Netavis Observer 5.0.1

SNAP URL Interface

Netavis SNAP URL Interface for Observer 5.0.1

Valid from Observer 5.0.1
Published in June 2019

Copyright

Netavis Software GmbH
Handelskai 388, Top 221
A-1020 Vienna
Austria

Tel. +43 1 503 1722 - 0
Fax. +43 1 503 1722 - 400
info@netavis.net
www.netavis.net

Contents

1. Introduction

The Netavis URL Command API is a simple and effective way to interface a 3rd party application (e.g. a company's intranet/internet site) to a Netavis Observer video management system.

Netavis URL commands are simple HTTP URL requests and responses. Both HTTP GET and POST methods are supported.

The URL Command API is also very suitable for the limited capabilities of mobile devices like smart phones, PDAs and tablets. For example, live and archive video streams are delivered as multipart HTTP streams that are well supported by such devices.

Format of an URL command

GET

A Netavis URL command looks like a standard HTTP request:

http://192.168.7.221/urlapi/live?SessionID=1 where

- http://192.168.7.221 is the server root context
- /urlapi/live is the resource base
- ? is the parameter separator (or & for the following parameters)
- SessionID=1 is the parameter part

The server root context may vary, depending on routers, firewalls or other user specific or environment setup conditions. The server ip/name may be followed by a colon and a port number.

The Netavis URL commands are grouped by resource bases:

- Live streaming
- Archive streaming
- Event streaming
- Control query commands

Resources and parameters are case sensitive.

POST

Same format as the previous GET request, but the parameters should be added to the request body, in the format of **application/x-www-form-urlencoded** mime type.

For example:

```
POST 192.168.7.221/urlapi/live HTTP/1.0\r\n
Content-Type: application/x-www-form-urlencoded\r\n
Content-Length: 22\r\n
\r\n
SessionID=1&User=admin\r\n
```

2. Live streaming

2.1 Getting a live stream from a camera

Resource base: /urlapi/live
Response Mime-Types: **image/jpeg** for snapshots and
multipart/x-mixed-replace for streams
application/json for errors

Parameter	Group	Description	Default Values
User	*Access*	The name of a valid Netavis User	default: 'guest'
Passwd		The MD5 hash coded value of the password	default: MD5('guest')
SessionID	*Access*	Optionally you can use an already opened session (e.g. via the OpenSession command) by supplying this parameter. For security reason, the session which is identifying itself via this parameter has to come from the same IP where it had been opened.	
EntityID		Unique ID of the camera	Mandatory
Size	*Format*	One of: Small, Medium, Large or WxH. WxH is width and height in pixels. The server tries to find and use the closest resolution to the given WxH.	default: Large
Quality		One of: Low, Medium, High	default: High
Fps		Frame per second	default: 5
Type		One of: JPEG, MPEG, H264	default: JPEG
Snapshot		Use when a snapshot is needed from the camera only	default: false
Multipart		Use when a stream of images is needed	default: true
OutputType		One of: JPEG, MPEG, MXPEG, H264. If given, the server transcodes the input stream to the required output format	
OutputSize		In case of transcoding this parameter defines the WxH value for the output image size in pixels	
StreamID		Supply all three parameters to close (from a parallel running program or thread) an opened and streaming (in multipart) SNAP channel	
ChannelID			
Close			

Examples

1. Requesting for a live 'Snapshot' of camera '5' as 'guest' user
   http://192.168.7.221/urlapi/live?EntityID=5&Snapshot=true

2. Requesting for a live stream with the user named 'admin' in a multipart motion JPEG stream from the camera '38' with the encoded password of 'admin' ('21232f297a57a5a743894a0e4a801fc3') in 'High' quality 'Medium' size and in '5' Fps:
   http://192.168.7.221/urlapi/live?User=admin&Passwd=21232f297a57a5a743894a0e4a801fc3&EntityID=38&Size=Medium&Quality=High&Fps=5&Type=JPEG

3. Request via encrypted credentials:
   http://192.168.7.221/urlapi/live?Login=Xx1D67xZKVYTgA2mWh6TJxBLkD5Jek%2FVrouDkScNiN4%3D&EntityID=38&Size=Medium

4. Request for the same live stream using the earlier given SessionID '1':
   http://192.168.7.221/urlapi/live?SessionID=1&EntityID=38&Size=Medium

5. A positive multipart HTTP sample response:

```
HTTP/1.0 200 OK\r\n
Content-Type: multipart/x-mixed-replace; boundary=--boundary\r\n
\r\n
--boundary\r\n
Content-Type: image/jpeg\r\n
Content-Length: <image size>\r\n
\r\n
<JPEG image data>\r\n
--boundary\r\n
...
```

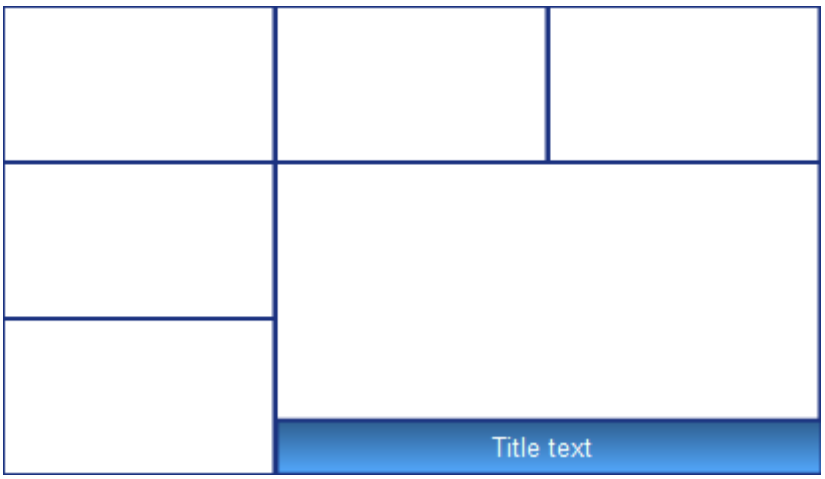
6. Negative “application/json” mime-type sample response :

```
{"SNAP": {"ExecutionStatus": {
    "ErrorText": "Illegal user/passwd. User=admin1",
    "ReturnCode": 103
}}}
```

2.2 Getting a live view stream

This API call has been designed for mobile devices with limited bandwidth and limited browser capabilities. The live view feature does all the hard work on the server-side. It creates a canvas (for the size of the screen of the mobile device) with a grid of cameras, collects and renders the images from each camera separately and sends only the final view to the mobile client, thus saving bandwidth and CPU on the client side.

The following figure depicts the main building-blocks of a view:



The whole area is called “canvas” which has the following attributes:

- width (c_w),
- height (c_h),
- background color (c_bg).

The canvas is divided into a “grid”. The attributes of a grid are:

- number of columns (g_c),
- number of rows (g_r),
- width of divider (g_g),
- color of divider (g_fg).

Each grid area can have a camera inside it or multiple rows and columns can be stitched together. Such an area is called a “viewport”. Each viewport has the following attributes:

- starting column,
- starting row,
- span of columns (vp_x_y_c),
- span of rows (vp_x_y_r),
- identifier of the camera it should contain (vp_x_y_eid),
- fill mode (vp_x_y_f={LETTERBOX|CROP|STRETCH}),
- title text (vp_x_y_tt),
- title line height (vp_x_y_th),
- title text color (vp_x_y_tfg),
- title line background colour (vp_x_y_tbg),
- title line position (vp_x_y_tp={OFF|TOP|BOTTOM}).
- title line transparency (0-100) (vp_x_y_ttr),

For a complete reset or remove of all view-ports use the v_ra=1 parameter.

The symbols in the parenthesis after the attribute are the keys which should be used in the URL command to set a given value for the attribute (e.g. g_r=2 to set the number of rows of the grid to two). Colors are hex encoded and should be given in the form: #rrggbb (e.g. g_fg=#ff0000 for setting the color for the grid divider to pure red). The “x_y” in the viewport keys represent the starting column and row, e.g. all keys starting with vp_0_0 will give the values for the viewport starting at 0,0.

Viewports cannot overlap. When a viewport is configured with zero camera ID, the viewport is removed from the view. A new ID causes the removal of the previous camera and the assignment of the new one.

If you submit your initial URL with no “Snapshot” and “Multipart” keys a so-called configuration view is created in the server. This view can later be modified and in a separate thread the client can single-get or stream multipart images from this view.

