Axxon Next HTTP API

1. General agreements . . . 3
2. Infrastructure . . . 3
2.1 Get unique identifier . . . 3
2.2 Get server list . . . 3
2.3 Get list of video sources (cameras) . . . 4
2.4 Get camera live stream . . . 5
2.4.1 Get high and low quality streams . . . 7
2.4.2 Configure tunneling RTSP over HTTP in VLC . . . 7
2.5 Get camera screenshot . . . 8
2.6 Switch between virtual sensor states . . . 8
2.7 Get archive contents . . . 9
2.8 Get info about archive . . . 9
2.9 Get archive stream . . . 10
2.9.1 Get archive stream info . . . 12
2.9.2 Control archive stream . . . 12
2.9.3 Review video footage by frame . . . 12
2.10 Archive search . . . 13
2.10.1 General interface . . . 13
2.10.1.1 Search request . . . 13
2.10.1.2 Search results request . . . 14
2.10.1.3 Search completion . . . 14
2.10.2 Face search API . . . 14
2.10.3 LPR search API . . . 15
2.10.4 Forensic Search MomentQuest (VMDA) API . . . 16
2.10.4.1 Additional conditions . . . 17
2.10.4.2 Types of requests and their parameters . . . 24
2.10.5 ‘Familiar face’-‘stranger’ face search API . . . 26
2.10.6 Define ‘familiar face’-‘stranger’ attribute from image . . . 27
2.11 Get signed links to video streams . . . 27
2.12 Get list of groups and their contents . . . 28
2.13 Export . . . 29
2.14 Get list of detection tools . . . 31
2.15 Get info about triggering of detection tools and alarms . . . 32
2.15.1 Get list of alarms . . . 32
2.15.2 Get list of detection tools events . . . 32
2.16 Telemetry control . . . 34
2.16.1 Get list of telemetry devices for specified video source . . . 34
2.16.2 Acquire telemetry control session . . . 34
2.16.3 Keep session alive . . . 34
2.16.4 Release session . . . 34
2.16.5 Control degrees of freedom . . . 35
2.16.6 Preset control . . . 36
2.16.7 Get information about errors . . . 37
2.17 Working with layouts and videowalls . . . 37
2.17.1 Sequence of actions . . . 37
2.17.2 Getting the list of layouts . . . 38
2.17.3 Switching the layout on the screen . . . 38
2.17.4 Getting the list of cameras displayed on the layout . . . 39
2.17.5 Adding and removing cameras . . . 39
2.17.6 Getting the list of displays . . . 39
2.17.7 Selecting active display . . . 40
2.17.8 Switching camera to archive mode . . . 40
2.18 Using macros . . . 40
2.19 Get data from system log . . . 41
2.20 Get statistics . . . 42
2.21 Get info about Server usage . . . 43
2.22 Get info about Server version . . . 43

General agreements

Rus

HTTP server NGP responds to method calls in the form of JSON.

> **Note**
> Configuring the web server.
>
> By default the Web-server port is **80**, prefix is **/** (empty).

> **Note**
> GET is not in use when request opening in the browser.

Authorization is needed for requests. Supported authorization type is basic.

Authorization is required in each HTTP request as follows:

```
http://[username]:[password]@[IP-address]:[port]/[prefix]
```

The number of active requests and requests in queue is limited.

The **503** error (Search query rejected. Too many requests) returns when there are too many requests.

Infrastructure

Get unique identifier

Rus

(UUID) is generated for every GET request to http://IP-Address:port/prefix/uuid.

Unique identifier is used to get in last frame info from archive video or to control archived stream.

Response sample:

```
{
  "uuid": "2736652d-af5f-4107-a772-a9d78dfaa27e"
}
```

Get server list

Rus

On page:

- List of domain servers
- Server info

List of domain servers

GET http://IP-Address:port/prefix/hosts/ - gets the list of all domain hosts.

Sample response:
["SERVER1", "SERVER2"]

Server info

GET http://IP-Address:port/prefix/hosts/HOSTNAME - gets host info.

Sample response:

```
{
   "hostname" : "SERVER2",
   "domainInfo" :
   {
```

```
      "domainName" : "DomainName",
    "domainFriendlyName" : "Custom domain name, if available"
},
"platformInfo" :
   {
"machine" : "ARM9",
"os" : "Linux"
   },
   "licenseStatus" : "Expired",
"timeZone" : "+180" // GMT+3
}
```


Get list of video sources (cameras)

Rus

On page:

- Get all available sources
- Get all available original sources of server
- Get source info
- Get all sources info

Get all available sources

GET http://IP-Address:port/prefix/video-origins/ - gets all available original sources (cameras). The requested identifiers will have the format as follows "HOSTNAME/ObjectType.Id/Endpoint.Name". Friendly name and other related meta data will be received.

> **Note**
> UTF-8 coding is to be in use in a browser for proper displaying of video camera names

Sample response:


```
{
   "SERVER1/DeviceIpint.3/SourceEndpoint.video:0:0" :
   {
      "origin" : "SERVER1/DeviceIpint.3/SourceEndpoint.video:0:0",

      "state" : "connected",
      "friendlyNameLong" : "Camera 3",
      "friendlyNameShort" : "3"
},
   "SERVER2/DeviceIpint.5/SourceEndpoint.video:0:0" :
   {
      "origin" : "SERVER2/DeviceIpint.5/SourceEndpoint.video:0:0",

      "state" : "disconnected",
      "friendlyNameLong" : "Camera 5",
      "friendlyNameShort" : "5"
   }
}
```


Get all available original sources of server

GET http://IP-Address:port/prefix/video-origins/HOSTNAME/ gets all available original sources (cameras) of the specified server.

Get source info

GET http://IP-Address:port/prefix/video-origins/VIDEOSOURCEID - gets source information. VIDEOSOURCEID – identifier of endpoint source consisting of 3 components (HOSTNAME/ObjectType.Id/Endpoint.Name).

Sample request:

GET http://IP-Address:port/prefix/video-origins/SERVER1

Sample response:


```
{
   "SERVER1/DeviceIpint.3/SourceEndpoint.video:0:0" :
   {
      "origin" : "SERVER1/DeviceIpint.3/SourceEndpoint.video:0:0",
```

```
      "state" : "none",
      "friendlyNameLong" : "Camera 3",
      "friendlyNameShort" : "3"
}
}
```


Get all sources info

GET http://IP-Address:port/prefix/video-sources/* - gets the list of all available sources not limited by original ones.

Sample request:

GET http://IP-Address:port/prefix/video-sources/SERVER2

Sample response:


```
{
   "SERVER2/DeviceIpint.5/SourceEndpoint.video:0:0" :
   {
      "origin" : "SERVER2/DeviceIpint.5/SourceEndpoint.video:0:0",

      "state" : "none",
      "friendlyNameLong" : " Camera 5",
      "friendlyNameShort" : "5"
   },
   "SERVER2/VideoDecoder.0/VideoSource" :
   {
      "origin" : "SERVER2/DeviceIpint.5/SourceEndpoint.video:0:0",

      "state" : "connected",
      "friendlyNameLong" : "SERVER2/Videodecoder 0",
      "friendlyNameShort" : "Videodecoder 0"
}
}
```


The "state" field display the source state. Available values:

– "connected" – video source is connected;

– "disconnected" – video source is disconnected;

– "signal_restored" – signal from video source is restored;

– "signal_lost" – signal from video source is lost.

Get camera live stream

Rus

On page:

- General information
- HLS video
- RTSP video
- HTTP video
- Tunneling RTSP over HTTP
- H.264 video

General information

GET http://IP-Address:port/prefix/live/media/VIDEOSOURCEID?parameters.

where **VIDEOSOURCEID** - three-component source endpoint ID (see Get list of video sources (cameras)). For instance, "SERVER1/DeviceIpint.3/SourceEndpoint.video:0:0".

Parameters:

- **format** - "rtsp" and "hls".
  Video can be received in the original format (without recompression) via RTSP and HLS protocols. HLS protocol supports only H.264 format. In all other cases the server recompresses it to MJPEG format.

  ⚠ **Important!**
  If video is requested in the format that differs from the original one, then recompression will be performed, therefore, Server load will increase.

- **w** – frame width
- **h** – frame height.

> **Note**
> If **h** and **w** values are more than size of original video, the video will be received with original size.
>
> Zooming out of width and height is available only discretely - in 2, 4, 8 times, etc. If specified sizes are not corresponding to 1/2, 1/4 etc. of original video size, the video will be received with size divisible by the original video size close to specified values.

Sample request:

GET http://IP-Address:port/prefix/live/media/HOSTNAME/DeviceIpint.23/SourceEndpoint.video:0:0?w=640&h=480

HLS video

HLS protocol video can be received in the original format only. The following parameters are in use when receiving HLS protocol video:

keep_alive – time in seconds in which the stream is to be kept alive.

hls_time - the segment length in seconds.

hls_list_size - the maximum number of playlist entries. If set to **0** the list file will contain all the segments.

hls_wrap - the number after which the segment filename number wraps. If set to **0** the number will be never wrapped.

.

Sample request:

GET http://IP-Address:port/prefix/live/media/HOSTNAME/DeviceIpint.23/SourceEndpoint.video:0:0?format=hls&keep_alive=60

Sample response:

{

"keep_alive_seconds": 60,
"keep_alive_url": "/live/media/hls/keep?stream_id=7e9d8c93-80e2-4521-9a54-cb854fe3cd2d",
"stop_url": "/live/media/hls/stop?stream_id=7e9d8c93-80e2-4521-9a54-cb854fe3cd2d",
"stream_url": "/hls/7e9d8c93-80e2-4521-9a54-cb854fe3cd2d/playout.m3u8"

}

where **keep_alive_url** - the url to keep the stream alive;

stop_url - the url to stop the stream;

stream_url - the url to access the list of segments.

