Intellect HTTP API

General information on HTTP API

HTTP API is represented by web2 module (*Web-server 2.0*).

> **Note.**
> See Administrator's Guide, Configuring the server for the clients connection via the Web-server 2.0 module section.

Port stands for port number.

/somecontext - optional web context where the app operates. This is the web-app context.

Thus there can be several systems in one domain:

www.example.com/**sistema1/**

www.example.com/**videosystem23/**

www.example.com/**a/**

This context can be more complicated:

www.example.com/**redirects/toitvwebserver/firstsystem/**

www.example.com/**redirects/toitvwebserver/secondsystem/**

www.example.com/**redirects/toitvwebserver/sauna/**

Further the description will be omitted when query action is clear in the context.

> **Important!**
> URL, id of objects and file extension are case-sensitive.

> **Note.**
> Date and time are specified in RFC 3339 format, see details at http://www.ietf.org/rfc/rfc3339.txt

Product version

`http://example.com:[port][/somecontext]/product/version`

Such URL can be used to identify server.

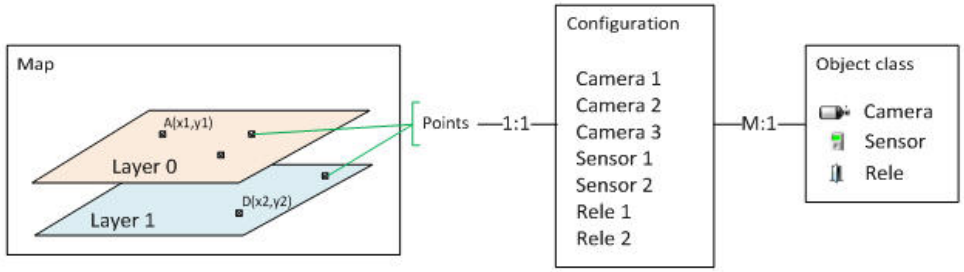
If the response is text/plain line like

Intellect 4.8.4

It means that the server supports the protocol described in this document. The line can change depending on the product version. This helps to distinguish 2 web servers with similar

functionality but different protocols in different products.

Map



Several maps can be created on the server. Each map can consists of one or more layers. There are points on each layer. Each point corresponds to one of the objects in configuration.

Configuration – objects in Intellect. Each object represents the object of specific class. Each object has one state and the list of actions to be performed.

The object class describes its icons, possible states and possible actions with the object in each state.

Getting the list of maps

There can be 0 or more maps

http://example.com:[port][/somecontext]/secure/kartas/

Sample of response:

```
<kartas>
<karta>
<id>plan</id>
<name>This is plan of a building</name>
</karta>
<karta>
<id>site</id>
<name>This is site around the building</name>
</karta>
</kartas>
```

Information on one map

plan – map id

http://example.com:[port][/somecontext]/secure/kartas/**plan**/

Sample of response:

```
<karta>
    <id>plan</id>
    <name>This is plan of a building</name>
</karta>
```

The list of layers for specific map

plan – map id

There can be 1 or more layers

http://example.com:[port][/somecontext]/secure/kartas/**plan**/layers/

Sample of response:

```
<layers>
    <layer>
        <height>1000</height>
        <id>base</id>
        <mapId>plan</mapId>
        <name>Base layer for plan</name>
        <width>1000</width>
        <zoomDef>1.0</zoomDef>
        <zoomMax>4.0</zoomMax>
        <zoomMin>0.25</zoomMin>
        <zoomStep>0.25</zoomStep>
    </layer>
</layers>
```

Parameters:

Height – height of layer’s image in pixels;

Width – width of layer’s image in pixels;

zoomMin – minimal zoom;

zoomMax – maximal zoom;

zoomStep – zoom step when zooming in or zooming out;

zoomDef – zoom by default.

Sample. If image width is 100 pixels, then width for 0.25 zoom is 100*0.25=25 pixels.

Information on a specific layer

Information on parameters is given in The list of layers for specific map section.

base – layer id

http://example.com:[port][/somecontext]/secure/kartas/**plan**/layers/**base**/

Sample of response:

```
<layer>
    <height>1000</height>
    <id>base</id>
    <mapId>plan</mapId>
    <name>Base layer for plan</name>
    <width>1000</width>
    <zoomDef>1.0</zoomDef>
    <zoomMax>4.0</zoomMax>
    <zoomMin>0.25</zoomMin>
    <zoomStep>0.25</zoomStep>
</layer>
```

Layer background

http://localhost:8080/server-1.0/secure/kartas/**plan**/layers/**base**/image.[png|jpg]

Image in png or jpg format comes in response.