Resource base: /urlapi/liveview
Response Mime-Types: **image/jpeg** for snapshots and
multipart/x-mixed-replace for streams
application/json for errors

Parameter	Group	Description	Default Values
User	*Access*	The name of a valid Netavis User	default: 'guest'
Passwd		The MD5 hash coded value of the password	default: MD5('guest')
SessionID	*Access*	Optionally you can use an already opened session (e.g. via the OpenSession command) by supplying this parameter. For security reason, the session which is identifying itself via this parameter has to come from the same IP where it had been opened.	
ChannelID		Use this parameter to start configure or get images from a previously created live view stream	
StreamID		Use this parameter to start configure or get images from a previously created live view stream	
Fps		Framerate at which the view is streamed to the client	default: 5
Snapshot		Use when only a snapshot is needed from the view	default: false
Multipart		Use when a stream of images is needed from the view	default: false
Close		Supply ChannelID and StreamID together with this parameter to close an opened and streaming channel	

Examples

1. Setup a live view of two cameras (id=2 and id=3) in a 3 by 3 grid on a 800 by 600 camvas. The first grid will start at 0,0 and will span one columns and one row, the secons will start at 1,1 and span 2 columns and 2 rows.
   http://192.168.7.221/urlapi/liveview?SessionID=1&Fps=10&c_w800&c_h=600&c_bg=%23888888&g_c=3&g_r=3&g_g=2&g_fg=%23ff0000&vp_0_0_eid=2&vp_0_0_c=1&vp_0_0_r=1&vp_0_0_tt=cam2&vp_0_0_th=15&vp_0_0_tbg=%23000000&vp_0_0_tfg=%23ffffff&vp_0_0_tp=top&vp_1_1_eid=3&vp_1_1_c=2&vp_1_1_r=2&vp_1_1_tt=cam3&vp_1_1_th=15&vp_1_1_tbg=%23000000&vp_1_1_tfg=%23ffffff&vp_1_1_tp=bottom
   Response for the above will contain the channel and stream IDs which must be used for any further request to refer to this view.
2. Requesting for a live stream with the user named 'admin' in a multipart motion JPEG stream from the camera '38' with the encoded password of 'admin' ('21232f297a57a5a894a0e4a801fc3') in 'High' quality 'Medium' size and in '5' Fps:
   http://192.168.7.221/urlapi/live?User=admin&Passwd=21232f297a57a5a894a0e4a801fc3&EntityID=38&Size=Medium&Quality=High&Fps=5&Type=JPEG
3. Request via encrypted credentials:
   http://192.168.7.221/urlapi/live?Login=Xx1D67xZKVYTgA2mWh6TJxBLkD5Jek%2FVrouDkScNiN4%3D&EntityID=38&Size=Medium
4. Request for the same live stream using the earlier given SessionID '1':
   http://192.168.7.221/urlapi/live?SessionID=1&EntityID=38&Size=Medium

3. Archive access

3.1 Getting an archive stream from a recorded video

Resource base: /urlapi/archive
Response Mime-Types: ***image/jpeg*** for images
application/json for errors

Parameter			Description	Default Values
User	*Access*		The name of a valid Netavis User	default: 'guest'
Passwd			The MD5 hash coded value of the password	default: MD5('guest')
SessionID	*Access*		Optionally you can use an already opened session (e.g. via the OpenSession command) by supplying this parameter. For security reason, the session which is identifying itself via this parameter has to come from the same IP where it had been opened.	
ChannelID			Use these parameters to download the images selected by a previous query	
StreamID				
EntityID			Unique ID of the camera	Mandatory
FrameStep			Gap between single images in the stream	default: 0
MaxFrames			Max. number of frames to stream	default: 0
Backward			Backward direction of archive streaming	default: false
Type			Image type to retrieve. One of: JPEG, MPEG4, H264	default: JPEG
Snapshot			Request to deliver one single image for the defined time stamp. Type will be reverted to JPEG!	default: false
Cache			The server will cache the requested number of images and later you can download them from this storage.	Mandatory for requests
Filter	DateTimeFilter	Start	Timestamp in human readable format to start the streaming from	default: none
		End	Timestamp in human readable format to finish streaming at	
		Duration	Duration to streaming until	
	MillisTimeFilter	Start	Timestamp in epoch format to start the streaming from	

Parameter			Description	Default Values
		End	Timestamp in epoch format to finish streaming at	
		Duration	Duration to streaming until	
ImageStamp			Supply ChannelID and StreamID together with this parameter to returns the image corresponding to the given time stamp (in milliseconds)	
Close			Supply ChannelID and StreamID together with this parameter to close an opened and streaming channel	

Explanation of parameters:

If you have read the SNAP programming manual you already know that external applications have to open sessions to Netavis to use its resources, including accessing archives. There are two major modes to achieve it, one is to work in an already opened session and second to open the session via issuing the above request. Normally the first method is the preferred one, because if you are writing an application, then an open session is necessary anyways. In this case you have to supply the known session ID as a parameter to identify the instance in which you are working. After you have issued the above command the answer will contain two more attributes which tell you the channel ID and the steam ID, which is automatically opened in order to deliver the requested images. These two parameters must be supplied at all further requests to access the images belonging to this archive query. Please note that channels are automaticcaly opened for each query and to close them you have to issue your last command with Close=true.

DateTimeFilter or MillisTimeFilter are only parameter values of Filter. Without it the archive stream starts from the last recent available and plays until the first recent available images in forward direction. In background direction it plays from the first to the last recent available images. DateTimeFilter is different from MillisFilter only the format; while DateTimeFilter uses human readable form, the MillisFilter uses epoch format.

To access a stream of archived images you have to use the following control flow:

- issue a command, which defines the time span for a camera and the Cache parameter.
  http://192.168.7.221/urlapi/archive?SessionID=1&EntityID=38&Filter=DateTimeFilter&Start=2013-08-05T07:00:00&Duration=PT10S&Cache=10

  the answer for this command will contain the session, channel, stream IDs of the request and the number of images, number of cached images and header information for all cached images (including their size and time stamp).


```
{"SNAP":{"ExecutionStatus":{
      "ReturnCode":0,
      "SessionID":1,
      "ChannelID":2,
      "StreamID":1,
      "NumberOfImages":10,
      "CachedImages":10},
      "ImageMetaData":{
            "ImageNode":[
                  {"ImageLength":7974,"ImageStamp":1375678800750},
                  {"ImageLength":7994,"ImageStamp":1375678801749},
                  {"ImageLength":8010,"ImageStamp":1375678802748},
                  {"ImageLength":7963,"ImageStamp":1375678803747},
```

```
                        {"ImageLength":7989,"ImageStamp":1375678804746},
                        {"ImageLength":7989,"ImageStamp":1375678805745},
                        {"ImageLength":7982,"ImageStamp":1375678806745},
                        {"ImageLength":7980,"ImageStamp":1375678807745},
                        {"ImageLength":7990,"ImageStamp":1375678808745},
                        {"ImageLength":7993,"ImageStamp":1375678809745}
                ]
        }
}}
```

- now you can access the images sequentially by issuing:
  http://192.168.7.221/urlapi/archive?SessionID=1&ChannelID=1&StreamID=1
- or by stamp by issuing:
  http://192.168.7.221/urlapi/archive?SessionID=1&ChannelID=1&StreamID=1&ImageStamp=1375678800750
- taken that you want to keep your session open but want to close the channel and streamer, issue:
  http://192.168.7.221/urlapi/archive?SessionID=1&ChannelID=1&StreamID=1&Close=true

Every time you issue a command with the Cache parameter will remove the previously cached frames and will return meta information about the currently cached images.

Examples

1. Requesting for a 'Snapshot' image of camera '5' as 'guest' user for the defined time.
   http://192.168.7.221/urlapi/archive?EntityID=5&Snapshot=true&Filter=DateTimeFilter&Start= 2013-08-05T07:00:00

   Note: Without user and passwd parameters the default (guest/guest) will be used.