> **Important!**
> HLS protocol video becomes available in several seconds after getting the response.

RTSP video

RTSP protocol video is sent in the original format only.

GET rtsp://login:passowrd@IP-Address:554/hosts/HOSTNAME/DeviceIpint.23/SourceEndpoint.video:0:0

HTTP video

GET ffplay.exe -v debug "http://login:password@IP-адрес:8001/asip-api/live/media/HOSTNAME/DeviceIpint.23/SourceEndpoint.video:0:0?w=1600&h=0"

> **Important!**
> HTTP sends video in mjpeg only, **w** and **h** parameters are mandatory.

Tunneling RTSP over HTTP

see Configure tunneling RTSP over HTTP in VLC

Video is sent over the tunnel in the original format.

Samples:

GET ffplay -rtsp_transport http "rtsp://login:password@IP-Address:8554/rtspproxy/hosts/HOSTNAME/DeviceIpint.23/SourceEndpoint.video:0:0"

GET for VLC: rtsp://login:password@IP-Address:8554/rtspproxy/hosts/HOSTNAME/DeviceIpint.23/SourceEndpoint.video:0:0

H.264 video

To get live video in the original H.264 format use RTSP or RTSP tunnel over HTTP.

Get high and low quality streams

Rus

Get list of video sources (cameras)

Get camera live stream

General case:

- GET http://IP-Address:port/prefix/live/media/SERVER1/DeviceIpint.3/SourceEndpoint.video:0:0?w=1600&h=0 – high quality stream
- GET http://IP-Address:port/prefix/live/media/SERVER1/DeviceIpint.3/SourceEndpoint.video:0:1?w=1600&h=0 –low quality stream

RTSP:

- GET rtsp://login:password@IP-Address:554/hosts/SERVER1/DeviceIpint.3/SourceEndpoint.video:0:0 - high quality stream
- GET rtsp://login:password@IP-Address:554/hosts/SERVER1/DeviceIpint.3/SourceEndpoint.video:0:1 - low quality stream

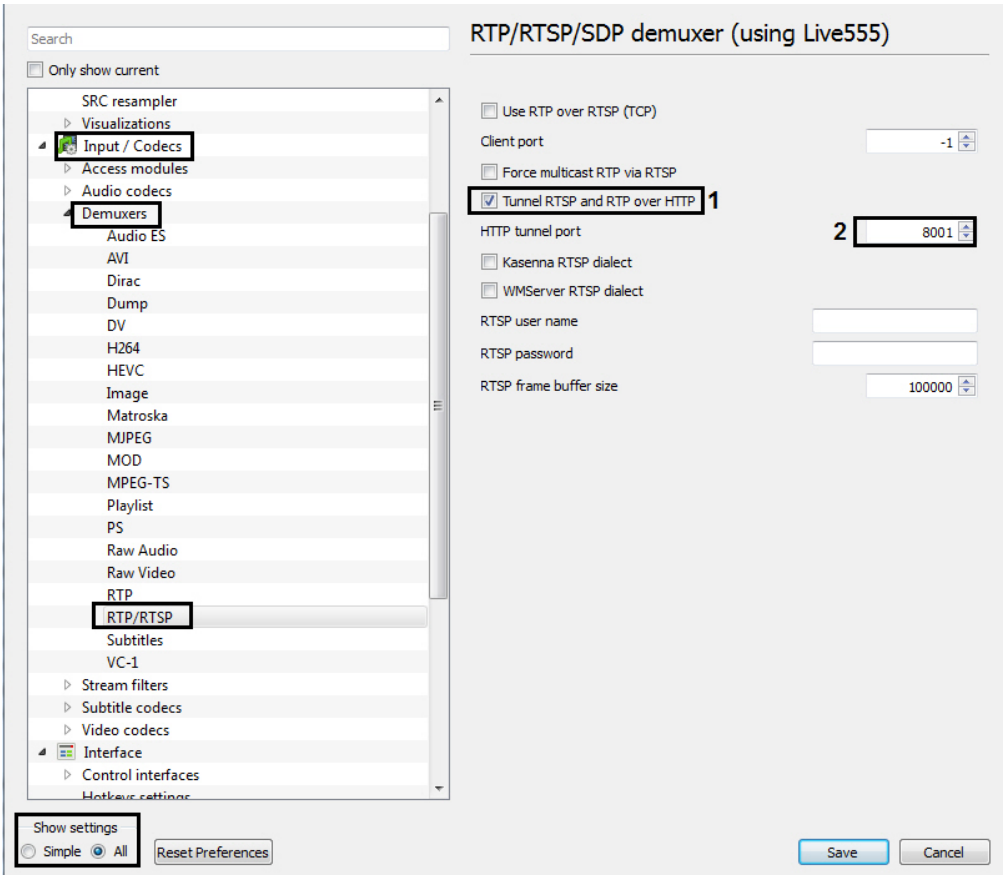
Tunneling RTSP over HTTP:

- GET rtsp://login:password@IP-Address:80/rtspproxy/hosts/SERVER1/DeviceIpint.3/SourceEndpoint.video:0:0 - high quality stream
- GET rtsp://login:password@IP-Address:80/rtspproxy/hosts/SERVER1/DeviceIpint.3/SourceEndpoint.video:0:1 - low quality stream

Configure tunneling RTSP over HTTP in VLC

Rus

To configure tunneling in VLC set the **Tunnel RTSP and RTP over HTTP** checkbox (1) checked and the HTTP tunnel port – **8001 (2)**.



Get camera screenshot

Rus

GET http://IP-Address:port/prefix/live/media/snapshot/VIDEOSOURCEID?parameters.

where **VIDEOSOURCEID** - three-component source endpoint ID (see Get list of video sources (cameras)).

Parameters:

w – frame width**.**

h – frame height.

Sample request:

To get a screenshot in the original resolution: GET http://IP-Address:port/prefix/live/media/snapshot/HOSTNAME/DeviceIpint.23/SourceEndpoint.video:0:0

To get a screenshot in 640*480 resolution: GET http://IP-Address:port/prefix/live/media/snapshot/HOSTNAME/DeviceIpint.23/SourceEndpoint.video:0:0?w=640&h=480

Switch between virtual sensor states

Rus

POST http:/IP-address:port/device/di/0

with content:

{"state": "closed"}

where

- **port** - http listener port.
- 0/1/2/3 - sensor id.
- **state** - **opened** or **closed**.

Example:

```
http://127.0.0.1:8080/device/di/0
{"state": "closed"}
```

Get archive contents

Rus

Get list of archives recording is performed to:

GET http://P-Address:port/prefix/archive/list/SERVER1/VIDEOSOURCEID

where **VIDEOSOURCEID** - three-component source endpoint ID (see Get list of video sources (cameras)). For instance, "SERVER1/DeviceIpint.3/SourceEndpoint.video:0:0".

Get default archive contents:

GET http://IP-Address:port/prefix/archive/contents/intervals/VIDEOSOURCEID/ENDTIME/BEGINTIME?limit=COUNT&scale=SIZE – get archive contents starting at BEGINTIME and ending at ENDTIME.

If BEGINTIME is not specified, infinite future is considered. If ENDTIME is not specified too, infinite past is considered. Words "past" and "future" can be used to set infinite past and infinite future as well.

Optional *limit* parameter is used to specify the limit of frames. The default value is **100**.

Optional *scale* parameter is used to specify the minimum time interval between the frames when they are treated as separate ones (not merged). The default value is **0**.

Interval sequence corresponds to the ratio between specified BEGINTIME and ENDTIME (in ascending order if BEGINTIME<ENDTIME, and in descending order if ENDTIME<BEGINTIME). Start and end points of interval are returned in its common order, i.e. the interval start time is less than the interval end time or equal to it.

The **intervals** property contains array of frame intervals returned as json object.

The returned json response contains the Boolean **more** property which is used to specify if complete time interval is selected (false) or some frames were not returned because their timestamps maxed out (true).

Get contents of specific archive:

GET http://IP-Address:port/prefix/archive/contents/intervals/SERVER1//VIDEOSOURCEID/future/past?archive=Archive_Name

Sample request:

GET http://IP-Address:port/prefix/archive/contents/intervals/SERVER1/DeviceIpint.2/SourceEndpoint.video:0:0/20101230T103904.000/20101230T103959.000?limit=3

Sample response:

```
{
   "intervals" :
   [
      { begin: "20101230T103950.000", end: "20101230T103955.230" },
      { begin: "20101230T103923.110", end: "20101230T103941.870" }
   ],
   "more" : true
}
```

Note
Time returns in the UTC format

Get info about archive

Rus

Archive depth

GET http://IP-Address:port/prefix/archive/statistics/depth/VIDEOSOURCEID/ENDTIME/BEGINTIME?threshold=7 - getting information about the archive depth starting at BEGINTIME and ending at ENDTIME.

VIDEOSOURCEID – identifier of endpoint source consisting of 3 components (HOSTNAME/ObjectType.Id/Endpoint.Name).

threshold - optional parameter. It is used to set treshold value (in days). When this treshold is crossed, intervals are not merged anymore. Default value is 1 day.

