES3_DOLBY_E"`, `"RAW_S16"`, `"RAW_S24"`
`channelCount`	`Integer`	Number of audio channels.
`rate`	`Integer`	Audio sampling rate in Hz.
`bitrate`	`Integer`	Bitrate of audio in bits per second.

Example of audio elementary stream:

```JSON
{
  "type" : "AUDIO",
  "id" : 0,
  "pid" : 503,
  "state" : "NEW",
  "messages" : [ {
    "severity" : "ERROR",
    "message" : "This audio stream wasn't detected in source stream."
  } ],
  "format" : "AES3",
  "channelCount" : 2,
  "realChannelCount" : null,
  "rate" : 48000,
  "bitrate" : 235000
}
```

Captions Elementary Stream

JSON object containing information about captions elementary stream. Except the common elementary stream fields it contains:

Path	Type	Description
`format`	`String`	Format of captions. Allowed values are **`"CEA_608_SMPTE_2038"`**, **`"CEA_708_SMPTE_2038"`**, **`"CEA_608_SDI"`** and **`"CEA_708_SDI"`**.

Example of captions elementary stream:

```JSON
{
  "type" : "CAPTIONS",
  "id" : 0,
  "pid" : 504,
  "state" : "NEW",
  "messages" : [ ],
  "format" : "CEA_708_SMPTE_2038"
}
```

Metadata Elementary Stream

JSON object containing information about metadata elementary stream. Except the common elementary stream fields it contains:

Path	Type	Description
`format`	`String`	Format of metadata. Allowed values are **`"SCTE_104"`**, **`"SCTE_35"`**, **`"SDI_ANC"`**, **`"SMPTE_2038"`**, **`"DVB_EIT"`**, **`"AIT"`**, **`"EBU_TELETEXT"`**, **`"UNKNOWN"`**.
`info`	`String`	Metadata description.

Example of metadata elementary stream:

```JSON
{
  "type" : "METADATA",
  "id" : 0,
  "pid" : 505,
  "state" : "NEW",
  "messages" : [ ],
  "format" : "UNKNOWN",
  "info" : "INFO1"
}
```

Stream State Message

JSON object containing stream state message.

Path	Type	Description
`severity`	`String`	Severity of the message. Allowed values are **`"ERROR"`** or **`"WARNING"`**.
`message`	`String`	Message text.

Stream state message example:

JSON


```
{
  "severity" : "ERROR",
  "message" : "Output Stream 1: Not producing any output."
}
```

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.

operation::api-pipelines-get[snippets='http-request,curl-request,response-fields,http-response']

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.

operation::api-pipelines-replace[snippets='request-fields,http-request,curl-request,http-response']

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.

operation::api-pipeline-create[snippets='request-fields,http-request,curl-request,http-response']

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.

operation::api-pipeline-get[snippets='http-request,curl-request,response-fields,http-response']

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.

operation::api-pipeline-update[snippets='request-fields,http-request,curl-request,http-response']

Patch an Existing Pipeline

You may patch an existing pipeline configuration by using this action. It takes pipeline identifier and a JSON Patch (RFC 6902) (https://datatracker.ietf.org/doc/html/rfc6902) request describing the changes that should be applied.

Note that all JSON Patch operations (https://datatracker.ietf.org/doc/html/rfc6902#section-4) are supported except the `test`. Examples of the particular operations can be found in the Appendix A (https://datatracker.ietf.org/doc/html/rfc6902#appendix-A) of the RFC-6902 specification.

One can also reference array objects by their `id` instead of the index by using `id={id}` inside the `path` value e.g. `/outputs/id=1/`. Note that the trailing slash `/` is mandatory in this case.

Be aware that the property `comment` in the following example is not part of the RFC-6902 standard and is there just for explanatory purposes.

operation::api-pipeline-patch[snippets='request-fields,http-request,curl-request,http-response']

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`.

operation::api-pipeline-start[snippets='http-request,curl-request,http-response']

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`.

operation::api-pipeline-stop[snippets='http-request,curl-request,http-response']

Remove an Existing Pipeline

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

operation::api-pipeline-delete[snippets='http-request,curl-request,http-response']

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.

operation::api-pipeline-state-get[snippets='http-request,curl-request,response-fields,http-response']

System

Version info

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

operation::api-system-version[snippets='http-request,curl-request']

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.
`transcoder.buildType`	`String`	CMake build type (e.g., **`Release`**, **`Debug`**, **`RelWithDebInfo`**).
`j2kDec`	`Object`	JPEG 2000 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`	JPEG 2000 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).
`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`	JPEG 2000 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>`**.
`bootMode`	`String`	Boot mode of the system (**`UEFI`** or **`Legacy`**). Not present in case of a Docker deployment.
`ostreeHash`	`String`	OSTree hash of the current system version. Not present in case of a Docker deployment.

operation::api-system-version[snippets='http-response']

Get hostname

Allows you to retrieve the hostname of the system.

operation::api-system-hostname[snippets='http-request,curl-request']

operation::api-system-hostname[snippets='http-response']

Get list of available network interfaces

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

operation::api-system-network-interfaces[snippets='http-request,curl-request,response-fields,http-response']

Get list of available SDI devices

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

operation::api-system-sdi-devices[snippets='http-request,curl-request,response-fields,http-response']

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 input stream source address URL. 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.

operation::api-system-ndi-devices[snippets='http-request,curl-request,response-fields,http-response']

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.

operation::api-system-reinstall[snippets='http-request,curl-request,response-fields,http-response']

Retrieve state of reinstallation process

It returns current state of a reinstallation process.

operation::api-system-reinstall-state[snippets='http-request,curl-request,response-fields,http-response']

Perform restart of main service

operation::api-system-restart-service[snippets='http-request,curl-request,response-fields,http-response']

Perform reboot of the machine

operation::api-system-reboot[snippets='http-request,curl-request,response-fields,http-response']

Perform shutdown of the machine

operation::api-system-shutdown[snippets='http-request,curl-request,response-fields,http-response']

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')"
}
```