   A positive snapshot sample response:

   ```
   HTTP/1.0 200 OK\r\n
   Content-Type: image/jpeg\r\n
   Content-Length: <image size>\r\n\r\n
   <JPEG image data>\r\n
   ```

   Negative response sample as “application/json” mime-type:

   ```
   {"SNAP": {"ExecutionStatus": {
       "ErrorText": "Illegal user/passwd. User=admin1",
       "ReturnCode": 103}}}
   ```

3.2 Getting a timed archive stream from a recorded video

Resource base: /urlapi/timedarchive
Response Mime-Types: ***image/jpeg*** for images
application/json for errors

Parameter			Description	Default Values
User		*Access*	The name of a valid Netavis User	default: 'guest'
Passwd			The MD5 hash coded value of the password	default: MD5('guest')
SessionID		*Access*	Optionally you can use an already opened session (e.g. via the OpenSession command) by supplying this parameter. For security reason, the session which is identifying itself via this parameter has to come from the same IP where it had been opened.	
ChannelID			Use this parameter to refer to the previously created query	
StreamID			Use this parameter to refer to the previously created query	
EntityID			Unique ID of the camera	Mandatory
Snapshot			Request to deliver one single image	default: false
Cmd	Start		Start the playback of the archive	
	Stop		Stop the playback and close the channel	
	Pause		Pause the playback	
	Resume		Resume the playback	
	GetNext		Returns the next frame in paused state	
	GetStamp		Returns the time stamp of the current frame	
	SetSpeed		Sets the playback speed. The value is defined by the Speed parameter. Floating point value, when > 1 quick playback, < 1 slow motion playback.	
Mode	Single		Retrieves the next frame, then exists	
	Multipart		Retrieves all frames in a multipart manner	

Parameter			Description	Default Values
Filter	DateTimeFilter	Start	Timestamp in human readable format to start the streaming from	default: none
		End	Timestamp in human readable format to finish streaming at	
		Duration	Duration to streaming until	
	MillisTimeFilter	Start	Timestamp in epoch format to start the streaming from	
		End	Timestamp in epoch format to finish streaming at	
		Duration	Duration to streaming until	

Explanation of parameters:

Usage of the session setup and time filter parameters are the same as for the normal archive access. Th eonly difference it that the frames are timed exactly as you have specified via the fps and speed parameters.

To access a stream of archived images you have to use the following control flow:

- issue a command, which defines the time span for a camera.
  http://192.168.7.221/urlapi/timedarchive?SessionID=1&EntityID=38&Filter=DateTimeFilter&Start=2013-08-05T07:00:00&Duration=PT10S

  the answer for this command will contain the session, channel, stream IDs of the request.

- now you can access the images and control the stream by issuing:
  http://192.168.7.221/urlapi/ timedarchive?SessionID=1&ChannelID=1&StreamID=1&Mode=Multipart&Cmd=Start
  http://192.168.7.221/urlapi/ timedarchive?SessionID=1&ChannelID=1&StreamID=1&Cmd=Pause
  http://192.168.7.221/urlapi/ timedarchive?SessionID=1&ChannelID=1&StreamID=1&Cmd=GetNext
  http://192.168.7.221/urlapi/ timedarchive?SessionID=1&ChannelID=1&StreamID=1&Cmd=GetStamp
  http://192.168.7.221/urlapi/ timedarchive?SessionID=1&ChannelID=1&StreamID=1&Cmd=Resume
  http://192.168.7.221/urlapi/ timedarchive?SessionID=1&ChannelID=1&StreamID=1&Cmd=SetSpeed&Speed=2.0

4. Receiving events

4.1 Getting a live or an archive event stream

Resource base: /urlapi/event

Response Mime-Types: **multipart/x-mixed-replace** with **application/json** contents or simple **application/json** for errors

Parameter			Description	Default Values
User		*Access*	The name of a valid Netavis User	default: 'guest'
Passwd			The MD5 hash coded value of the password	default: MD5('guest')
SessionID		*Access*	Optionally you can use an already opened session (e.g. via the OpenSession command) by supplying this parameter. For security reason, the session which is identifying itself via this parameter has to come from the same IP where it had been opened.	
Mode			One of {Live, Archive, ArchiveAndLive}	default: Live if Snapshot is false, Archive if Snapshot is true
EventType			Event type to filter, or none for all	default: none
MaxEvents			The number of max events to retrieve, 0 means unlimited	default: 50 if Mode is not “Live”, zero for all other Mode cases
PropertyFilterNameX		Property Filter	The name of the custom event property to filter on	default: none
PropertyFilterValueX			The value of the custom event property to filter on	
Filter	DateTimeFilter	Start	Timestamp in human readable format to start the streaming from	default: none
		End	Timestamp in human readable format to finish streaming at	
		Duration	Duration to streaming until	
	MillisTimeFilter	Start	Timestamp in epoch format to start the streaming from	
		End	Timestamp in epoch format to finish	

Parameter			Description	Default Values
			streaming at	
		Duration	Duration to streaming until	
Backward			Backward direction on an archive transmission	default: false
Snapshot			Use when you want to retrieve the last 50 available events	default: false
SendEventsSeparately			True if the stream is manually controlled	default: false

Explanation of parameters:

EventType: The list of valid events and their actual parameters in Netavis may change from version to version. This information can be found and downloaded from each Netavis server via the web interface. Start an internet browser and enter the IP address of your Netavis host. Click on the **Customizer login** and then to the **Download configuration files** link. On the next page lookup and download the EMSAPIdoc.zip. After un-compressing the archive open the index.html file with your browser. All filterable events can be looked at in details after selecting the **server.event_manager.event** package.

The PropertyFilter parameter handling was designed to only search for custom events and events of a specific CameraID. All other events are not affected and therefore the PropertyName will not match. For example use the following link to filter by CameraID: http://IP-ADDRESS/urlapi/event?SessionID=XXXXXX&Mode=Archive&EventType=MotionDetection&MaxEvents=50&PropertyFilterName0=CameraID&PropertyFilterValue0=X

PropertyFilterName should be followed by an index from 0 to define a PropertyFilter together with the same indexed PropertyFilterValue.

PropertyFilter is not a parameter it symbolizes only the usage of PropertyFilterName & PropertyFilterValue together.

Filter is an optional parameter and not a symbol. DateTimeFilter or MillisTimeFilter are only parameter values of TimeFilter. Without it the archive stream starts from the last recent available event and sends until the first recent available event in forward direction. In background direction it sends from the first to the last recent available event. DateTimeFilter differs from MillisTimeFilter only in the format; while DateTimeFilter uses human readable form, the MillistimeFilter uses epoch format.

Backward is available only in Archive event stream.

Only the Access and MaxEvents parameters are used if Snapshot has been defined as true.

SendEventsSeparately is a secure way of receiving all events of a stream with server side care of those events what arrives in time when the client is not listening on the channel. The server replies to SendEventsSeparately a StreamID and a ChannelID belongs to the newly created event stream what needs to be used for ReadNextEvent call via a new control query. See the documentation of the ReadNextEvent command in the next paragraph, too.

Examples

1. Request for the last 50 live events:
   http://192.168.7.221/urlapi/event?SessionID=1&Snapshot=true
2. A positive multipart HTTP sample response:

```
HTTP/1.0 200 OK\r\n
```

```
Content-Type: multipart/x-mixed-replace;boundary=--boundary\r\n
\r\n
--boundary\r\n
Content-Type: application/json\r\n
Content-Length: <json event size>\r\n\r\n
<json event data>\r\n
--boundary\r\n
...
```

3. Negative “application/json” mime-type sample response :

```
{"SNAP": {"ExecutionStatus": {
    "ErrorText": "Illegal user/passwd. User=admin1",
    "ReturnCode": 103
}}}
```

4. A positive response to SendEventsSeparately=true kind event setup:

```
{"SNAP":{"EventStream":{"SessionID":4823,"LiveStream":{"StreamID":1,"MaxEve
nts":0},"ChannelID":1}}}
```

5. Control commands

Resource base: /urlapi/control

Response Mime-Types: ***application/json***

Every Control request has a mandatory parameter named **Command** in the first position of the URL parameters. The different commands may have further mandatory or optional parameters what are described in the following. In the description of the next URL commands we hide the mandatory parameter COMMAND and list only the other parameters. Other words, every Control query URL Commands must start with the URL Base right after followed by the parameter Command=<COMMAND>. Please have a look at the examples.

5.1 GetServerTime

Used for getting the current time of the server in milliseconds, UNIX epoch time format. This command has no other parameters.

Example

- Sample GetServerTime command request:
  http://192.168.7.221/urlapi/control?Command=GetServerTime
- Positive response to an OpenSession command:
  ```
  {"SNAP": {"ServerTime": 1324481877510}}
  ```
- Negative response to an OpenSession command:
  ```
  {"SNAP": {"ExecutionStatus": {"ErrorText": "No Command:
  'GetServerTimestmap' matched"}}}
  ```

5.2 OpenSession

Used for opening a communication channel between client and server. The channel can be used to send further requests to or retrieve data from the server. Every session has its own unique SessionID produced by the server in the reply to the OpenSession request command. A session belongs to its user with the IP address of the remote client machine where it was opened. Other clients from different IP address can not use session opened from other hosts. An opened session can handle requests of live or archive streams. Because this session is a SNAP session it is also important to handle an open session by the SNAP requirements E.g. periodically sending LiveSignal requests to keep it open if no channel was open on it. At the end of the communication, it is highly recommended to close the session, too.