> **Note**
> The ENDTIME and BEGINTIME syntax is described in Get MM archive contents section.

Sample request:

GET http://localhost:8000/archive/statistics/depth/SERVER1/DeviceIpint.23/SourceEndpoint.video:0:0?threshold=2

Sample response:

```
{
  "start": "20160823T141333.778000"
  ,"end": "20160824T065142"
}
```

where 20160823T141333.778000 - 20160824T065142 is a time interval for which arhive recordings are available.

Recording capacity to specific camera archive

GET http://IP-Address:port/prefix/archive/statistics/capacity/VIDEOSOURCEID/ENDTIME/BEGINTIME - getting information about the recording capacity to specific camera archive starting at BEGINTIME and ending at ENDTIME.

> **Note**
> The ENDTIME and BEGINTIME syntax is described in Get MM archive contents section.

Sample request:

GET http://IP-Address:port/prefix/archive/statistics/capacity/SERVER1/DeviceIpint.23/SourceEndpoint.video:0:0/past/future

Sample response:

```
{
  "size": 520093696
  ,"duration": 32345
}
```

where **size** - archive size (in bytes) over the specified period;

duration - archive duration (in seconds) over the specified period.

Get archive stream

Rus

On page:

- Get archive stream from default archive
- Get archive stream from specific archive
- RTSP archive video
- HTTP archive video
- Tunneling RTSP over HTTP
- H.264 archive video

Get archive stream from default archive

GET http://IP-Address:port/prefix/archive/media/VIDEOSOURCEID/STARTTIME?parameters,

where

- **VIDEOSOURCEID** - three-component source endpoint ID (see Get list of video sources (cameras)). For instance, "SERVER1/DeviceIpint.3/SourceEndpoint.video:0:0".
- **STARTTIME** - time in ISO format.

> **Important!**
> Set the timezone to UTC+0

Parameters:

speed – playback speed, values can be negative.

format - parameter values are 'mjpeg', 'rtsp or 'hls'. If the format is not specified, 'rtsp' is selected or it is not recognized, then the native format is selected by server to prevent additional encoding. If the native format is not supported by client, server selects WebM.

If neither of parameters is specified, the speed is equal to 0, JPEG format is selected and the request is handled as a request to review video footage by frames.

id – unique identifier of archive stream (optional). It is used to get stream info or control the stream.

w – frame width.

h – frame height.

Sample request:

GET http://IP-Address:port/prefix/archive/media/HOSTNAME/DeviceIpint.23/SourceEndpoint.video:0:0/20110608T060141.375?format=rtsp&speed=1&w=640&h=480

Assign ID to the stream to receive information about this stream.

http://IP-Address:port/prefix/archive/media/VIDEOSOURCEID/STARTTIME/20140723T120000.000?format=rtsp&speed=1&w=640&h=480&id=f03c6ccf-b181-4844-b09c-9a19e6920fd3

It is possible to use other values consisting of latin letters and digits. It is recommended to use the uuid function (see Get unique identifier).

Important!
HLS archive video becomes available in 30 seconds after getting the response

Sample response:


```
{"http":
{"description":"RTP/RTSP/HTTP/TCP","path":"archive/hosts/HOSTNAME/DeviceIpint.1/SourceEndpoint.video:0:0/20161206T060141.375000?speed=1&id=729955cd-7787-4d6c-87eb-cd6dd6d4a940","port":"8554"}
,"rtsp":
{"description":"RTP/UDP or RTP/RTSP/TCP","path":"archive/hosts/HOSTNAME/DeviceIpint.1/SourceEndpoint.video:0:0/20161206T060141.375000?speed=1&id=729955cd-7787-4d6c-87eb-cd6dd6d4a940","port":"554"}
}
```


Get archive stream from specific archive

GET http://IP-Address:port/prefix/archive/media/VIDEOSOURCEID/STARTTIME?parameters&arhive=hosts/SERVER1/MultimediaStorage.Archive_Name/MultimediaStorage

RTSP archive video

GET rtsp://login:password@IP-Address:554/archive/hosts/SERVER1/DeviceIpint.0/SourceEndpoint.video:0:0/20160907T050548.723000

HTTP archive video

GET ffplay.exe -v debug "http://login:password@IP-Address:8001/asip-api/archive/media/SERVER1/DeviceIpint.4/SourceEndpoint.video:0:0/20170112T113526?w=1600&h=0&speed=1"

Tunneling RTSP over HTTP

see Configure tunneling RTSP over HTTP in VLC.

GET ffplay -rtsp_transport http "rtsp://login:password@IP-Address:8554/rtspproxy/archive/hosts/SERVER1/DeviceIpint.4/SourceEndpoint.video:0:0/20170115T113526"

For VLC: GET rtsp://login:password@IP-Address:8554/rtspproxy/archive/hosts/SERVER1/DeviceIpint.4/SourceEndpoint.video:0:0/20170115T113526

H.264 archive video

To get H.264 archive video use RTSP protocol:

GET rtsp://login:password@IP-Address:554/archive/hosts/SERVER1/DeviceIpint.4/SourceEndpoint.video:0:0/20170112T113526

or tunneling RTSP over HTTP:

GET rtsp://login:password@IP-Address:8001/rtspproxy/archive/hosts/SERVER1/DeviceIpint.4/SourceEndpoint.video:0:0/20170115T113526

Get archive stream info

Rus

GET http://IP-Address:port/prefix/archive/media/rendered-info/UUID -gets info of the frame last displayed,

where UUID is a unique identifier of the requested archive stream.

The following frame info is available:

timestamp – frame time token.

Sample request:

GET http://IP-адрес:port/prefix/archive/media/rendered-info/22996cea31-91c4-9a46-9269-48b998fd2f29

Sample response:

```
{
  "timestamp": "20110408T103627.048"
}
```

Control archive stream

Rus

GET http://IP-Address:port/prefix/archive/media/stop/UUID - stops archive stream that has the specified UUID.

When completed, the last frame info is received.

> **Note**
> The archive stream stop command is not applied to video in rtsp format.
>
> The stop command breaks connection with NGP service for video in hls format.

Review video footage by frame

Rus

On page:

- Get frame by timestamp
- Get frame registration time

Get frame by timestamp

GET http://IP-Address:port/prefix/archive/media/VIDEOSOURCEID/STARTTIME - gets frame by its STARTTIME. Frame is returned in JPEG format.

Get frame registration time

GET http://IP-Address:port/prefix/archive/contents/frames/VIDEOSOURCEID/ENDTIME/BEGINTIME?limit=COUNT – gets the time of frame registration in MM archive. Parameter semantics is described in section Get archive contents. The default value of *limit* parameter is 250. This parameter is optional for server and it can return fewer search results.

The **frames** property will contain an array of frame timestamps returned as json object.

The returned json response will contain the **more** Boolean property, which is used to specify if complete time interval is selected (false) or some frames were not returned because their timestamps maxed out.

Sample request:

GET http://IP-Address:port/prefix/archive/contents/frames/SERVER1/DeviceIpint.2/SourceEndpoint.video:0:0/20101230T103943.000/20101230T103952.000?limit=3

Sample response:

```
{
"frames" :
[ "20101230T103951.800", "20101230T103951.760", "20101230T103951.720" ],
"more" : false
}
```

Archive search

General interface

Search request

Rus

Search by one source

Method: POST http://IP-Address:port/prefix/search/(auto|face|vmda|stranger)/DETECTORID/BEGINTIME/ENDTIME

Where

- **auto**|**face**|**vmda**|**stranger** – search type.
- **DETECTORID** – endpoint detection tool ternary ID (HOSTNAME/AVDetector.ID/EventSupplier for auto and face search, HOSTNAME/AVDetector.ID/SourceEndpoint.vmda for vmda).

> **Note**
> The list of detectors is available in the \ProgramData\AxxonSoft\AxxonNext\Config.local\config_repo system configuration.

- **ENDTIME**, **BEGINTIME** – time in ISO format.

A request for search on a single computer is also supported for auto and face search, the request structure is as follows:

http://localhost/prefix/search/(auto|face)/HOSTID/BEGINTIME/ENDTIME,

where **HOSTID** is a computer name.

Search by multiple sources

Method: POST http://IP-Address:port/prefix/search/(auto|face|vmda|stranger)/BEGINTIME/ENDTIME

This search type always accepts JSON in the POST body that is to include at least one section of the form:

```
"sources": [
 "hosts/Server1/AVDetector.1/EventSupplier"
]
```

When the search request is performed, JSON is to include image in base64 format.

```
{
 "sources": [
    "hosts/Server1/AVDetector.1/EventSupplier",
    "hosts/Server1/AVDetector.2/EventSupplier"
  ],
   "image" : "base64 encoded image"
}
```

Result

The request will return either error or response like:

```
HTTP/1.1 202 Accepted
Connection: Close
Location: /search/(auto|face|vmda|stranger)/GUID
Cache-Control: no-cache
```

The **Location** field contains an identifier for future access to search results. Example: /search/vmda/3dc15b75-6463-4eb1-ab2d-0eb0a8f54bd3

Receiving the **Accepted** code does not guarantee successful execution of the search. This code only shows that the command has been taken to process.

Error codes:

400 – incorrect request.

500 – internal Server error.

Search results request

Rus

Method: GET http://IP-Address:port/search/(auto|face|vmda|stranger)/GUID/result?offset=0&limit=10

The **/search/(auto|face|vmda)/GUID** part is a result of the POST command (see Search request).

limit (uint32_t::max() by default) is a maximum number of events returned by the request.

offset (0 by default) is a resulting sample offset.