404 error comes to JPG, Jpg, JPEG, PNG query.

The list of point on the layer

http://example.com:[port][/somecontext]/secure/kartas/**plan**/layers/**base**/points/

id is the same as id of object in configuration. Id is not necessarily equal to CAM:1. Id is to be considered as a line.

The coordinate plane is attached to the layer as follows:

I.e. x and y cannot be negative, but can be fractional.

Sample of response:

```
<points>
    <point>
        <id>CAM:1</id>
        <layerId>base</layerId>
        <mapId>plan</mapId>
        <x>100.0</x>
        <y>100.0</y>
    </point>
    <point>
        <id>CAM:2</id>
        <layerId>base</layerId>
        <mapId>plan</mapId>
        <x>200.0</x>
        <y>200.0</y>
    </point>
    <point>
```

```
        <id>GRAY:1</id>
        <layerId>base</layerId>
        <mapId>plan</mapId>
        <x>300.0</x>
        <y>300.0</y>
    </point>
    <point>
        <id>GRAY:2</id>
        <layerId>base</layerId>
        <mapId>plan</mapId>
        <x>400.0</x>
        <y>400.0</y>
    </point>
    <point>
        <id>GRELE:1</id>
        <layerId>base</layerId>
        <mapId>plan</mapId>
        <x>500.0</x>
        <y>500.0</y>
    </point>
    <point>
        <id>GRELE:2</id>
        <layerId>base</layerId>
        <mapId>plan</mapId>
        <x>600.0</x>
        <y>600.0</y>
    </point>
</points>
```

Information on a specific point on the layer

http://example.com:[port][/somecontext]/secure/kartas/**plan**/layers/**base**/points/CAM:2 – request for information on a point corresponding to the camera with id 2.

Sample of response:

```
<point>
<id>CAM:2</id>
<layerId>base</layerId>
<mapId>plan</mapId>
<x>200.0</x>
<y>200.0</y>
</point>
```

Object classes

The list of object classes on the server

http://example.com:[port][/somecontext]/secure/objectClasses

Sample of response:

```
<objectClasses>
    <objectClass>
        <id>GRELE</id>
    </objectClass>
    <objectClass>
        <id>USERS</id>
    </objectClass>
    <objectClass>
        <id>CAM</id>
    </objectClass>
    <objectClass>
        <id>RIGHTS</id>
    </objectClass>
```

```
    <objectClass>
        <id>GRAY</id>
    </objectClass>
</objectClasses>
```

Specific object class

http://example.com:[port][/somecontext]/secure/objectClasses/**GRELE**/

Sample of response:

```
<objectClass>
    <id>GRELE</id>
</objectClass>
```

The list of states for a specific object class

http://example.com:[port][/somecontext]/secure/objectClasses/**GRELE**/states/ - get the list of states for the **Relay** object class.

Sample of response:

```
<states>
    <state>
        <id>off</id>
    </state>
    <state>
        <id>on</id>
    </state>
    <state>
        <id>disabled</id>
    </state>
</states>
```

Information on a specific state

http://example.com:[port][/somecontext]/secure/objectClasses/**[ObjectClass]**/states/**[State]**/

Sample:

http://example.com:[port][/somecontext]/secure/objectClasses/**GRELE**/states/**off**/ - getting information on the OFF state for the **Relay** object class.

Sample of response:

```
<state>
    <id>off</id>
</state>
```

Getting the icon for a specific state

http://example.com:[port][/somecontext]/secure/objectClasses/**[ObjectClass]**/states/**[State]**/image.png

Sample:

http://example.com:[port][/somecontext]/secure/objectClasses/**GRELE**/states/**off**/image.png - getting the icon for the OFF state for the **Relay** object class.

The image in png format comes in response:

The list of events for a specific object class

GET

http://example.com:[port][/somecontext]/secure/objectClasses/**GRELE**/events/ - getting the list of events for the **Relay** object class.