Parameter		Description	Default Values
User	*Access*	The name of a valid Netavis user	default: 'guest'
Passwd		The MD5 hash coded value of the password	default: MD5('guest')

Example

- Sample OpenSession command request:
  http://192.168.7.221/urlapi/control?Command=OpenSession&User=admin&Passwd=21232f297a57a5a743894a0e4a801fc3
- Positive response to an OpenSession command:
  ```
  {"SNAP": {"NewSession": { "UID": 1, "SessionID": 4332 }}}
  ```
- Negative response to an OpenSession command:
  ```
  {"SNAP": {"ExecutionStatus": {
  "ErrorText": "Session not found. SessionID=4325",
  "ReturnCode": 200
  }}}
  ```

5.3 CloseSession

Used for closing an earlier opened session. It also closes all related channels, live or archive streams of the session. It also frees all resources on the server side.

Parameter	Description	Default Values
SessionID	The session identifier given from OpenSession	Mandatory

Example

- Sample request for closing an open session:
  http://192.168.7.221/urlapi/control?Command=CloseSession&SessionID=4325
- Response of the CloseSession command:
  ```
  {"SNAP": {"ExecutionStatus": {"ReturnCode": 0}}}
  ```
- Negative response to CloseSession command:
  ```
  {"SNAP": {"ExecutionStatus": {
  "ErrorText": "Session not found. SessionID=4325",
  "ReturnCode": 200
  }}}
  ```

5.4 OpenChannel

Parameter	Description	Default Values
SessionID	The session identifier given from OpenSession	Mandatory
TimeLimit	Timeout after which the server terminates a ReadChannel request in any case (seconds).	default: 0
DataLimit	Size of transferred data in the channel after which the server terminates a ReadChannel request in any case (KBytes). The value 1 cause termination of ReadChannel request frame by frame.	default: 0
FrameBased	Streaming without GOP	default: false

Example

- Sample request for getting a new channel:
  http://192.168.7.221/urlapi/control?Command=OpenChannel&SessionID=4325
- Positive response to the OpenChannel command:
  ```
  {"SNAP": {"NewChannel": {"ChannelID": 1}}}
  ```
- Negative response to OpenChannel command:

  ```
  {"SNAP": {"ExecutionStatus": {
  "ErrorText": "Session not found. SessionID=4325",
  "ReturnCode": 200
  }}}
  ```

5.5 ReadChannel

Parameter	Description	Default Values
SessionID	The session identifier given from OpenSession	Mandatory
ChannelID	Channel identifier given from OpenChannel	Mandatory
ForceSingleFrame	True if only one frame is needed	default: true

Note: for getting any content via ReadChannel you have to setup the Channel before starting to read. E.g. you have to subscribe to retrieve event notifications of an event type. Please read the related SNAP ReadChannel documentation for further information.

Example

- Sample request for reading the next frame from an open channel:
  http://192.168.7.221/urlapi/control?Command=ReadChannel&SessionID=4325&ChannelID=1&ForceSingleFrame=true
- Positive response to the ReadChannel command:
- Negative response to ReadChannel command:

```
{"SNAP": {"ExecutionStatus": {
"ErrorText": "Session not found. SessionID=4325",
"ReturnCode": 200
}}}
```

5.6 CloseChannel

Parameter	Description	Default Values
SessionID	The session identifier given from OpenSession	Mandatory
ChannelID	Channel identifier given from OpenChannel	Mandatory

Example

- Sample request for closing an open channel:
  http://192.168.7.221/urlapi/control?Command=CloseChannel&SessionID=4325&ChannelID=1
- Positive response to the CloseChannel command:
  {"SNAP": {"ExecutionStatus": {"ReturnCode": 0}}}
- Negative response to CloseChannel command:

```
{"SNAP": {"ExecutionStatus": {
"ErrorText": "Session not found. SessionID=4325",
"ReturnCode": 200
}}}
```

5.7 GetEntityTree

Used for retrieving available entities and its hierarchy from the server. Optionally the root point of the tree can define via the RootEntityID, without it the complete entity tree are going to be send.

Parameter	Description	Default Values
SessionID	The session identifier given from OpenSession	Mandatory
RootEntityID	The ID of the root entity to start the tree from	default: 0
RequestedDetails	Comma separated list of detail descriptors. One or list of: IP_ADDRESS, STATUS_CODES, PTZ_DETAILS	optional
Sort	if set to “true” sort the tree by entity name	optional, default: false

Example

- Sample request for a GetEntityTree command:
  http://192.168.7.221/urlapi/control?Command=GetEntityTree&SessionID=4325&RequestedDetails=PTZ_DETAILS
- Positive Response to GetEntityTree command:

```
{"SNAP": {"EntityTree": {"EntityTreeNode": [
  {
  "Host": {
  "HostID": 13792383803876560,
  "Address": "127.0.0.1",
  "Connected": true,
  "Name": "netavis"
  },
  "ParentEntityID": 0,
  "Type": "Group",
  "Name": "cust1 Cameras",
  "EntityID": 1
  }
]}}}
```

- Negative response to GetEntityTree command:

```
{"SNAP": {"ExecutionStatus": {
"ErrorText": "Session not found. SessionID=4325",
"ReturnCode": 200 }}}
```

5.8 GetIODeviceList

Used for retrieving list of I/O devices.

Parameter	Description	Default Values
SessionID	The session identifier given from OpenSession	Mandatory

Example

- Sample request for a GetIODeviceList command:
  http://192.168.7.221/urlapi/control?Command=GetIODeviceList&SessionID=4325
- Positive Response to GetIODeviceList command:

```
{"SNAP":{"IODeviceList":{
  "IODevice":{
  "StatusText":"",
  "DeviceID":1000,
  "IOPort":[
  {"PortValue":1,"PortID":1,"PortType":"DIGITAL_INPUT"},
  {"PortValue":1,"PortID":2,"PortType":"DIGITAL_OUTPUT"}],
  "StatusID":1,
  "DeviceName":"Canon cam"
}}}}
```

- Negative response to GetIODeviceList command:

```
{"SNAP": {"ExecutionStatus": {
"ErrorText": "Session not found. SessionID=4325",
"ReturnCode": 200 }}}
```

5.9 GetVideoArchiveMap

Get *recording-status-map* of the video archive of a camera. The *recording-status-map* contains information about recording status at the timeline specified by the request. The timeline is split into several time periods. Each period is represented by an element <MapItem> in the response to this request. Thus, the response is a list of <*MapItem*>s. Each <MapItem> covers a part of the requested timeline. The start of time period covered by <*VideoArchiveMap*> is described by attribute *StartDateTime.* The <*VideoArchiveMap*> contains a *Unit* attribute, which describes the duration of one *time-unit* within the <MapItem>. The *time-unit* can be either “Day” or “Minute”. If the *time-unit* is “Day” then the timeline is split into <MapItems> having duration of 1 month (28-31 *time-units*). If the *time-unit* is “Minute” then the timeline is split into <MapItems> having duration of 1 hour (60 *time-units*). Each <*MapItem*> contains attribute *FrameCount* which contains the total number of frames recorded in the time period covered by the item. Each <*MapItem*> contains a sequence of hexadecimal coded character (0-F). Each character represents one *time-unit*. The information coded in one time-unit is the following:

- Bit0 is set: there are recorded frames within time unit
- Bit1 is set: there are event-triggered recordings within time unit
- Bit2 is set: frames within time unit are protected against removal
- Bit3: reserved for future use

Values should be extracted using bit operations.

Parameter		Description		Default Values
SessionID		The session identifier given from OpenSession		Mandatory
CameraID		The camera identifier		Mandatory
Unit		One of: Minute, Day		default: Minute
Filter	DateTimeFilter	Start	Timestamp in human readable format to start the streaming from	
		End	Timestamp in human readable format to finish streaming at	
		Duration	Duration to streaming until	
	MillisFilter	Start	Timestamp in epoch format to start the streaming from	
		End	Timestamp in epoch format to finish streaming at	
		Duration	Duration to streaming until	

Explanation of parameters:

DateTimeFilter or MillisFilter are the only parameter values of TimeFilter. Without it, the archive stream starts from the last recent available and plays until the first recent available images in forward direction. In backward direction it plays from the first to the last recent available images. DateTimeFilter differs from MillisFilter only in the format; DateTimeFilter uses human readable form, the MillisFilter uses epoch format. The returned range (including the start and end time) is always rounded to a full day.