Returned result depends on the search type. The request can return two successful statuses:

206 – search is not over. Repeat search results requests until status code 200 is returned. Set delays between repeated requests in order to reduce computational burden.

200 – search is over.

Error codes:

400 – incorrect request.

404 – the **offset** value is greater than current quantity of results or requested search ID (**GUID**) not found.

Search completion

Rus

Method: DELETE http://IP-адрес:порт/search/(auto|face|vmda|stranger)/GUID

The **/search/(auto|face|vmda)/GUID** part is a result of the POST command (see Search request).

The command terminates the search operation and deallocates resources. Search results are not available after it is executed.

Error codes:

400 – incorrect request.

Face search API

Rus

The POST request (see Search request) used for search start must contain binary data of searched face in jpeg format.

The **accuracy** parameter is a recognition rate from range [0, 1] (1 – complete match). This parameter is specified additionally. Otherwise, the default value 0.9 will be used.

The search result is the following JSON response:

```
{
  "events" : [
    {
      "accuracy" : 0.90591877698898315,
      "origin" : "hosts/SERVER1/DeviceIpint.2/SourceEndpoint.video:0:0", "position" : {
      "bottom" : 0.10694444444444445, "left" : 0.69687500000000002, "right" :
0.74687500000000007, "top" : 0.018055555555555554
    },
    "timestamp" : "20160914T085307.499000"
    },
    {
      "accuracy" : 0.90591877698898315,
      "origin" : "hosts/SERVER1/DeviceIpint.2/SourceEndpoint.video:0:0", "position" : {
      "bottom" : 0.10694444444444445, "left" : 0.69687500000000002, "right" :
0.74687500000000007, "top" : 0.018055555555555554
    },
      "timestamp" : "20160914T085830.392000"
    }
  ]
}
```


Parameters:

- **origin** is a camera channel to take analyzed video stream from.
- **timestamp** is a time stamp of a video frame with a face detected by the detection tool.
- **accuracy** is recognition accuracy ranged [0,1], with 1 corresponding to full match.
- **position** sets coordinates of a frame border enclosing face on a video frame.

LPR search API

Rus

The POST request (see Search request) used for search start must contain the following JSON:


```
{
  "plate": "mask"
}
```


The **plate** parameter sets a search mask. The mask format corresponds to the one used in GUI (see LPR search).

The search result is the following JSON response:

```
{
 "events" : [
    {
      "origin" : "hosts/V-SHMELEV/DeviceIpint.4/SourceEndpoint.video:0:0",
      "plates" : [ "T470PM197", "T470PM19" ],
      "timestamp" : "20160921T084300.235000"
    },
    {
      "origin" : "hosts/V-SHMELEV/DeviceIpint.4/SourceEndpoint.video:0:0",
      "plates" : [ "T715EP199", "T715EP14" ],
      "timestamp" : "20160921T084301.795000"
    },
    {
      "origin" : "hosts/V-SHMELEV/DeviceIpint.4/SourceEndpoint.video:0:0",
      "plates" : [ "Y497XY197" ],
      "timestamp" : "20160921T084336.915000"
    }
  ]
}
```


Parameters:

- **origin** – camera channel to take analyzed video stream from.
- **timestamp** – time stamp of a frame with a license plate detected by the detection tool.
- **plates** – list of supposed hypotheses.

Forensic Search MomentQuest (VMDA) API

Rus

The POST request (see Search request) for search start must contain JSON of one of the following types:

1. Constructor describing parameters for metadata database request.

   There are three logical parts of the search request:
   a. Request type (`queryType`, see `Types of requests and their parameters`)
   b. Parameters specific for the specified type of request (`figures`, `queryProperties`, see `Additional conditions`)
   c. Additional filter conditions (`objectProperties`, `conditions`, see `Additional conditions`)
2. Direct request in metadata database language.


```
{
 "query": "figure
fZone=polygon(0.4647676,0.3973333,0.7946027,0.5493333,0.8650675,0.7946666,0.4647676,0.7946666); figure fDir=(ellipses(-10000, -10000, 10000, 10000) -       ellipses(-0, -0, 0, 0));set r = group[obj=vmda_object] { res = or(fZone((obj.left + obj.right) / 2, obj.bottom)) }; result = r.res;"
}
```


Important!
If input JSON has both the constructor and the direct request sections, the direct request has higher priority.

Note
To perform search in offline analytics data, use the following request:

```
POST
/search/vmda/SERVER-NAME/OfflineAnalytics.c95ad5a581094845995ee28a7f097797/SourceEndpoint.vmda:AVDetector:1/past/future
```

This request will be performed even if AxxonNext archive is removed, but VMDA metadata is saved.

Object ID is to be specified without the **hosts/** prefix.

Valid request: /search/vmda/SERVER-NAME/OfflineAnalytics...

Invalid request: /search/vmda/hosts/SERVER-NAME/OfflineAnalytics...

The search result is the following JSON response:

```
{
  "intervals" : [
    {
      "endTime" : "20160919T064640.430000",
      "startTime" : "20160919T064636.390000"
    },
    {
      "endTime" : "20160919T073204.113000",
      "startTime" : "20160919T073201.513000"
    }
}
```

where **Intervals** is a set of time intervals for which the search condition is fulfilled.

Additional conditions

Rus

On page:

- Object type (objectProperties/category)
- Object size (objectProperties/size)
- Object color (objectProperties/color)
- Velocity (conditions/velocity)
- Directions (conditions/directions)
- Duration (conditions/duration)
- Object number (condtions/count)

Additional conditions match all kinds of requests. Conditions are always joined with logical *"AND".* For instance, the "*object of height not more than a quarter of the frame that is in the camera field of view for 5 seconds*" request looks like this:

```
{
  "queryType": "zone",
  "figures": [
     {
        "shape": [
           [0.3, 0.3],
           [0.7, 0.3],
           [0.7, 0.7],
           [0.3, 0.7]
        ]
     }
  ],
  "objectProperties": {
     "size": {
        "height": [0, 0.25]
     }
  },
  "conditions": {
     "duration": 5
  }
}
```

Object type (`objectProperties/category`)

An object can be abandoned or moving (face, human, group, vehicle). In the request abandoned type cannot be mixed with other object types (otherwise the abandoned requirement will be ignored).

Search for objects abandoned in any point of the frame looks like this:

```
{
  "queryType": "zone",
  "figures": [
     {
        "shape": [
           [0, 0],
           [1, 0],
           [1, 1],
           [0, 1]
        ]
     }
  ],
  "objectProperties": {
     "category": ["abandoned"],
  }
}
```

The search for a human or a small group of people crossing the line looks like this:

```
{
  "queryType": "line",
  "figures": [
     {
        "shape": [
           [0.5, 0.8],
           [0.5, 0.2]
        ]
     }
  ]
  "objectProperties": {
     "category": ["human", "group"],
  }
}
```

Object size (`objectProperties/size`)

It sets minimum and maximum width and height of an object.

For instance, to find objects that are not bigger than a quarter of the frame in height one can use this request:

:

```
{
   "queryType": "zone",
   "figures": [
      {
         "shape": [
            [0.3, 0.3],
            [0.7, 0.3],
            [0.7, 0.7],
            [0.3, 0.7]
         ]
      }
   ],
   "objectProperties": {
      "size": {
         "width": [0, 1],
         "height": [0, 0.25]
      }
   }
}
```

As both dimensions are not necessary to be set, this request will be similar to the previous one:

```
{
  "queryType": "zone",
  "figures": [
     {
        "shape": [
           [0.3, 0.3],
           [0.7, 0.3],
           [0.7, 0.7],
           [0.3, 0.7]
        ]
     }
  ],
  "objectProperties": {
     "size": {
        "height": [0, 0.25]
     }
  }
}
```

Object color (`objectProperties/color`)

It sets minimum and maximum coordinates of the object color in HSV space. hue is measured in degrees (from 0 to 360), saturation and brightness – in fractions from 0 to 1.

The request to get bright green objects in the zone looks like this:

```
{
  "queryType": "zone",
  "figures": [
     {
        "shape": [
           [0.3, 0.3],
           [0.7, 0.3],
           [0.7, 0.7],
           [0.3, 0.7]
        ]
     }
  ],
  "objectProperties": {
     "color": {
        "hue": [75, 135],
        "saturation": [0.5, 1],
        "brightness": [0.5, 1]
     }
  }
}
```

In HSV space dark almost black colors can have any hue and saturation. So to search black objects the request should look like this:

```
{
   "queryType": "zone",
   "figures": [
      {
         "shape": [
            [0.3, 0.3],
            [0.7, 0.3],
            [0.7, 0.7],
            [0.3, 0.7]
         ]
      }
   ],
   "objectProperties": {
      "color": {
         "hue": [0, 360],
         "saturation": [0, 1],
         "brightness": [0, 0.2]
      }
   }
}
```

Here is the same request for white objects:

```
 {
   "queryType": "zone",
   "figures": [
      {
         "shape": [
            [0.3, 0.3],
            [0.7, 0.3],
            [0.7, 0.7],
            [0.3, 0.7]
         ]
      }
   ],
   "objectProperties": {
      "color": {
         "hue": [0, 360],
         "saturation": [0, 0.1],
         "brightness": [0.8, 1]
      }
   }
}
```

Velocity (`conditions/velocity`)

It sets minimum and maximum velocity of the object.