Sample of response:

XML

```
<events>
    <event>
        <id>23</id>
        <sid>grele.disable</sid>
        <description>Disable rele</description>
    </event>
    <event>
        <id>24</id>
        <sid>grele.enable</sid>
        <description>Enable rele</description>
    </event>
```

```
<baseObject>
    <CAM>
        <id>CAM:1</id>
        <name>-</name>
        <state>
            <id>disconnected</id>
        </state>
    </CAM>
    <GRELE>
        <id>GRELE:2</id>
        <name>[GRELE] 2</name>
        <state>
            <id>off</id>
        </state>
    </GRELE>
    <GRELE>
        <id>GRELE:1</id>
        <name>Relay 1</name>
        <state>
            <id>disabled</id>
        </state>
    </GRELE>
    <GRAY>
        <id>GRAY:2</id>
        <name> 2</name>
        <state>
            <id>alarmed</id>
        </state>
    </GRAY>
```

```
</baseObjects>
```

Objects

Getting list of objects

http://example.com:[port][/somecontext]/secure/configuration/

JSON

```
[ {
  "type" : "CAM",
  "id" : "CAM:2",
  "extId" : "2",
  "name" : "Camera 2",
  "regionId" : "2.1",
  "state" : {
    "id" : "alarmed",
    "type" : "ALARM"
  },
  "presets" : [ ]
}, {
  "type" : "CAM",
  "id" : "CAM:1",
  "extId" : "1",
  "name" : "Camera 1",
  "state" : {
    "id" : "armed",
    "type" : "NORMAL"
  },
  "presets" : [ ]
}, {
  "type" : "GRAY",
  "id" : "GRAY:1",
  "extId" : "1",
  "name" : "Sensor 1",
  "state" : {
    "id" : "disconnected",
    "type" : "ALARM"
  }
}, {
  "type" : "GRELE",
  "id" : "GRELE:2",
  "extId" : "2",
  "name" : "Relay 2",
  "state" : {
    "id" : "disabled",
```

```
    "type" : "NORMAL"
  }
}, {
  "type" : "GRELE",
  "id" : "GRELE:1",
  "extId" : "1",
  "name" : "Relay 1",
  "regionId" : "2.1",
  "state" : {
    "id" : "disabled",
    "type" : "NORMAL"
  }
}, {
  "type" : "GRAY",
  "id" : "GRAY:2",
  "extId" : "2",
  "name" : "Sensor 2",
  "state" : {
    "id" : "disconnected",
    "type" : "ALARM"
  }
} ]
```

Information on a specific object

http://example.com:[port][/somecontext]/secure/configuration/GRAY:2/ - getting information on the **Sensor** object with id 2.

Sample of response:

```
<GRAY>
    <id>GRAY:2</id>
    <name> 2</name>
    <state>
        <id>alarmed</id>
    </state>
</GRAY>
```

State of a specific object

http://example.com:[port][/somecontext]/secure/configuration/GRAY:2/state/ - getting a state for the **Sensor** object with id 2.

Sample of response:

```
<state>
    <id>alarmed</id>
```

```
</state>
```

The list of available actions with the object in a specific state

The list of actions is requested not by the object class, but is taken in the context of a specific object as various user rights are possible for the objects of the same class.

Information on how to use this list is given in Sending commands to server section.

http://example.com:[port][/somecontext]/secure/configuration/**GRAY:2**/state/actions/ - getting the list of available actions for the **Sensor** object with id 2.

Sample of response:

```
<actions>
    <action>
        <description>Disarm ray</description>
        <id>ray.disarm</id>
    </action>
    <action>
        <description>Confirm alarm</description>
        <id>ray.confirm</id>
    </action>
</actions>
```

If the object state considers no actions, then xml is:

```
<actions/>
```

Getting events

Connection is not lost and events are always received.

action – type of event. Possible values: create, delete, update.

The fields below are optional:

objectId - id of object that is the source of event (always with update, delete, create).

state – id of a new object state (always with create. If the state has not changed, then there is no update state).

x, y – new coordinates (if changed).

Query:

http://example.com:[port][/somecontext]/secure/feed/

Samples of response:

```
<message>
    <action>update</action>
    <objectId>CAM:1</objectId>
    <state>disconnected</state>
</message>

<message>
    <action>state</action>
    <objectId>CAM:1</objectId>
    <x>10.0</x>
    <y>123.9</y>
</message>

<message>
    <action>state</action>
    <objectId>CAM:1</objectId>
    <state>connected</state>
    <x>300.8</x>
    <y>670</y>
</message>

<message>
    <action>state</action>
    <objectId>CAM:1</objectId>
    <x>100</x>
    <y>100</y>
</message>
```

```
<message>
    <action>ping</action>
</message>
```

Getting events of video subsystem in blocks

`http://example.com:[port][/somecontext]/secure/events/`

Parameters:

from – the oldest date of message search period. 2012-12-27T15%3A19%3A16.000%2B04%3A00

to – the latest date of message search period. 2012-12-27T15%3A19%3A16.000%2B04%3A00

count – maximum number of messages in reply [1, 200]. On default – 20. Server can return more messages if there are few messages in the database.

objectId – object id (CAM:1, GRAY:5 etc.). If parameter is not specified, events are returning for all.