Example

- Sample request for getting an archive video map:
  http://192.168.7.221/urlapi/control?Command=GetVideoArchiveMap&SessionID=4325&CameraID=38

- Positive response to the GetVideoArchiveMap command:

```
{"SNAP": {"VideoArchiveMap": {
"Unit": "Minute",
"StartDateTime": "2011-10-07T00:00:00.000+02:00",
"EntityID": 38,
"MapItem": [ { "FrameCount": 1800, "content":
"00000000000000000000000000000011111111111111111111111111111111" }]}}}
```

- Negative response to GetVideoArchiveMap command:

```
{"SNAP": {"ExecutionStatus": {
"ErrorText": "Session not found. SessionID=4325",
"ReturnCode": 200
}}}
```

5.10 GetUserList

Parameter	Description	Default Values
SessionID	The session identifier given from OpenSession	Mandatory
UserD	Unique user identifier	Mandatory Non-zero value filters one single user, zero value returns all users

Example

- Sample request for getting list of users
  http://192.168.7.221/urlapi/control?Command=GetUserList&SessionID=4255&UserID=0
- Positive response to the GetUserList command:

```
{"SNAP": {"UserList": {"User": [
  {
    "UserID": 1,
    "Name": "admin",
    "LoginName”: "Admin User”,
    "GroupID”: 1,
    "CustomerID”: 1,
   ….
  },
  {
    "UserID": 2,
    "Name": "Guest User"
   ….
  }
]}}}
```

- Negative response to GetUserList command:

```
{"SNAP": {"ExecutionStatus": {
"ErrorText": "Session not found. SessionID=4325",
"ReturnCode": 200
}}}
```

5.11 PropagateEvent

Parameter	Description	Default Values
SessionID	The session identifier given from OpenSession	Mandatory
EventType	Type of the Event	Mandatory
EventStamp		
EventPropertyName[x]	Name of the [x] event property	
EventPropertyValue[x]	Value of the [x] event property	
EventPropertyType[x]	Type of the [x] event property	

Example

- Sample request for propagating a new event:
  http://192.168.7.221/urlapi/control?Command=PropagateEvent&SessionID=4325&EventType=Motion Detection&EventPropertyName0=CameraID&EventPropertyType0=Integer&EventPropertyValue0=4
- Positive response to the PropagateEvent command:
  ```
  {"SNAP": {"ExecutionStatus": {"ReturnCode": 0}}}
  ```
- Negative response to PropagateEvent command:

  ```
  {"SNAP": {"ExecutionStatus": {
  "ErrorText": "Session not found. SessionID=4325",
  "ReturnCode": 200
  }}}
  ```

5.12 PerformAction

Parameter	Description	Default Values
SessionID	The session identifier given from OpenSession	Mandatory
ActionType	Type of the Event	Mandatory
ActionPropertyName[x]	Name of the [x] action property	
ActionPropertyValue[x]	Value of the [x] action property	
ActionPropertyType[x]	Type of the [x] action property	

Example

- Sample request for performing an action on the server:
  http://192.168.7.221/urlapi/control?Command=PerformAction&ActionType=SetCameraPTZPosition&SessionID=4325&ActionPropertyName0=CameraID&ActionPropertyType0=Integer&ActionPropertyValue0=1&
  ActionPropertyName1=PositionName&ActionPropertyType1=String&ActionPropertyValue1=Door
- Positive response to the PerformAction command:
  ```
  {"SNAP": {"ExecutionStatus": {"ReturnCode": 0}}}
  ```
- Negative response to PerformAction command:

  ```
  {"SNAP": {"ExecutionStatus": {
  "ErrorText": "Session not found. SessionID=4325",
  "ReturnCode": 200
  }}}
  ```

5.13 LiveSignal

Parameter	Description	Default Values
SessionID	The session identifier given from OpenSession	default: 0

Example

- Sample request for sending a live signal:
  http://192.168.7.221/urlapi/control?Command=LiveSignal&SessionID=4325
- Positive response to the LiveSignal command:
  ```
  {"SNAP": {"ExecutionStatus": {"ReturnCode": 0}}}
  ```
- Negative response to LiveSignal command:

  ```
  {"SNAP": {"ExecutionStatus": {
  "ErrorText": "Session not found. SessionID=4325",
  "ReturnCode": 200
  }}}
  ```

5.14 VideoStreamAnnotation

Parameter	Description	Default Values
SessionID	The session identifier given from OpenSession	Mandatory
CameraID	ID of a camera or camera group	Mandatory
Text	Text to display on camera view	
Action	One of: Start, Stop	default: Start
Destination	One of: Live, Archive, LiveAndArchive	default: Live
StartTimestamp	Epoch or human readable timestamp format of annotation starting	default: <now>
StopTimestamp	Epoch or human readable timestamp format of annotation stopping	
StopAfterMillisec	Millisec value of stoppping after start	default: 10 sesc
EventStorage	One of: UnSave, SaveStart, SaveStop, SaveAll	default: SaveStart
EventVisibility	One of: Hide, ShowStart, ShowStop, ShowAll	default:Hide
ForegroundColor		
BackgroundColor		
FontName		
FontSize		
Alignment		
Margin		
Wrapping		
StopOnClick		
BlinkDuration		

Note: The parameters from ForegroundColor to BlinkDuration are not yet supported on server-side

Example

- Sample request for sending a live VideoStreamAnnotation:
  http://192.168.7.221/urlapi/control?Command=VideoStreamAnnotation&SessionID=4325&CameraID=1&Text=AnnotationMessage
  Note: Do not forget to encode the url before send it like this:
  http%3A%2F%2F192.168.7.221%2Fmobile%2Fcontrol%3FCommand%3DVideoStreamAnnotation%26SessionID%3D42234%26CameraID%3D1%26Text%3DAnnotation+message

- Positive response to the VideoStreamAnnotation command:
  ```
  {"SNAP": {"ExecutionStatus": {"ReturnCode": 0}}}
  ```

- Negative response to VideoStreamAnnotation command:

  ```
  {"SNAP": {"ExecutionStatus": {
  "ErrorText": "Session not found. SessionID=4325",
  "ReturnCode": 200
  }}}
  ```

5.15 ReadNextEvent

Parameter	Description	Default Values
SessionID	The session identifier given from OpenSession	Mandatory
ChannelID	Channel identifier given from event query setup	Mandatory

Note: For using ReadNextEvent, the event stream must be setup first via an event query with a SendEventsSeparately=true parameter. Please read the documentation for the event query command.

Example

- Sample request for reading the next event from an already opened channel:
  http://192.168.7.221/urlapi/control?Command=ReadNextEvent&SessionID=4325&ChannelID=1
- Positive response ReadNextEvent command:


```
{"SNAP": {"Event": {
  "EventType": "MotionDetection",
  "EventID": 89009793,
  "DateTimeStamp": "2012-01-05T14:03:35.136+01:00",
  "MillisStamp": 1325768615136,
  "EventProperty": [
    {
      "Value": "",
      "Type": "String",
      "Name": "EventText"
    },
    {
      "Value": "Motion event forgalom on camera cust1:4.120 ().",
      "Type": "String",
      "Name": "EventText"
    },
    {
      "Value": 7,
      "Type": "Integer",
      "Name": "ClassID"
    },
    {
      "Value": 38,
      "Type": "Integer",
      "Name": "CameraID"
    },
    {
      "Value": "cust1:4.120",
      "Type": "String",
      "Name": "CameraName"
    },
    {
      "Value": "forgalom",
      "Type": "String",
      "Name": "EventName"
    },
    {
      "Value": "",
      "Type": "String",
      "Name": "EventComment"
```

```
      },
      {
        "Value": 5564921012190495,
        "Type": "Long",
        "Name": "MDParams"
      },
      {
        "Value": 100,
        "Type": "Integer",
        "Name": "Priority"
      },
      {
        "Value": 1325768615136,
        "Type": "Long",
        "Name": "EventStamp"
      },
...
    ]
}}}
```

- Negative response to ReadNextEvent command:

```
{"SNAP": {"ExecutionStatus": {
"ErrorText": "Session not found. SessionID=4325",
"ReturnCode": 200
}}}
```

5.16 RegisterCustomEvent

Parameter	Description	Default Values
SessionID	The session identifier given from OpenSession	Mandatory
EventType	Name of the custom event	Mandatory
EventText	Log text of the event (appears in the Observer event list). It may contain placeholders for event parameters which will be replaced by the actual value of the parameter. The format of the placeholder is %ParamN%, where N is a number between 1 and 5	Mandatory
EventPropertyName[x]	Name of the [x] event property	Mandatory

Example

- Sample request for registering a new custom event:
  http://192.168.7.221/urlapi/control?Command=RegisterCustomEvent&SessionID=4325&EventType=Square&EventText=”Square event”&EventPropertyName0=X&EventPropertyName1=Y
- Positive response to the RegisterCustomEvent command:
  ```
  {"SNAP": {"ExecutionStatus": {"ReturnCode": 0}}}
  ```
- Negative response to RegisterCustomEvent command:

  ```
  {"SNAP": {"ExecutionStatus": {
  "ErrorText": "Session not found. SessionID=4325",
  "ReturnCode": 200
  }}}
  ```