It is measured in frame rates per second – i.e. the velocity of the object moving from the left edge of the frame to the right one over 1 second is 1.

.

```
{
  "queryType": "zone",
  "figures": [
     {
        "shape": [
           [0.3, 0.3],
           [0.7, 0.3],
           [0.7, 0.7],
           [0.3, 0.7]
        ]
     }
  ],
  "conditions": {
     "velocity": [0.25, 1]
  }
}
```

Directions (`conditions/directions`)

It sets the direction for the object as an array of angles. Angles are measured in radians and are counted from the axis directed to the right clockwise.

So the request to get objects moving to the right ±45° looks like this:

:

```
{
  "queryType": "zone",
  "figures": [
     {
        "shape": [
           [0.3, 0.3],
           [0.7, 0.3],
           [0.7, 0.7],
           [0.3, 0.7]
        ]
     }
  ],
  "conditions": {
     "directions": [
        [315, 45]
     ]
  }
}
```

Pay attention that 45° -- 315° angle covers all directions except “to the right”.

If one needs to find objects moving mainly horizontally, then two angles are to be set:

```
{
  "queryType": "zone",
  "figures": [
     {
        "shape": [
           [0.3, 0.3],
           [0.7, 0.3],
           [0.7, 0.7],
           [0.3, 0.7]
        ]
     }
  ],
  "conditions": {
     "directions": [
        [315, 45],
        [135, 225]
     ]
  }
}
```

Duration (`conditions/duration`)

It sets time (in seconds) during which the object is to continuously meet all conditions.

Using this condition one can make the *"long presence in the zone"* request:

:

```
{
  "queryType": "zone",
  "figures": [
     {
        "shape": [
           [0.3, 0.3],
           [0.7, 0.3],
           [0.7, 0.7],
           [0.3, 0.7]
        ]
     }
  ],
  "conditions": {
     "duration": 5
  }
}
```

Object number (`condtions/count`)

It sets the minimum required number of objects that simultaneously meet other request conditions.

It is usually used for search of a big number of objects in the zone. For instance:

```
{
  "queryType": "zone",
  "figures": [
     {
        "shape": [
           [0.3, 0.3],
           [0.7, 0.3],
           [0.7, 0.7],
           [0.3, 0.7]
        ]
     }
  ],
  "conditions": {
     "count": 3
  }
}
```

Types of requests and their parameters

Rus

On page:

- Object in the zone (queryType="zone")
- Object transition from one zone to another (queryType="transition")
- Line crossing (queryType="line")

Object in the zone (`queryType="zone"`)

figures/shape is a required parameter. It sets the zone the object to be within as the list of polygon apexes. Coordinates are set in fractions of frame width/height (values from 0 to 1). It allows not to be tied to specific camera resolution.

The simplest request looks like this:

```
{
  "queryType": "zone",
  "figures": [
     {
        "shape": [
           [0.3, 0.3],
           [0.7, 0.3],
           [0.7, 0.7],
           [0.3, 0.7]
        ]
     }
  ]
}
```

Here the zone describes the rectangular in the centre of camera field of view.

queryProperties/action is an optional parameter of the request:

- If this parameter is not set, then objects in the zone are searched.
- `queryProperties/action="enter"` - objects entering the zone are searched.
- `queryProperties/action="exit"` - objects exiting the zone are searched.

Here is an example of request for objects entering the zone:

```
{
   "queryType": "zone",
   "figures": [
      {
         "shape": [
            [0.3, 0.3],
            [0.7, 0.3],
            [0.7, 0.7],
            [0.3, 0.7]
         ]
      }
   ],
   "queryProperties": {
      "action": "enter"
   }
}
```

Object transition from one zone to another (`queryType="transition"`)

There is one required parameter — **figures**. It has to contain two zones describing start and end zones.

There are no optional parameters.

Search for objects that moved from the left part of the frame to the right one:

```
 {
   "queryType": "transition",
   "figures": [
      {
         "shape": [
            [0, 0],
            [0.45, 0],
            [0.45, 1],
            [0, 1]
         ]
      },
      {
         "shape": [
            [0.55, 0],
            [1, 0],
            [1, 1],
            [0.55, 1]
         ]
      }
   ]
}
```

Line crossing (`queryType="line"`)

Required parameter **figures** defines a segment crossing of which triggers the condition. The segment is to be set by two points.

```
{
  "queryType": "line",
  "figures": [
     {
        "shape": [
           [0.5, 0.8],
           [0.5, 0.2]
        ]
     }
  ]
}
```

Optional parameter **queryProperties/direction** sets direction of line crossing.

- If this parameter is not set, then objects crossing the line in any direction will be in results.
- queryProperties/direction="left" means that the object is to cross the line from the right to the left if look from the right point of the segment.
- queryProperties/direction="right" means that the object is to cross the line from the left to the right if look from the right point of the segment.

.

```
{
  "queryType": "line",
  "figures": [
     {
        "shape": [
           [0.5, 0.8],
           [0.5, 0.2]
        ]
     }
  ],
  "queryProperties": {
     "direction": "left"
  }
}
```

‘Familiar face’-‘stranger’ face search API

Rus

This search type compares every recognized face with all faces in the camera database over 30 days (or for the current archive depth if it is less than 30 days) and sets the number of days over which this face was recognized by the camera. The search decides if this is a “familiar face” or a “stranger” by the specified criteria.

The POST request is used for search start (see Search request), search type is **stranger**, request body is empty.

The following parameters are available:

- **accuracy** – sets face similarity level in the range [0,1] **(1** means complete match). If this parameter is not set, then the default value (**0.9**) is in use. If the compared face was in the camera field of view on a specific day and it was recognized with accuracy that is not less than specified one, then this face is considered to be present on that day. Otherwise, the algorithm considers this face was absent on that day.
- **threshold** – defines the threshold value to recognize a face as a “stranger”. The value is set in the range from **0** to **1** and it defines the number of days within which the face was absent to be considered as a “stranger”: 30-30***threshold**. For instance, the value **0.8** means “the required object appeared in the search area within (30 - 30 * 0.8 = 6) days”. All faces that appeared within 6 and more days will be defined as “familiar faces”, others – as “strangers”.
- **op** – defines search direction.
  Allowable values:
  lt – “familiar face” search (based on **treshold** parameter).
  gt – “stranger” search.

Important!
The **threshold** and **op** parameters should **only** be used together. If any of parameters is not set or has incorrect

value, then both parameters will be ignored.

JSON search result looks like this:


```
{
  "events" : [
    {
      "rate" : 0.90591877698898315,
      "origin" : "hosts/SERVER1/DeviceIpint.2/SourceEndpoint.video:0:0",
      "position" : {
        "bottom" : 0.10694444444444445,
        "left" : 0.69687500000000002,
        "right" : 0.74687500000000007,
        "top" : 0.018055555555555554
      },
      "timestamp" : "20160914T085307.499000"
    },
    {
      "rate" : 0.90591877698898315,
      "origin" : "hosts/SERVER1/DeviceIpint.2/SourceEndpoint.video:0:0",
      "position" : {
        "bottom" : 0.10694444444444445,
        "left" : 0.69687500000000002,
        "right" : 0.74687500000000007,
        "top" : 0.018055555555555554
      },
      "timestamp" : "20160914T085830.392000"
    }
}
```


where

- **origin** - camera channel to take analyzed video stream from.
- **timestamp** - time stamp of a frame with a face detected by the detection tool.
- **rate** - rate of identifying a face as a "stranger", the value in the [0,1] range. **1** means a complete stranger.
- **position** – coordinates of a frame border enclosing face on a video frame.

Define 'familiar face'-'stranger' attribute from image

Rus

'Familiar face'-'stranger' face search API

The body of POST request used for search start must contain binary data of searched face in jpeg format. The request itself can be represented in two ways:

1. POST http://IP-Address:port/prefix/faceAppearanceRate/DETECTORID/BEGINTIME/ENDTIME
   where
   DETECTORID - three-component detector endpoint ID (HOSTNAME/A VDetector.ID/EventSupplier).
   ENDTIME, **BEGINTIME** - time in ISO format.
2. POST http://IP-Address:port/prefix/faceAppearanceRate/HOSTID/BEGINTIME/ENDTIME
   where **HOSTID** - computer name.

The **accuracy** parameter is specified additionally – detection accuracy in the range [0,1] **(1** means complete match). If this parameter is not set, then the default value (**0.9**) is in use.

This request is performed synchronously and returns JSON:


```
{
  "rate": 0.13333334028720856
}
```


where **rate** – rate of identifying a face as a "stranger", the value in the [0,1] range. **1** means a complete stranger.

Get signed links to video streams

Rus

Add 2 parameters to the request in order to get pre-authorized and signed links to video streams (both live and archive video):

- **enable_token_auth** - enable authorization by token **=1**.
- **valid_token_hours** - signature validation time (in hours). The maximum value is a week. The default value is 12 hours.