Return codes:

200 - OK
400 - invalid parameter (e.g. date format)
500 - error
503 - error of core connection
504 - time-out (core failed to return data within 2000 milliseconds)

Examples of reply

XML:

```
<events>
    <event>
        <description>Record is on</description>
        <id>{E56B09A0-1A50-E211-840E-005056C00008}</id>
        <objectId>CAM:1</objectId>
        <ts>2012-12-27T15:43:27+04:00</ts>
    </event>
    <event>
        <description>Record is off</description>
        <id>{4482F63F-1A50-E211-840E-005056C00008}</id>
```

```
        <objectId>CAM:1</objectId>
        <ts>2012-12-27T15:40:50+04:00</ts>
    </event>
    <event>
        <description>Record is off</description>
        <id>{35D4BE3E-1750-E211-840E-005056C00008}</id>
        <objectId>CAM:1</objectId>
        <ts>2012-12-27T15:19:16+04:00</ts>
    </event>
</events>
```

JSON:

```
[ {
  "id" : "{E56B09A0-1A50-E211-840E-005056C00008}",
  "description" : "Record is off",
  "ts" : "2012-12-27T15:43:27.000+04:00",
  "objectId" : "CAM:1"
}, {
  "id" : "{4482F63F-1A50-E211-840E-005056C00008}",
  "description" : "Record is off",
  "ts" : "2012-12-27T15:40:50.000+04:00",
  "objectId" : "CAM:1"
}, {
  "id" : "{35D4BE3E-1750-E211-840E-005056C00008}",
  "description" : "Record is off",
  "ts" : "2012-12-27T15:19:16.000+04:00",
  "objectId" : "CAM:1"
} ]
```

Sending commands to server

PUT

http://example.com:[port][/somecontext]/secure/configuration/**GRAY:2**/state/actions/**disarm**/execute - sample of how to send the disarm **Sensor** with id 2 command to the sever.

Macros

In the section:
• Getting the list of macros • Getting parameters of macros • Macros execution on server request

Macros - predefined sequence of responses to certain events. Macros are created on the server and have IDs and names. They are similar to actions with objects, but are not attached to the object.

Getting the list of macros

GET

http://example.com:[port][/somecontext]/secure/actions/

Sample of response:

```
<actions>
    <action>
        <description>Start recording by all cameras</description>
        <id>macro2</id>
    </action>
    <action>
        <description>Disarm all zones</description>
        <id>1</id>
    </action>
</actions>
```

Getting parameters of macros

Object has no extra parameters. The list of macros is sufficient.

GET

http://example.com:[port][/somecontext]/secure/actions/**macro2**/ - getting parameters of macro with macro2 id.

Sample of response:

```
<action>
    <description>Start recording by all cameras</description>
    <id>macro2</id>
</action>
```

Macros execution on server request

PUT

http://example.com:[port][/somecontext]/secure/actions/**macro2**/execute – request to execute macro with id macro2 on the server.

Video

Configuration request

GET

http://example.com:[port][/somecontext]/secure/video/config.properties?version=4.7.8.0&login=XXX&password=YYY

Parameters:

- version – mandatory field. Client version (if protocol is changed). Now the **4.7.8.0** value is to be sent.
- login – optional field.
- password - optional field. It is in use if access is protected with the password.

Features of using

At the start it is not clear if the login and password are in use. To make it clear, send the following query:

GET

http://www.examplehost.com[/somecontext]/secure/video/config.properties?version=4.7.8.0

The server sends back the config.properties text file:

```
password.enabled=true
login.enabled=true
password.invalid=true#
```

Note.
symbol means the ending of configuration file.

When you get this file, it becomes clear that the password has been set and it is not correct. It is not correct because the login and password fields were blank.

Request the login and password and send configuration query to the server once again:

GET

http://www.examplehost.com[/somecontext]/secure/video/config.properties?version=4.7.8.0&login=XXX&password=YYY

If the password is correct or access is allowed without the password, then the server sends the following configuration:

```
password.enabled=true
login.enabled=true
password.invalid=false
cam.0.id=2
cam.0.name=Face
cam.0.rights=11
cam.1.id=3
cam.1.name=Camera 3
cam.1.rights=11
cam.2.id=5
cam.2.name=Camera 5
cam.2.rights=11
cam.2.telemetry_id=1.1
cam.count=3#
```

password.invalid=false means the specified password is not correct.