5.17 UnregisterCustomEvent

Parameter	Description	Default Values
SessionID	The session identifier given from OpenSession	Mandatory
EventType	Name of the custom event	Mandatory

Example

- Sample request for unregistering a custom event:
  http://192.168.7.221/urlapi/control?Command=UnregisterCustomEvent&SessionID=4325&EventType=Square
- Positive response to the UnregisterCustomEvent command:
  ```
  {"SNAP": {"ExecutionStatus": {"ReturnCode": 0}}}
  ```
- Negative response to UnregisterCustomEvent command:

  ```
  {"SNAP": {"ExecutionStatus": {
  "ErrorText": "Session not found. SessionID=4325",
  "ReturnCode": 200
  }}}
  ```

5.18 PropagateCustomEvent

Parameter	Description	Default Values
SessionID	The session identifier given from OpenSession	Mandatory
EventType	Name of the custom event	Mandatory
EventPropertyName[x]	Name of the [x] event property	
EventPropertyValue[x]	Value of the [x] event property	
EventPropertyType[x]	Type of the [x] event property	

Example

- Sample request for propagating a new custom event:
  http://192.168.7.221/urlapi/control?Command=PropagateCutomEvent&SessionID=4325&EventType=Square&EventPropertyName0=X&EventPropertyType0=Integer&EventPropertyValue0=4&EventPropertyName1=Y&EventPropertyType1=Integer&EventPropertyValue1=4
- Positive response to the PropagateCustomEvent command:
  ```
  {"SNAP": {"ExecutionStatus": {"ReturnCode": 0}}}
  ```
- Negative response to PropagateCustomEvent command:

  ```
  {"SNAP": {"ExecutionStatus": {
  "ErrorText": "Session not found. SessionID=4325",
  "ReturnCode": 200
  }}}
  ```

5.19 SetArchiveProtection

Parameter	Description	Default Values
SessionID	The session identifier given from OpenSession	Mandatory
EntityID	Unique camera identifier	Mandatory
Set	Protect or unprotect the selected interval	Mandatory, one of: { true, false }
From	Timestamp in format of yyyy-MM-ddTHH:mm:ss.SSS	Mandatory
To	Timestamp in format of yyyy-MM-ddTHH:mm:ss.SSS	Mandatory

Example

- Sample request for protecting archive records of camera #1 from noon to one o'clock:
  http://192.168.7.221/urlapi/control?Command=SetArchiveProtection&SessionID=4255&EntityID=1&Set=true&From=2011-12-14T12:00:00.000&To=2011-12-14T13:00:00.000
- Positive response to the SetArchiveProtection command:
  ```
  {"SNAP": {"ExecutionStatus": {"ReturnCode": 0}}}
  ```
- Negative response to SetArchiveProtection command:

  ```
  {"SNAP": {"ExecutionStatus": {
  "ErrorText": "Session not found. SessionID=4325",
  "ReturnCode": 200
  }}}
  ```

5.20 ShowArchiveForCamera

Parameter		Description		Default Values
SessionID		The session identifier given from OpenSession		Mandatory
EntityID		Unique camera identifier		Mandatory
TargetIP		IP address of the client on which the action will be executed		Mandatory
TargetUser		The action will be executed on all clients where the givel user is logged in		Mandatory
Filter	DateTimeFilter	Start	Start of archive in human readable form	Mandatory
		End	End timestamp in human readable form	
		Duration	Duration	
	MillisFilter	Start	Start of archive in epoch format	
		End	End timestamp in epoch format	
		Duration	Duration	

Example

- Sample request to start playing back the archive records of camera #1 from noon to one o'clock:
  http://192.168.7.221/urlapi/control?Command=ShowArchiveForCamera&SessionID=4255&EntityID=1&TargetIP=&TargetUser=admin&Filter=DateTimeFilter&Start=2011-10-18T12:00:00.000&End=2011-10-18T13:00:00.000

5.21 GetUserLayouts

Parameter	Description	Default Values
SessionID	The session identifier given from OpenSession	Mandatory
UserID	User identifier	Mandatory

Example

- Sample request for getting all layouts of user #1:
  http://192.168.7.221/urlapi/control?Command=GetUserLayout&SessionID=4255&UserID=1
- Positive response to the command:


```
{"SNAP":{"UserLayouts":{"Windows":{"Window":[{"ViewGroups":{"ViewGroup":{"V
iewGroupElements":{"ViewGroupElement":[{"Duration":10,"ViewGroupID":3,"Elem
entID":1,"PanelNumber":1},{"Duration":10,"ViewGroupID":3,"ElementID":2,"Pan
elNumber":2}],"count":2},"ViewGroupName":"nana","ViewGroupID":3,"UserID":1,
"WindowID":3},"count":1},"WindowName":"Window
3","WindowID":3,"Views":{"View":{"UserID":1,"Flags":1,"PanelNr":1,"WindowID
":3,"ViewPorts":{"ViewPort":{"PortEntities":{"PortEntity":[{"PortNr":1,"Fla
gs":4101,"EntityNr":0,"Cameras":{"Camera":{"Name":"cust1
Cam1","EntityID":3},"count":1},"Info":"crp1=-1,crp0=-
1,isize=18","EntityID":3},{"PortNr":1,"Flags":4100,"EntityNr":1,"Cameras":{
"Camera":{"Name":7.92,"EntityID":37},"count":1},"Info":"crp1=-1,crp0=-
1,isize=21","EntityID":37}],"count":2},"PortNr":1,"Flags":6,"PanelNr":1,"In
fo":"zm-
aids,prefType=2,crop=0,strech=1"},"count":1},"PanelType":{"AspectRatio":"Fi
ll","ViewIndexName":"Matrix-
1S","ViewIndex":13},"Info":"car=0.0","PanelName":1},"count":1}},
          ...
```


- Negative response to the command:


```
{"SNAP": {"ExecutionStatus": {
"ErrorText": "Session not found. SessionID=4325",
"ReturnCode": 200
}}}
```

5.22 GetSmartGuardsOfWindow

Parameter	Description	Default Values
SessionID	The session identifier given from OpenSession	Mandatory
UserID	User identifier for which the guards are requested	Mandatory
WindowID	Identifier of the window for which the guards are requested	Mandatory

Example

- Sample request for getting the guard list for user 1, window 1
  http://192.168.7.221/urlapi/control?Command=GetSmartGuardsOfWindow&SessionID=4255&UserID=1&WindowID=1

- Positive response to the command:
  {"SNAP":{"ViewGroups":{"ViewGroup":{"ViewGroupElement":[{"Duration":10,"ViewGroupID":1,"ElementID":1,"PanelNumber":99996},{"Duration":10,"ViewGroupID":1,"ElementID":2,"PanelNumber":5},{"Duration":10,"ViewGroupID":1,"ElementID":3,"PanelNumber":4},{"Duration":10,"ViewGroupID":1,"ElementID":4,"PanelNumber":3},{"Duration":10,"ViewGroupID":1,"ElementID":5,"PanelNumber":2},{"Duration":10,"ViewGroupID":1,"ElementID":6,"PanelNumber":1}],"ViewGroupName":"gray","ViewGroupID":1,"UserID":1,"WindowID":1}}}}

- Negative response to the command:

{"SNAP": {"ExecutionStatus": {
"ErrorText": "Session not found. SessionID=4325",
"ReturnCode": 200
}}}

5.23 GetPTZPositionList

Parameter	Description	Default Values
SessionID	The session identifier given from OpenSession	Mandatory
EntityID	Unique camera identifier	Mandatory

Example

- Sample request for getting a list of ptz positions of camera #38.
  http://192.168.7.221/urlapi/control?Command=GetPTZPositionList&SessionID=4255&EntityID=38
- Positive response to the GetPTZPositionList command:

```
{"SNAP": {"PTZPositionList": {"PTZPosition": [
  {
    "PresetPosID": 1,
    "Zoom": 7201,
    "Name": "kapu",
    "Tilt": -1246,
    "Pan": -1979,
    "EntityID": 37,
    "Comment": ""
  },
  {
    "PresetPosID": 2,
    "Zoom": 7201,
    "Name": "ház",
    "Tilt": -1011,
    "Pan": -2243,
    "EntityID": 37,
    "Comment": ""
  },
  {
    "PresetPosID": 3,
    "Zoom": 7201,
    "Name": "épület",
    "Tilt": -1006,
    "Pan": -2537,
    "EntityID": 37,
    "Comment": ""
  }
]}}}
```