Sample:

http://127.0.0.1:8000/live/media/NGP/DeviceIpint.60/SourceEndpoint.video:0:0?w=800&h=0&format=mjpeg&vc=3&enable_token_auth=1&valid_token_hours=1

Get camera live stream

Get archive stream

Get list of groups and their contents

Rus

Get list of all available groups

GET http://IP-Address:port/prefix/group

Sample response:


```
{
  "groups" : [
    {
      "Brief" : "Group1",
      "Description" : "",
      "Id" : "35fc84a0-2280-4b30-acd2-cc8419a2dc68",
  "ObjectCount"" : "14"
      "groups" : [
        {
          "Brief" : "Group2",
          "Description" : "",
          "Id" : "dac24803-313c-43ab-aa9a-276922a55cb6",
    "ObjectCount"" : "5"
          "groups" : []
        },
        {
          "Brief" : "Group3",
          "Description" : "",
          "Id" : "13764152-6910-44b6-99b5-f74641ad4a14",
    "ObjectCount"" : "3"
          "groups" : [
            {
              "Brief" : "Group4",
              "Description" : "Group4",
              "Id" : "9a64e2a0-eb92-4adc-bc4f-81d30ceb6c2f",
    "ObjectCount"" : "6"
              "groups" : []
            }
          ]
        }
      ]
    }
  ]
}
```


ObjectCount – number of video cameras in the group.

Get group contents

GET http://IP-Address:port/prefix/group/GROUPID

where **GROUPID** – value of the **Id field** received using the previous request.

Sample response:

```
{
  "members" : [ "hosts/SERVER1/DeviceIpint.1/SourceEndpoint.video:0:0" ]
}
```

Get list of groups containing specified camera

GET http://IP-Address:port/prefix/group/contains/VIDEOSOURCEID

where **VIDEOSOURCEID** - three-component source endpoint ID (see Get list of video sources (cameras)).

Sample:

```
http://localhost:8000/group/contains/SERVER1/DeviceIpint.1/SourceEndpoint.video:0:0
```

Sample response:

```
{
  "groups" : [
    "35fc84a0-2280-4b30-acd2-cc8419a2dc68",
    "13764152-6910-44b6-99b5-f74641ad4a14",
    "dac24803-313c-43ab-aa9a-276922a55cb6"
  ]
}
```

Export

Rus

On page:

- Export start
- Get export status
- Export completion
- Download file

Export start

Export is initiated using one of the following POST requests:
http://IP-Address:port/prefix/export/archive/SERVER1/VIDEOSOURCEID/BEGINTIME/ENDTIME - archive export
http://IP-Address:port/prefix/export/live/SERVER1/VIDEOSOURCEID/BEGINTIME/ENDTIME - live video export

where **BEGINTIME** and **ENDTIME** set time in the YYYYMMDDTHHMMSS format. If **BEGINTIME is greater than ENDTIME**, then the values will swap. **BEGINTIME must be equal to ENDTIME for snapshot export.**

Complex data structures are in use to describe frames and masks. These structures can be divided into several types:

- **point** is set using *x,y* syntax: sample - [0.5, 0.5].
- **area** sets square frame; defined by two points ! separated. Sample - [[0.5, 0.5], [0.4,0.4]].
- **polygon** sets closed curve; contains at least 3 points enclosed in []. Sample - [[0.5, 0.5], [0.4,0.4],[0.3,0.3]].
- **mask** is a collection of polygons. Sample - [[[0.5, 0.5],[0.6, 0.6],[0.7, 0.7]], [[0.1, 0.1],[0.2, 0.2],[0.3, 0.3]]].

Supported parameters that are sent in the body of initial POST request:
Expand the list

1. **format** (string) is a **mandatoryparameter**; available values are mkv, avi, exe, jpg and pdf. It sets the format of output export container.
2. **maxfilesize** (number) is a maximum export file size (in bytes). A new file will be created when the size limit is exceeded. Export results in the collection of files. Default value is 0 (as a result, a single file).
3. **vc**, **ac** (number) is a compression quality level for video and audio respectively. Allowed values are from 0 to 6 (6 means the worst). The initial quality level will be set when 0 is specified (original API fault). Default value is 0.
4. **freq** (number) – frame rate of the output stream. Default value is 0. Available values:
    a. **0** – original
    b. **1** – half of original
    c. **2** - quarter of original
    d. **3** – one-eighth of original
5. **tsformat** (string) is a time stamp format template. Any string can be generated http://www.boost.org/doc/libs/1_55_0/doc/html/date_time/date_time_io.html. Default value is %Y-%b-%d %H:%M:%S.

    Important!
    Server does not check the format of the input string.
6. **croparea** (area) is a snapshot area for export (Default value is 0,0!0,0 – the entire snapshot).
7. **maskspace** (mask) is a snapshot mask space. By default a snapshot is not masked.
8. **color** (string) is a color of a comment and time stamp text. It is set in the #FFFFFF format.
9. **comment** (string) is a comment.

    Parameters relevant to PDF format only.

1. **snapshotplace** (area) is a snapshot location on the page
2. **commentplace** (area) is a comment location on the page
3. **tsplace** (area) is a time stamp location on the page

layout (number) – page layout. Available values are **0** (portrait), **1** (landscape).

The request will result in an error response or response that looks like:

```
HTTP/1.1 202 Accepted
Connection: Close
Location: /export/3dc15b75-6463-4eb1-ab2d-0eb0a8f54bd3
Cache-Control: no-cache
```

Note.
Possible error codes:

- **400** - incorrect request.
- **500** - Server internal error.

Get export status

GET http://IP-Address:port/prefix/export/id/status

where **id** is the value from the **Location** field (here 3dc15b75-6463-4eb1-ab2d-0eb0a8f54bd3)

Sample response:

```
{
  "id": "73c2e1d2-0f8f-414c-9cc0-ac5fb43cd8dd"
  ,"state": 1
  ,"progress": 0.51062298
  ,"error": ""
  ,"files":
  [
  ]
}
```

where

- **state** defines the current state of export. Available values:
  1 – export is performed

2 – export is completed
3 – export error
4 – not enough space to complete the operation

- **progress** – progress of export session in the range from 0 to 1.
- **error** – description of error (if any)
- **files** - the list of files created as the export result

Export completion

DELETE http://IP-Address:port/prefix/export/id

where **id** is the value from the **Location** field.

Download file

GET http://IP-Address:port/prefix/export/id/file?name=SERVER_DeviceIpint.1[20160527T132900-20160527T133000].mkv

where

- **id** is the value from the **Location** field
- **file?name** is the name of file from the **files** field

Get list of detection tools

Rus

GET http://IP-adress:port/prefix/detectors/DeviceIpint.N

where **N** - video camera id (см. Get list of video sources (cameras)).

Sample response:

```
{
   "detectors": [
      {
         "name": "hosts/SERVER1/AVDetector.1/EventSupplier",
         "type": "SceneDescription"
      },
      {
         "name": "hosts/SERVER1/AVDetector.12/EventSupplier",
         "type": "NullAudioDetection"
      }
   ]
}
```

Available values of **type** parameter:

SceneDescription	Situation Analysis Detection Tools
FireDetector	Smoke detection
SmokeDetector	Fire detection
LprDetector	Automatic Number Plate Recognition
TvaFaceDetector	Face detection
QualityDegradation_v2	Image Noise Detection
QualityDegradation	Loss of quality
SceneChange	Position change

BlurredDegradation	Blurred Image Detection
MotionDetection	Motion detection
CompressedDegradation	Compression Artifacts Detection
SignalAudioDetection	Signal
NoiseAudioDetection	Noise
NullAudioDetection	No signal

Types of Video Detection

Types of Audio Detection

Situation Analysis Detection Tools

Get info about triggering of detection tools and alarms

Get list of alarms

Rus

GET http://IP-Address:port/prefix/archive/events/alerts/VIDEOSOURCEID/ENDTIME/BEGINTIME?limit=COUNT&offset=COUNT – gets the list of alarms. If *limit* is not specified, it is equal to 100. Field **raisedAt** is not unique so passing of previously received alarms starting from the search interval can be requested.

Sample response:

```
{
   "events" :
   [
      {
          "type": "alert",
          "id": "42D43A79-90D6-4ba7-BD23-1714996A2F88",
          "raisedAt": "20101230T103950.000",
          "zone": "SERVER1/DeviceIpint.3/SourceEndpoint.video:0:0",
          "reasons": ["ruleAlert", "videoDetector"],
          "initiator": "plateRecognized",

          "reaction":
          {
             "user": "root",
             "reactedAt": "20101230T103958.000",
             "severity": "alarm"
          },

         "detectorName": "5.License Plate Recognition Detection"

      },
      ...
   ],
   "more": true
}
```

Possible values of **reasons** array: armed, disarmed, userAlert, ruleAlert, videoDetector, audioDetector, ray.

Possible values of field **severity**: unclassified, false, notice, warning, alarm.

Get list of detection tools events

Rus

GET http://IP-Address:port/prefix/archive/events/detectors/VIDEOSOURCEID/ENDTIME/BEGINTIME?limit=COUNT&offset=COUNT – gets the list of detection tool events. If *limit* is not specified, it is equal to 100. Field **timestamp** is not unique so passing of previously received alarms starting from the search interval can be requested.

Sample response:

```
{
"events":
```

```
[
{
"id": "433d45ec-0b7f-aa43-8491-c8acb7d0ac56"
,"source": "hosts/SERVER1/DeviceIpint.3/SourceEndpoint.video:0:0"
,"origin": "hosts/SERVER1/DeviceIpint.3/EventSupplier.analytics:0:motion_detection_snb_5001"
,"detectorId": "1"
,"type": "oneLine"
,"alertState": "ended"
,"timestamp": "20120314T121512.597"
,"rectangles":
[
{
"index": "1"
,"left": "0.622086710929871"
,"top": "0.68798337459564196"
,"right": "0.65736908435821495"
,"bottom": "0.79889315128326399"
}
]
},
...
],
"more": true
}
```


In this response the VIDEOSOURCEID can be:

– as usual, consisting of three components, e.g. - "HOST/DeviceIpint.2/EventSupplier.ray0:0";

– host name to receive events from it;

– empty, i.e. be missing to receive all domain events.