Note.
If access is allowed without the password, then password.enabled=false and configuration will be received on the first try.

cam.count=3 - total of cameras in the configuration (id starts at 0).

Get the data for each of 3 cameras in configuration.

cam.N.id – camera id.

cam.N.name - camera name.

cam.N.rights – rights.

cam.N.telemetry_id – telemetry id (there can be no value if there is no telemetry – then hide telemetry control elements).

cam.N.rights – rights (they are checked on the server; available on the client in order not to show the user odd options). The parameter is represented by flags. If the flag is set, then the interface element is to be shown; if not – hidden.

```
static final int RIGHT_VIEW = 0x1;  // live video is available(always 1)
static final int RIGHT_CONTROL = 0x2; // control (telemetry, arming and disarming)
static final int RIGHT_CONFIG = 0x4;  // reserved
static final int RIGHT_HISTORY = 0x8; // access to archive
```

Video query

http://example.com:[port][/somecontext]/secure/video/action.do?version=4.7.8.0&sessionid=FC126734&cam.id=5&login=XXX&password=YYY – request of video for camera with id 5.

cam.id – camera id.

sessionid – any value.

Format of main stream

Stream of following view will be received in reply

```
HTTP/1.0 200 OK
Connection: close
Server: ITV-Intellect-Webserver/4.9.0.0
Cache-Control: no-store,no-cache,must-revalidate,max-age=0
Pragma: no-cache
Date: Mon, 13 Jan 2013 10:44:27 GMT
Content-Type: multipart/mixed;boundary=videoframe

--videoframe
Content-Type: text/xml
Content-Length: 138

<video_in>
  <sessionid>FC126734</sessionid>
```

```
  <video_in>CAM:5</video_in>
  <newstate>started</newstate>
  <errcode>100</errcode>
</video_in>
 --videoframe
Content-Type: image/jpeg
Content-Length: 23978
X-Width: 320
X-Height: 240
X-Time: 2013-03-15T10:51:44.314+04:00
X-Timestamp: 0.000

 <jpeg image>
--videoframe
Content-Type: image/jpeg
Content-Length: 23651
X-Width: 320
X-Height: 240
X-Time: 2013-03-15T10:51:44.314+04:00
X-Timestamp: 0.152

 <jpeg image>
```

Where:

- X-Width - image width.
- X-Height- image height.
- X-Time - absolute time of frame forming.
- X-Timestamp - relative frame time in seconds (relatively stream head).

In case of stream end due to the fault of server, the end packet can be received:

```
--videoframe
Content-Type: text/xml
Content-Length: 106
<video_in>
  <sessionid>FC126734</sessionid>
  <video_in>CAM:5</video_in>
  <newstate>closed</newstate>
  <errcode>103</errcode>
</video_in>
```

- sessionid - session ID (the same as at start).
- video_in - camera ID.
- errcode - error code:
    - 100 – error absence.
    - 101 – too many connected users.
    - 102 – invalid password (theoretically, it's possible to change password at any moment).
    - 103 – unavailable video.
    - 104 – old client version. Update version.

Managing records

Start recording

GET

http://example.com:[port][/somecontext]/secure/video/action.do?version=4.7.8.0&sessionid=29101F1&cam.id=1&target=CAM&targetid=1&command=REC&login=XXX&password=YYY

Stop recording

GET

http://example.com:[port][/somecontext]/secure/video/action.do?version=4.7.8.0&sessionid=29101F1&cam.id=5&target=CAM&targetid=1&command=REC_STOP&login=XXX&password=YYY

Here targetid==cam.id

Arming and disarming the camera

Arming

GET

http://example.com:[port][/somecontext]/secure/video/action.do?version=4.7.8.0&sessionid=29101F1&cam.id=1&target=CAM&targetid=1&command=ARM&login=XXX&password=YYY

Disarming

GET

http://example.com:[port][/somecontext]/secure/video/action.do?version=4.7.8.0&sessionid=29101F1&cam.id=5&target=CAM&targetid=1&command=DISARM&login=XXX&password=YYY

targetid==cam.id

PTZ control

GET

http://example.com:[port][/somecontext]/secure/video/action.do?version=4.7.8.0&sessionid=29101F1&cam.id=5&target=PTZ&targetid=1.1&command=RIGHT&login=XXX&password=YYY&speed=2