- Negative response to GetPTZPositionList command:

```
{"SNAP": {"ExecutionStatus": {
"ErrorText": "Session not found. SessionID=4325",
"ReturnCode": 200
}}}
```

5.24 GetPTZRoutesList

Parameter	Description	Default Values
SessionID	The session identifier given from OpenSession	Mandatory
EntityID	Unique user identifier	Mandatory

Example

- Sample request for getting a list of available routes on camera #38
  http://192.168.7.221/urlapi/control?Command=GetPTZRoutesList&SessionID=4255&EntityID=38
- Positive response to the command:

```
{"SNAP": {"PTZRoutesList": {"PTZRoutes": {
  "RouteID": 1,
  "Name": "környék",
  "EntityID": 37,
  "Comment": ""
}}}}
```

- Negative response to the command:

```
{"SNAP": {"ExecutionStatus": {
"ErrorText": "Session not found. SessionID=4325",
"ReturnCode": 200
}}}
```

5.25 GetPTZRouteDefinition

Parameter	Description	Default Values
SessionID	The session identifier given from OpenSession	Mandatory
EntityID	Unique user identifier	Mandatory
RouteID	Unique route identifier	Mandatory

Example

- Sample request for getting the route definition #1 on camera #38
  http://192.168.7.221/urlapi/control?Command=GetPTZRouteDefinition&SessionID=4255&EntityID=38&RouteID=1
- Positive response to the command:

```
{"SNAP": {"PTZRouteDefinition": {"PTZRoute": [
  {
    "PresetPosID": 1,
    "Sequence": 0,
    "RouteID": 1,
    "StayOnTargetTime": 10000,
    "MoveToTargetTime": 0,
    "EntityID": 37
  },
  {
    "PresetPosID": 2,
    "Sequence": 1,
    "RouteID": 1,
    "StayOnTargetTime": 10000,
    "MoveToTargetTime": 0,
    "EntityID": 37
  }
]}}}
```

- Negative response to the command:

```
{"SNAP": {"ExecutionStatus": {
"ErrorText": "Session not found. SessionID=4325",
"ReturnCode": 200
}}}
```

5.26 LockPTZResource

Parameter	Description	Default Values
SessionID	The session identifier given from OpenSession	Mandatory
EntityID	Unique identifier of the camera	Mandatory

Example

- Sample request for locking PTZ resource camera #38
  http://192.168.7.221/urlapi/control?Command=LockPTZResource&SessionID=4255&EntityID=38
- Positive response to the command:
  ```
  {"SNAP": {"ExecutionStatus": {"ReturnCode": 0}}}
  ```
- Negative response to the command:

  ```
  {"SNAP": {"ExecutionStatus": {
  "ErrorText": "Session not found. SessionID=4325",
  "ReturnCode": 200
  }}}
  ```

5.27 RefreshPTZResource

Parameter	Description	Default Values
SessionID	The session identifier given from OpenSession	Mandatory
EntityID	Unique identifier of the camera	Mandatory

Example

- Sample request for refreshing the lock on the PTZ resource of the camera 38
  http://192.168.7.221/urlapi/control?Command=RefreshPTZResource&SessionID=4255&EntityID=38
- Positive response to the command:
  ```
  {"SNAP": {"ExecutionStatus": {"ReturnCode": 0}}}
  ```
- Negative response to the command:

  ```
  {"SNAP": {"ExecutionStatus": {
  "ErrorText": "Session not found. SessionID=4325",
  "ReturnCode": 200
  }}}
  ```

5.28 ReleasePTZResource

Parameter	Description	Default Values
SessionID	The session identifier given from OpenSession	Mandatory
EntityID	Unique identifier of the camera	Mandatory

Example

- Sample request for unlocking the PTZ resource of the camera #38
  http://192.168.7.221/urlapi/control?Command=ReleasePTZResource&SessionID=4255&EntityID=38
- Positive response to the command:
  ```
  {"SNAP": {"ExecutionStatus": {"ReturnCode": 0}}}
  ```
- Negative response to the command:

  ```
  {"SNAP": {"ExecutionStatus": {
  "ErrorText": "Session not found. SessionID=4325",
  "ReturnCode": 200
  }}}
  ```

5.29 StartPTZRoute

Parameter	Description	Default Values
SessionID	The session identifier given from OpenSession	Mandatory
EntityID	Unique identifier of the camera	Mandatory
RouteID	Unique identifier of the route	Mandatory

Example

- Sample request for starting the PTZ route #1 of the camera #38
  http://192.168.7.221/urlapi/control?Command=StartPTZRoute&SessionID=4255&EntityID=38&RouteID=1
- Positive response to the command:
  ```
  {"SNAP":{"ExecutionStatus":{"ReturnCode":0}}}
  ```
- Negative response to the command:

  ```
  {"SNAP": {"ExecutionStatus": {
  "ErrorText": "Session not found. SessionID=4325",
  "ReturnCode": 200
  }}}
  ```

5.30 StopPTZRoute

Parameter	Description	Default Values
SessionID	The session identifier given from OpenSession	Mandatory
EntityID	Unique identifier of the camera	Mandatory

Example

- Sample request for stopping the PTZ route #1 of the camera #38
  http://192.168.7.221/urlapi/control?Command=StopPTZRoute&SessionID=4255&EntityID=38
- Positive response to the command:
  ```
  {"SNAP":{"ExecutionStatus":{"ReturnCode":0}}}
  ```
- Negative response to the command:

  ```
  {"SNAP": {"ExecutionStatus": {
  "ErrorText": "Session not found. SessionID=4325",
  "ReturnCode": 200
  }}}
  ```

5.31 SetPTZPosition

Parameter	Description	Default Values
SessionID	The session identifier given from OpenSession	Mandatory
EntityID	Unique identifier of the camera	Mandatory
PositionID	Identifier of the position	Mandatory

Example

- Sample request for directing the PTZ to position #1 of the camera #38
  http://192.168.7.221/urlapi/control?Command=SetPTZPosition&SessionID=4255&EntityID=38&PositionID=1
- Positive response to the command:
  {"SNAP":{"ExecutionStatus":{"ReturnCode":0}}}
- Negative response to the command:

```
{"SNAP": {"ExecutionStatus": {
"ErrorText": "Session not found. SessionID=4325",
"ReturnCode": 200
}}}
```

5.32 PTZCenterClick

Parameter	Description	Default Values
SessionID	The session identifier given from OpenSession	Mandatory
EntityID	Unique identifier of the camera	Mandatory
X	Horizontal position of the click within the frame	Mandatory
Y	Vertical position of the click within the frame	Mandatory
ImageWidth	Width of the frame in which X is measured	Mandatory
ImageHeight	Height of the frame in which Y is measured	Mandatory

Example

- Sample request for moveing the clicked 10, 10 point into the middle of the camera #38
  http://192.168.7.221/urlapi/control?Command=PTZCenterClick&SessionID=4255&EntityID=38&X=10&Y=10&ImageWidth=640&ImageHeight=480
- Positive response to the command:
  ```
  {"SNAP":{"ExecutionStatus":{"ReturnCode":0}}}
  ```
- Negative response to the command:

  ```
  {"SNAP": {"ExecutionStatus": {
  "ErrorText": "Session not found. SessionID=4325",
  "ReturnCode": 200
  }}}
  ```

5.33 ContinuousPTZ

Parameter	Description	Default Values
SessionID	The session identifier given from OpenSession	Mandatory
EntityID	Unique identifier of the camera	Mandatory
PanSpeed	Horizontal speed of the PTZ head, -100 >= speed <= 100	Mandatory
TiltSpeed	Vertical speed of the PTZ head, -100 >= speed <= 100	Mandatory
ZoomSpeed	Zoom speed of the PTZ head, -100 >= speed <= 100	Mandatory

Example

- Sample request for continuously moving the PTZ up/left of the camera #38
  http://192.168.7.221/urlapi/control?Command=ContinuousPTZ&SessionID=4255&EntityID=38&PanSpeed=10&TiltSpeed=10&ZoomSpeed=0
- Positive response to the command:
  {"SNAP":{"ExecutionStatus":{"ReturnCode":0}}}
- Negative response to the command:

```
{"SNAP": {"ExecutionStatus": {
"ErrorText": "Session not found. SessionID=4325",
"ReturnCode": 200
}}}
```