Sample response:

http://IP-Address:port/prefix/archive/events/detectors/**HOST/DeviceIpint.2/EventSupplier.ray0:0**/past/future?limit=10&offset=0 - gets sensor events sorted descending. Maximum number is 10.

http://IP-Address:port/prefix/archive/events/detectors/**HOST**/past/future?limit=5&offset=0 - gets events of all detection tools created on the HOST machine. Maximum number by every detection is 5.

http://IP-Address:port/prefix/archive/events/detectors/**HOST**/past/future?limit=5&offset=0&**type=Ray** - gets events of all sensors created on the HOST machine. Maximum number by every sensor is 5.

http://IP-Address:port/prefix/archive/events/detectors/past/future?limit=1&**type=Ray** - gets the current state of all domain sensors.

Available values of type parameter:

- SceneChangeDetected;

- CameraBlindDetected;

- Disconnected;

- MotionDetected;

- NullAudio;

- NoiseAudio;

- SignalAudio;

- Ray;

- oneLine;

- comeInZone;

- lostObject;

- outOfZone;

- longInZone;

- moveInZone;

- stopInZone;

- faceAppeared;

- plateRecognized.

Note
If the non-included in the list value is received, then an embedded detection tool triggered.

Telemetry control

Get list of telemetry devices for specified video source

Rus

GET http:/IP-Address:port/prefix/control/telemetry/list/OBJECTID - Gets the list of telemetry devices for video sources where OBJECTID is object identifier based on 2 components (HOSTNAME/ObjectType.Id).

Sample response:

[
"SERVER1/DeviceIpint.2/TelemetryControl.0"
]

TELEMETRYCONTROLID template will be used to define telemetry devices of the HOSTNAME/ObjectType.Id/TelemetryContol.n type hereinafter.

Acquire telemetry control session

Rus

GET http://IP address:port/prefix/control/telemetry/session/acquire/[server_name]/[device_name]/[telemetry_name]?session_priority=[priority],

where server_name - Server name (see Get server list);

device_name - device name (see Get list of video sources (cameras));

telemetry_name - name of telemetry device (see Get list of telemetry devices for specified video source);

priotity - telemetry control priority from 1 (maximum) to 5 (minimum).

If a telemetry device is not in use or a user with lower priority controls it , then telemetry control is taken over and Server sends the response:

{

"session_id" : [id]

}

where id is the session id.

Keep session alive

Rus

GET http://IP address:port/prefix/control/telemetry/session/keepalive/[server_name]/[device_name]/[telemetry_name]?session_id=[id],

where server_name - Server name (see Get server list);

device_name - device name (see Get list of video sources (cameras));

telemetry_name - name of telemetry device (see Get list of telemetry devices for specified video source);

id - id of captured session (see Acquire telemetry control session).

Such request is to be sent not less than once in 10 seconds in order to keep the session alive. Otherwise telemetry control fails.

Release session

Rus

GET http://IP address:port/prefix/control/telemetry/session/release/[server_name]/[device_name]/[telemetry_name]?session_id=[id],

where server_name - Server name (see Get server list);

device_name - device name (see Get list of video sources (cameras));

telemetry_name - name of telemetry device (see Get list of telemetry devices for specified video source);

id - id of captured session (see Acquire telemetry control session).

Control degrees of freedom

Rus

On page:

- Get info of degrees of freedom
- Edit tilt and pan
- Edit degree of freedom
- Capture screen point
- Zoom image area
- Auto focus and iris

Get info of degrees of freedom

GET http://IP-Address:port/prefix/control/telemetry/info/TELEMETRYCONTROLID - Gets info about supported controllable degrees of freedom and the ways of controlling them (continuous, relative, discrete) and their max values.

Sample response:


```
{
"degrees":
{
"tilt":
{
"relative": {"min": "-45", "max": "45"},
"continuous": {"min": "-10", "max": "10"}
},
"pan":
{
"absolute": {"min": "-170", "max": "170"},
"continuous": {"min": "-10", "max": "10"}
},
"zoom":
{
"absolute": {"min": "0", "max": "20"}
}
},
"feature": ["autoFocus", "areaZoom", "pointMove"]
}
```


degrees – information about degrees of freedom. (tilt, pan, zoom, focus, iris). Every degree of freedom contains the list of supported ways of control (absolute, relative, continuous).

feature – list of supported functions (autoFocus, autoIris, areaZoom, pointMove).

Edit tilt and pan

GET http://IP-Address:port/prefix/control/telemetry/move/TELEMETRYCONTROLID?parameters&session_id=N – changes tilt, pan.

Parameters:

mode – way of control (absolute, relative, continuous);

pan, **tilt** – values for corresponding degrees of freedom;

hereinafter **session_id=N** - session id (see Acquire telemetry control session).

Sample request: GET http://IP-Address:port/prefix/control/telemetry/move/HOSTNAME/DeviceIpint.25/TelemetryControl.0?mode=absolute&pan=-99&tilt=10&session_id=0

Edit degree of freedom

GET http://IP-Address:port/prefix/control/telemetry/**{degree}**/TELEMETRYCONTROLID?parameters&session_id=N – changes one of degrees (zoom, focus, iris).

Parameters:

{degree} – degree of freedom to be updated (zoom, focus, iris);

mode – way of control (absolute, relative, continuous);

value - value.

Sample request:

GET http://IP-Address:port/prefix/control/telemetry/zoom/HOSTNAME/DeviceIpint.25/TelemetryControl.0?mode=absolute&value=6&session_id=0 - zoom change;

GET http://IP-Address:port/prefix/control/telemetry/focus/HOSTNAME/DeviceIpint.25/TelemetryControl.0?mode=relative&value=3&session_id=0 - focus change;

GET http://IP-Address:port/prefix/control/telemetry/iris/HOSTNAME/DeviceIpint.25/TelemetryControl.0?mode=continuous&value=1&session_id=0 - iris change.

Capture screen point

GET http://IP-Address:port/prefix/control/telemetry/move/point/TELEMETRYCONTROLID?parameters&session_id=N – captures the screen point.

Parameter:

x,y – values of vertical and horizontal coordinates, specified in relation to image size;

Sample request: GET http://IP-Address:port/prefix/control/telemetry/move/point/HOSTNAME/DeviceIpint.23/TelemetryControl.0?x=0.14&y=0.32&session_id=0

Zoom image area

GET http://IP-Address:port/prefix/control/telemetry/zoom/area/TELEMETRYCONTROLID?parameters&session_id=0 – zooms selected image area.

Parameters:

x,y – left upper corner of selected area;

w,h – width and height of area.

Coordinates and dimensions are specified in relation to image size.

Sample request:

GET http://IP-Address:port/prefix/control/telemetry/zoom/area/HOSTNAME/DeviceIpint.24/TelemetryControl.0?x=0.23&y=0.089&w=0.25&h=0.25&session_id=0

Auto focus and iris

GET http://IP-Address:port/prefix/control/telemetry/auto/TELEMETRYCONTROLID?parameters&session_id=0 – auto focus/iris.

Parameters:

degree – is focus or iris.

Sample request:

GET http://IP-Address:port/prefix/control/telemetry/auto/HOSTNAME/DeviceIpint.24/TelemetryControl.0?degree=iris&session_id=0

Preset control

Rus

On page:

- Get list of presets
- Create and edit preset
- Go to preset and delete preset

Get list of presets

GET http://IP-Address:port/prefix/control/telemetry/preset/info/TELEMETRYCONTROLID- gets the list of existing presets.

Sample request:

GET http://IP-Address:port/prefix/control/telemetry/preset/info/HOSTNAME/DeviceIpint.23/TelemetryControl.0

Sample response:

{
"0": "Corridor",

"1": "Entrance",
"4": "Hole in fence"
}

Create and edit preset

GET http://IP-Address:port/prefix/control/telemetry/preset/set/TELEMETRYCONTROLID?parameters&session_id=N - create/edit preset.

Parameters:

pos - position;

label – preset name;

hereinafter **session_id=N** - session id (see Acquire telemetry control session).

If a preset with specified position already exists, its label will be deleted.

Sample request:

GET http://IP-Address:port/prefix/control/telemetry/preset/set/HOSTNAME/DeviceIpint.23/TelemetryControl.0?pos=0&label=Exit&session_id=0

Go to preset and delete preset

GET http://IP-Address:port/prefix/control/telemetry/preset/**{action}**/TELEMETRYCONTROLID?parameters&session_id=N - go to preset or delete it.

Parameters:

{action} – can be **go** or **remove** and is used to go to preset or delete it;

pos – preset position.

Sample request:

Going to preset with pos 1:

GET http://IP-Address:port/prefix/control/telemetry/preset/go/HOSTNAME/DeviceIpint.23/TelemetryControl.0?pos=1&session_id=0

Deleting of preset with position 2:

GET http://IP-Address:port/prefix/control/telemetry/preset/remove/HOSTNAME/DeviceIpint.23/TelemetryControl.0?pos=2&session_id=0

Get information about errors

Rus

When errors occur with telemetry requests, there will be the { "error_code" : [numeric error code] } response.

Possible error codes:

1 - General error, details are in Server logs.
2 - Request parameters are set incorrectly.
3 - Telemetry control session is not available.
4 - Preset control error.