Parameters:

command – required parameter. It takes the following values:

- RIGHT
- UP
- LEFT
- DOWN
- ZOOM_IN
- ZOOM_OUT
- GO_PRESET – go to the specified preset.
- POINTMOVE – zooming of selected point on the image (x,y).
- AREAZOOM – zooming of selected area on the image (x,y,w,h).

speed – required parameter. Command-processing speed (from 0 to 10). It is recommended to use low values due to delays while controlling by network

cam.id – required parameter. Camera ID.

target – required parameter. Always is equal to PTZ.

targetid – required parameter. Id of PTZ connected with camera (is sent in the SETUP configuration).

preset – number of preset. Required parameter only for the command=GO_PRESET. Otherwise its value is ignored.

x – x coordinate relatively the image size. It takes values from 0.0 to 1.0. Required parameter only for the command=POINTMOVE or command=AREAZOOM. Otherwise its value is ignored.

y – y coordinate relatively the image size. It takes values from 0.0 to 1.0. Required parameter only for the command=POINTMOVE or command=AREAZOOM. Otherwise its value is ignored.

w – width of zooming area relatively the image size. It takes values from 0.0 to 1.0. Required parameter only for the command=AREAZOOM. Otherwise its value is ignored.

h – height of zooming area relatively the image size. It takes values from 0.0 to 1.0. Required parameter only for the command=AREAZOOM. Otherwise its value is ignored.

Using the archive

In the section:

Video archive stream is sent on the same format as live video.

Getting list of records - "arc.intervals"

GET

http://example.com:[port][/somecontext]/secure/video/action.do?version=4.9.0.0&sessionid=29101F1&video_in=CAM:5&command=arc.intervals&time_from=2013-03-20T00:00:00.000+04:00&time_to=2013-03-22T23:59:59.999+04:00&max_count=100&split_threshold=10399&login=XXX&password=YYY

Required parameters:

command=arc.intervals – command to receive list of records

video_in – camera ID.

time_from – start of interested time range.

Optional parameters:

time_to – start and end of interested time range.

max_count – maximal number of records in reply

split_threshold – time for combining several intervals (in seconds). Intervals, distance between which will be less than specified value, will be combined in one.

In return **XML** will be received**:**

```
<?xml version="1.0" encoding="UTF-8"?>
<records>
    <record>
        <from>2011-09-01T00:00:00-05:00</from>
        <to>2011-09-01T00:00:35-05:00</to>
    </record>
    <record>
        <from>2011-09-01T00:00:35-05:00</from>
        <to>2011-09-01T00:01:10-05:00</to>
    </record>
</records>
```

Getting one frame from archive - "arc.frame"

GET

http://example.com:[port][/somecontext]/secure/video/action.do?version=4.9.0.0&sessionid=29101F1&video_in=CAM:5&command=arc.frame&time=2013-03-22T13:04:52.312+04:00&range=0.1&login=XXX&password=YYY

Required parameters:

command=arc.frame - command for one frame;

video_in - camera ID;

time - interested time.

Optional parameters:

range - time in seconds to specify search range of the nearest frame relatively the time parameter (the nearest frame all over archive is searched if this parameter is not specified);

imageWidth - width in pixels (is counted automatically with saving proportions if it isn't specified or equal 0);

imageHeight - height in pixels (is counted automatically with saving proportions if it isn't specified or equal 0);

fps - maximal frame frequency per second (if it isn't specified or equal 0, frame frequency won't be limited if).

In return http-headings and the nearest frame from the [time - range, time + range] range in the jpeg format will be received. The reply body will be empty if there is no frame in the range.

Getting video from archive - "arc.play"

GET

http://example.com:[port][/somecontext]/secure/video/action.do?version=4.9.0.0&sessionid=29101F1&video_in=CAM:5&command=arc.play&time_from=2013-03-22T13:04:52.312+04:00&time_to=2013-03-22T13:16:31.873+04:00&login=XXX&password=YYY

Required parameters:

command=arc.play - command to get video from archivea;

video_in - camera ID;

time_from - start time of archive playing back.

Optional parameters:

time_to - completion time of archive playing back (if parameter is not specified, all archive will play back);

imageWidth - width in pixels (is counted automatically with saving proportions if it isn't specified or equal 0);

imageHeight - height in pixels (is counted automatically with saving proportions if it isn't specified or equal 0);

maximal frame frequency per second (if it isn't specified or equal 0, frame frequency won't be limited if).