5.34 MovePTZRelative

Parameter	Description	Default Values
SessionID	The session identifier given from OpenSession	Mandatory
EntityID	Unique identifier of the camera	Mandatory
PanDir	Horizontal direction: left, right	Mandatory
PanSpeed	Horizontal speed of the PTZ head: 1, 2, 3	Mandatory
TiltDir	Vertical direction: up, down	Mandatory
TiltSpeed	Vertical speed of the PTZ head: 1, 2, 3	Mandatory
ZoomDir	Zoom direction: in, out	Mandatory
ZoomSpeed	Zoom speed of the PTZ head: 1, 2, 3	Mandatory

Example

- Sample request for moving relatively to current position of the camera #38
  http://192.168.7.221/urlapi/control?Command=MovePTZRelative&SessionID=4255&EntityID=38&PanDir=left&PanSpeed=3&TiltDir=up&TiltSpeed=1&ZoomDir=in&ZoomSpeed=0
- Positive response to the command:
  ```
  {"SNAP":{"ExecutionStatus":{"ReturnCode":0}}}
  ```
- Negative response to the command:

  ```
  {"SNAP": {"ExecutionStatus": {
  "ErrorText": "Session not found. SessionID=4325",
  "ReturnCode": 200
  }}}
  ```

5.35 ShowOnlineMonitorTool

Parameter	Description	Default Values
SessionID	The session identifier given from OpenSession	Mandatory
TargetIP	IP address of the client on which the action will be executed	Mandatory
TargetUser	The action will be executed on all clients where the givel user is logged in	Mandatory
WindowID	Identifier of the Online Monitor window for the action	Mandatory

Example

- Sample request for activating the „view1“ panel on window 1
  http://192.168.7.221/urlapi/control?Command=ShowViewOfWindow&SessionID=4255&TargetIP=&TargetUser=admin&WindowID=1&PanelName=view1
- Positive response to the command:
  ```
  {"SNAP":{"ExecutionStatus":{"ReturnCode":0}}}
  ```
- Negative response to the command:

  ```
  {"SNAP": {"ExecutionStatus": {
  "ErrorText": "Session not found. SessionID=4325",
  "ReturnCode": 200
  }}}
  ```

5.36 ShowViewOfWindow

Parameter	Description	Default Values
SessionID	The session identifier given from OpenSession	Mandatory
TargetIP	IP address of the client on which the action will be executed	Mandatory
TargetUser	The action will be executed on all clients where the givel user is logged in	Mandatory
WindowID	Identifier of the Online Monitor window for the action	Mandatory
PanelName	Name of the Online Monitor panel which will be activated	Mandatory

Example

- Sample request for activating the „view1“ panel on window 1
  http://192.168.7.221/urlapi/control?Command=ShowViewOfWindow&SessionID=4255&TargetIP=192.168.7.100&TargetUser=admin&WindowID=1&PanelName=view1
- Positive response to the command:
  ```
  {"SNAP":{"ExecutionStatus":{"ReturnCode":0}}}
  ```
- Negative response to the command:

  ```
  {"SNAP": {"ExecutionStatus": {
  "ErrorText": "Session not found. SessionID=4325",
  "ReturnCode": 200
  }}}
  ```

5.37 ShowCameraInViewport

Parameter	Description	Default Values
SessionID	The session identifier given from OpenSession	Mandatory
TargetIP	IP address of the client on which the action will be executed	Mandatory
TargetUser	The action will be executed on all clients where the givel user is logged in	Mandatory
EntityID	Camera to be displayed	Mandatory
WindowID	Identifier of the Online Monitor window for the action	Mandatory
PanelName	Name of the Online Monitor panel which will be activated	Mandatory
RowIndex	Row in which the camera will be display (or -1 for any position)	Mandatory
ColumnIndex	Column in which the camera will be display (or -1 for any position)	Mandatory

Example

- Sample request for activating camera „1“ in panel „view1“ on window 1
  http://192.168.7.221/urlapi/control?Command=ShowCameraInViewport&SessionID=4255&TargetIP=192.168.7.100&TargetUser=admin&WindowID=1&PanelName=view1&EntityID=1&RowIndex=1&ColumnIndex=1
- Positive response to the command:
  {"SNAP":{"ExecutionStatus":{"ReturnCode":0}}}
- Negative response to the command:

```
{"SNAP": {"ExecutionStatus": {
"ErrorText": "Session not found. SessionID=4325",
"ReturnCode": 200
}}}
```

5.38 StartSmartGuard

Parameter	Description	Default Values
SessionID	The session identifier given from OpenSession	Mandatory
TargetIP	IP address of the client on which the action will be executed	Mandatory
TargetUser	The action will be executed on all clients where the givel user is logged in	Mandatory
ViewGroupName	Name of guard which will be started	Mandatory

Example

- Sample request for starting guard name „G1“ on all clients where the admin user is logged in
  http://192.168.7.221/urlapi/control?Command=StartSmartGuard&SessionID=4255&TargetIP=&TargetUser=admin&ViewGroupName=G1
- Positive response to the command:
  ```
  {"SNAP":{"ExecutionStatus":{"ReturnCode":0}}}
  ```
- Negative response to the command:

  ```
  {"SNAP": {"ExecutionStatus": {
  "ErrorText": "Session not found. SessionID=4325",
  "ReturnCode": 200
  }}}
  ```

5.39 StopSmardGuard

Parameter	Description	Default Values
SessionID	The session identifier given from OpenSession	Mandatory
TargetIP	IP address of the client on which the action will be executed	Mandatory
TargetUser	The action will be executed on all clients where the givel user is logged in	Mandatory

Example

- Sample request for stopping the guard all clients where the admin user is logged in
  http://192.168.7.221/urlapi/control?Command=StopSmartGuard&SessionID=4255&TargetIP=&TargetUser=admin
- Positive response to the command:
  ```
  {"SNAP":{"ExecutionStatus":{"ReturnCode":0}}}
  ```
- Negative response to the command:

  ```
  {"SNAP": {"ExecutionStatus": {
  "ErrorText": "Session not found. SessionID=4325",
  "ReturnCode": 200
  }}}
  ```

5.40 GetLicensePlateLists

Retrieve saved license plate lists from the server.

Parameter	Description	Default Values
SessionID	The session identifier given from OpenSession	Mandatory

Example

- Sample request for getting the saved license plate lists from the server:
  http://192.168.7.221/urlapi/control?Command=GetLicensePlateLists&SessionID=4255
- Positive response to the command:

```
{
   "SNAP":{
      "LicensePlateLists":{
         "LicensePlateList":[
            {
               "Name":"my_list1",
               "ListID":1,
               "IgnoreSeparators":true,
               "Tolerance":1,
               "LicensePlates":{
                  "LicensePlate":[
                     "ACI-142",
                     "BBC-132",
                     "BOBO-00"
                  ]
               }
            },
            {
               "Name":"my_list2",
               "ListID":2,
               "IgnoreSeparators":true,
               "Tolerance":1,
               "LicensePlates":[
                  "GHI-142",
                  "BII-123",
                  "BOBO-01"
               ]
            }
         ]
      }
   }
}
```

- Negative response to the command:

```
{
   "SNAP":{
      "ExecutionStatus":{
         "ReturnCode":200,
         "ErrorText":"Session not found. SessionID: 1234"
      }
   }
}
```

5.41 SetLicensePlateLists

Set license plate lists to the desired values. Any field missing from the request json (except *ListID* will be ignored by the server (e.g.: omitting the “Name” field for a list will not change the name of the list).

This is a POST only request (you cannot use query parameters here, except for the SessionID).

Parameter	Description	Default Values
SessionID	The session identifier given from OpenSession	Mandatory
[body]	LicensePlateLists json	Mandatory

Example

- Sample setting the license plate lists:

```
POST /urlapi/control?Command=SetLicensePlateLists&SessionID=4255
Content-Type: application/json

{
   "SNAP":{
      "LicensePlateLists":{
         "LicensePlateList":[
            {
               "Name":"my_list1",
               "ListID":1,
               "IgnoreSeparators":true,
               "Tolerance":1,
               "LicensePlates":{
                  "LicensePlate":[
                     "ACI-142",
                     "BBC-132",
                     "BOBO-00"
                  ]
               }
            },
            {
               "Name":"my_list2",
               "ListID":2,
               "IgnoreSeparators":true,
               "Tolerance":1,
               "LicensePlates":[
                  "GHI-142",
                  "BII-123",
                  "BOBO-01"
               ]
            }
         ]
      }
   }
}
```

- Positive response to the command:

```
{
   "SNAP":{
      "ExecutionStatus":{
         "ReturnCode":0
      }
   }
}
```

- Negative response to the command:

```
{
   "SNAP":{
      "ExecutionStatus":{
         "ReturnCode":211,
         "ErrorText":"No licenseplate list found for ListID 1"
      }
   }
}
```