Working with layouts and videowalls

Sequence of actions

Rus

Before one starts working with layouts and videowalls HTTP API, run this command in the command prompt:

```
netsh http add urlacl url=http://+:8888/ user=DOMAIN\username
```

where DOMAIN\username relates to Windows (**whoami** command in the command prompt). Port 8888 is to be vacant, antivirus and/or firewall are to be disabled.

> ⚠ **Attention!**
> All requests for working with layouts and videowalls are performed on the Client that is to be run as administrator.

The Client's IP-Adress is to be specified in the requests.

Attention!
If there are any errors, run the command prompt as administrator and then run the command once again.

When the command is successfully executed, one can make requests listed below.

Getting the list of layouts

Rus

GET http://IP-adress:8888/GetLayouts - getting available layouts for current logged user.

Here is an example of response:

```
{
   "Description": "",
   "Status": "OK",
   "LayoutInfo": [
      {
         "Id": "102",
         "Name": "Layout name 2"
      },
      {
  "Id": "103",
         "Name": "Layout name 3"
      }
   ]
}
```

Here is an example of an error message:

```
 {
\"result\":\"no layouts\"
}
```

Note
An error can occur while requesting the list of Server layouts if the UAC is enabled on the Server. Disable this function in order to eliminate the error.

Switching the layout on the screen

Rus

GET http://IP address:8888/SwitchLayout?layoutId=N&displayId=\\.\DISPLAY1 - layout with N ID is selected on DISPLAY1 screen.

Here is an example of response:

```
{
   "Description": "",
   "Status": "OK"
}
```

Here is an example of an error message:

```
{
\"result\":\"error\"
}
```

Note
An error can occur if a layout with non-existent ID is specified.

Getting the list of cameras displayed on the layout

Rus

GET http://IP-address:8888/GetCameras?layoutId=N&displayId=\\.\DISPLAY1 - getting the list of cameras from layout N of DISPLAY1 for current logged user.

Here is an example of response:


```
{
   "Description": "",
   "Status": "OK",
   "CameraInfo": [
      {
         "DisplayName": "1.Camera",
         "Id": "1",
         "Name": "host/HOSTNAME/DeviceIpint1/SourceEndPoint.video:0:0"
      },
      {
         "DisplayName": "2.Camera",
         "Id": "2",
         "Name": " host/HOSTNAME/DeviceIpint2/SourceEndPoint.video:0:0"
      }
   ]
}
```


Note
If the layout with specified id will not be found, then the query will return the list of cameras of the current layout for the specified display.

Adding and removing cameras

Rus

GET http://IP-address:8888/RemoveCamera?displayId=\\.\DISPLAY1&cameraName=Name - removing a camera from the current layout of DISPLAY1.

GET http://IP-address:8888/RemoveAllCameras?displayId=\\.\DISPLAY1 - removing all cameras from the current layout of DISPLAY1.

GET http://IP-address:8888/AddCamera?displayId=\\.\DISPLAY1&cameraName=Name - adding a camera to the current layout of DISPLAY1.

where Name - camera name from the response to Getting the list of cameras displayed on the layout request.

Here is an example of response to all requests:

{ "Description": "", "Status": "OK" }

Here is an example of an error to all requests:

{ "Description": "Error description", "Status": "ERROR" }

Getting the list of displays

Rus

GET http://IP-address:8888/GetDisplays - getting available displays for current logged user.

Here is an example of response:

```
{
   "Description": "",
   "Status": "OK",
   "DisplayInfo": [
      {
         "Id": "\\\\.\\DISPLAY1",
         "IsMainForm": true
      },
      {
         "Id": "\\\\.\\DISPLAY2",
         "IsMainForm": false
      }
   ]
}
```


where

id - display ID;

IsMainForm - the 'true' value corresponds to the main display.

Here is an example of an error message:

{

"{\"result\":\"no displays\"}"

}

Selecting active display

Rus

GET http://IP-address:8888/SelectDisplay?displayId=\\.\DISPLAY1 - selecting an active display.

Here is an example of Server response:


```
{
"Description": "",
"Status": "OK"
}
```


Here is an example of an error message:

"{\"result\":\"error\"}"

Switching camera to archive mode

Rus

GET http://IP address:8888/GoToArchive?displayId=\\.\DISPLAY1&cameraName=Name×tamp=Timestamp

where

- **displayId** – ID of display in GetDisplays request (see Getting the list of displays).
- **cameraName** – name of the camera received as a result of GetCameras request (see Getting the list of cameras displayed on the layout).
- **timestamp** – time in ISO format.

Sample request:

GET http://localhost:8888/GoToArchive?displayId=\\.\DISPLAY2&cameraName=hosts/SERVER1/DeviceIpint.1/SourceEndpoint.video:0:0&timestamp=2017-04-07T00:00:00.000

Using macros

Rus

GET http://IP-Address:port/prefix/macro/list/ - getting the list of macros.

JSON sample response:


```
{
  "macroCommands" : [
    {
      "id" : "04eb71b0-e2e0-445e-ae7a-a036951fb595",
      "name" : "MacroName1"
    },
    {
      "id" : "3fd3bfb0-3a6e-467a-8ff2-88f7b165cf5b",
      "name" : "MacroName2"
    },
    {
      "id" : "941f88d1-b512-4189-84a6-7d274892dd95",
      "name" : "MacroName3"
    }
  ]
}
```


GET http://IP-Address:port/prefix/macro/execute/id - executing macro

where id is an id form the list of macros.

Possible error codes when executing macros:

- **400** - incorrect request.
- **500** - Server internal error.
- **404** - incorrect id (only for execute macro)

Get data from system log

Rus

GET http://IP-address:port/prefix/audit/HOST/beginTime/endTime?filter=17-20,6,1:4

where

- **HOST** – the name of Server events from which are to be received.
- **beginTime** and **endTime** set time in the YYYYMMDDTHHMMSS format in the timezone UTC+0.
- **filter** – the list of events that can be represented both as a range separated with <-> and <:> and a simple code.

The list of event types:

Click here to expand...

1 - not shown in the event log
2 - violation of BOT of audit events (e.g., it is edged manually)
3 - violation of EOT of audit events (e.g., it is edged manually)
4 - some audit events are absent (e.g., they are deleted manually)
5 - table entry is changed
6 - inactive log (no events or log update mark)
/// External events
7 - user is added
8 - user is deleted
9 - user parameters are changed
10 - role is added
11 - role is deleted
12 - role parameters are changed
13 - user login
14 - user logout
15 - device is added
16 - device is deleted
17 - device parameters are changed
18 - detection is added
19 - detection is deleted
20 - detection parameters are changed
21 - archive is added
22 - archive is deleted
23 - archive parameters are changed
24 - detection rule (macro) is created
25 - detection rule (macro) is deleted
26 - detection rule (macro) parameters are changed
27 - alarm is triggered

28 - zone is armed
29 - zone is disarmed
30 - export from archive is performed
31 - notification sender (sound-, email-, sms-) is added
32 - notification sender is deleted
33 - notification sender parameters are changed
34 - general parameter is changed
35 - recording to the archive parameters are changed
36 - agent of export is added
37 - agent of export is deleted
38 - agent of export parameters are changed
39 - macro is created
40 - macro is deleted
41 - macro parameters are changed
42 - alarm is processed by user
43 - dangerous alarm
44 - suspicious alarm
45 - false alarm
46 - skipped alarm
47 - Server is included into Axxon-domain
48 - Server is excluded from Axxon-domain
49 - view archive
50 - view camera
51 - view layout
52 - forensic search in the archive
53 - area search by faces in the archive
54 - area search by license plates in the archive
55 - system log export
56 - LDAP folder is added
57 - LDAP folder is deleted
58 - LDAP folder parameters are changed

JSON sample response:


```
{
 "events": [
  {
   "data": {
    "component": "Camera3",
    "componentType": "camera",
    "device": "Camera3",
    "host": "V-SHMELEV",
    "property": "vstream-virtual/folder",
    "setting": "Directory",
    "value": "D:/Movies/Spirit"
   },
   "eventType": 17,
   "timestamp": "20161205T120410.698000"
  },
  {
   "data": {
    "detector": "Face detection",
    "device": "Camera1",
    "host": "V-SHMELEV"
   },
   "eventType": 18,
   "timestamp": "20161205T120459.319000"
  }
 ]
}
```


Get statistics

Rus

GET http://IP-Address:port/prefix/statistics/HOSTNAME/DeviceIpint.23/SourceEndpoint.video:0:0 - gets statistics for the specified video source.

GET http://IP-Address:port/prefix/statistics/webserver - gets statistics for server.

Get info about Server usage

Rus

GET http://IP-address:port/prefix/statistics/hardware - get information about usage of network and CP of a specific Server.

GET http://IP-address:port/prefix/statistics/hardware/domain - get information about usage of network and CP of all Servers within Axxon Domain.

Sample response:


```
[
 {
  "drives": [
   {
    "capacity": 523920994304,
    "freeSpace": 203887943680,
    "name": "C:\\"
   },
   {
    "capacity": 475912990720,
    "freeSpace": 148696813568,
    "name": "D:\\"
   },
   {
    "capacity": 0,
    "freeSpace": 0,
    "name": "E:\\"
   }
  ],
  "name": "SERVER1",
  "netMaxUsage": "0,0062719999999999998",
  "totalCPU": "16,978111368301985"
 }
]
```


Get info about Server version

Rus

GET http://IP-adress:port/prefix/product/version

Sample response:


```
{
"version": "AxxonNext 4.0.2.4483"
}
```