End packet with newstate=closed and errcode=100 will be received when stream completion.

Getting list of records (the second way)

GET

http://example.com:[port][/somecontext]/secure/archive/**CAM:2/[2011-12-30|2011-12]?[splitThreshold=50]&[days=1]**

splitThreshold – if difference between end of previous record and start of next record less than specified value (in milliseconds), than records will be combined in one. Specify splitTreshold=0. [default: 50] not to combine records.

days - number of days from the current, for which archive is required. [default: 1]

All time is interpreted as local time for server.

Get records for 18 November 2013 and combine all records, interval between which less than 2000 milliseconds:

http://example.com:[port][/somecontext]/secure/archive/**CAM:1/2013-10-18/?splitTreshold=2000**

Get records for 10 days from 18 November 2013:

http://example.com:[port][/somecontext]/secure/archive/**CAM:1/2013-10-18/?days=10**

XML:

```
<?xml version="1.0" encoding="UTF-16"?>
<days>
    <day>
        <id>2013-11-10T00:00:00-02:00</id>
        <records>
            <from>2013-11-10T18:44:01.579-02:00</from>
            <to>2013-11-10T18:44:09.717-02:00</to>
        </records>
    </day>
    <day>
        <id>2013-11-18T00:00:00-02:00</id>
        <records>
            <from>2013-11-18T18:38:30.252-02:00</from>
            <to>2013-11-18T18:38:56.942-02:00</to>
        </records>
        <records>
            <from>2013-11-18T18:39:08.321-02:00</from>
            <to>2013-11-18T18:39:10.080-02:00</to>
        </records>
    </day>
</days>
```

JSON:

```
[ {
  "id" : "2013-11-10T00:00:00.000-02:00",
  "records" : [ {
    "from" : "2013-11-10T18:44:01.579-02:00",
    "to" : "2013-11-10T18:44:09.717-02:00"
  } ]
}, {
  "id" : "2013-11-18T00:00:00.000-02:00",
  "records" : [ {
    "from" : "2013-11-18T18:38:30.252-02:00",
    "to" : "2013-11-18T18:38:56.942-02:00"
  }, {
    "from" : "2013-11-18T18:39:08.321-02:00",
    "to" : "2013-11-18T18:39:10.080-02:00"
  } ]
} ]
```


Getting records for a month (shows days of September at which there are records):

http://example.com:[port][/somecontext]/secure/archive/**CAM:2/2011-12/**

XML:

```
<?xml version="1.0" encoding="UTF-8"?>
<days>
    <day>
        <id>2011-09-02T00:00:00-05:00</id>
    </day>
    <day>
        <id>2011-09-03T00:00:00-05:00</id>
    </day>
    <day>
        <id>2011-09-05T00:00:00-05:00</id>
    </day>
</days>
```

JSON:

```
[ {
  "id" : "2011-09-01T00:00:00-0500",
  "records" : [ ]
}, {
  "id" : "2011-09-03T00:00:00-0500",
  "records" : [ ]
}, {
  "id" : "2011-09-01T00:00:00-0500",
  "records" : [ ]
} ]
```

If there are no records, the following will be received

XML:

<days/>

JSON:

[]

Thumbnails request

http://example.com:[port][/somecontext]/secure/video/image.jpg?cam.id=1&version=4.7.8.0

Parameters:
cam.id – required. Camera ID.
width – optional. Value can be in [64, 1600] range. Server automatically rounds width to larger value divisible by 4.
height – optional. Value can be in [30, 1200] range.
version – optional. client version (for protocol exchange). It's required to send "4.7.8.0" value now.
login – optional. Login;
password – optional. If password access is set.

Size of returned image is taken from video stream if width and height parameters are not set.

Returned value:
jpeg image of approximately requested size.

In case of error the http error code is returned:
404 – camera disabled or not in use (disabled);
403 – invalid password;
426 – old client version;
429 – too many requests;
444 – camera signal is lost or camera disabled (coaxial conductor disconnected from card);
503 – archive error.

Example:
Get image from camera 5, width is approximately 85 pixels:
http://localhost:8079/web2/secure/video/image.jpg?cam.id=5&width=85&version=4.7.8.0&login=username&password=secret

Jpg image of width 88 pixels or error code and zero length body (i.e. only headings) will be received in reply).

Notification

In the section:
• Subscribing • Subscription cancellation • APN message format

APNS(iOS), C2DN (Android), etc. notification systems are in use.

deviceid – device token (APNs), registration id (C2DN), etc.;

username – login. The field can be blank.

Subscribing

When the app connects to service, the APNS messages subscription is performed. So when the app is closed, device will receive events notifications.

POST

http://example.com:[port][/somecontext]/secure/subscription/

Reply with “201 Created” code means that subscribing is performed successfully.

Code 400 means that specified parameters are invalid (deviceId is not to be empty, it's length should be from 5 to 150 symbols and contain only digits and letter of English alphabet).

The POST body should contain information about created subscribing. Only JSON format is received. It's required to specify the Content-Type title properly.

Sample of response:

JSON

Content-Type : application/json

```
{
    "username" : "johndoe",
    "deviceid" : "somedeviceid"
}
```

Subscription cancellation

Subscription is cancelled in the following cases:

- The user subscribed to events from other device;
- Device token or registration id is changed;
- Another user subscribed to events from this device;
- Subscription is cancelled manually.

DELETE

http://example.com:[port][/somecontext]/secure/subscription/[deviceId]

Reply with “204 No Content” code means that subscribing is performed successfully.

APN message format

```
{
"aps" : {
                "alert" : "Motion Detected",
                "badge" : 2 //ordinal number of the message. Numbers are given one after another after the latest subscribing.
         },
    "e" : {
            "srv" : "XXX",  //server id. Unique in one iOS device
            "stt" : 88,   //state id(see The list of states for a specific object class)
            "obj" : "6", //object id
            "ts" : "2010-08-02T23:30:00Z" //time of sending the event
         }
}
```

Sound

Getting live sound

GET

http://example.com:[port][/somecontext]/secure/video/action.do?version=4.9.0.0&sessionid=FC126734&command=audio.play&audio_in=MIC:5&format=L16&login=XXX&password=YYY

sessionid – session ID (is not in use yet).
audio_in – audio stream ID.
format – format of audio data (only L16 yet).

Audio packets of the following view will be received:

```
HTTP/1.0 200 OK

Connection: close

Server: ITV-Intellect-Webserver/4.9.0.0

Cache-Control: no-store,no-cache,must-revalidate,max-age=0

Pragma: no-cache

Date: Mon, 13 Jan 2013 10:44:27 GMT

Content-Type: multipart/mixed;boundary=audioframe
```

```
--audioframe
Content-Type: text/xml
Content-Length: 138

<audio_in>
  <sessionid>FC126734</sessionid>
  <audio_in>MIC:5</audio_in>
  <newstate>started</newstate>
  <errcode>100</errcode>
</audio_in>
--audioframe
Content-Type: audio/L16;rate=8000;channels=1
Content-Length: 1024
X-Time: 2013-03-22T13:16:31.371+04:00

<audio packet PCM16>
--audioframe
Content-Type: audio/L16;rate=8000;channels=1
Content-Length: 1278
X-Time: 2013-03-22T13:16:31.873+04:00
<audio packet PCM16>
```

To stop stream without connection break send the following command in the same connection

GET
http://example.com:[port][/somecontext]/secure/video/action.do?version=4.9.0.0&sessionid=29101F1&command=audio.stop&audio_in=MIC:5&login=XXX&password=YYY

The end xml packet will be received::

```
--audioframe
Content-Type: text/xml
Content-Length: 106
<audio_in>
  <sessionid>FC126734</sessionid>
  <audio_in>MIC:5</audio_in>
  <newstate>closed</newstate>
  <errcode>100</errcode>
 </audio_in>
```

Playing sound from archive

GET

http://example.com:[port][/somecontext]/secure/video/action.do?version=4.9.0.0&sessionid=29101F1&command=arc.play&audio_in=MIC:5&format=L16&time_from=2013-03-22T13:16:31.873+04:00&time_to=2013-03-22T13:04:52.312+04:00&login=XXX&password=YYY

audio_in – audio stream ID;

format – requested format (only L16 yet);

time_from - start time of archive playing back;

time_to - end time of archive playing back.

The stream will be received in the same view as in case of live sound.

The end xml packet will be received when data completion (as when getting live sound – see Getting live sound).

Sending live sound

Sending of sound is performed by serial communication of packets using commands:

POST

http://example.com:[port][/somecontext]/secure/video/action.do?version=4.9.0.0&sessionid=FC126734&command=audio.receive&audio_out=SPEAKER:3&login=XXX&password=YYY

```
Content-type: audio/L16;rate=8000;channels=1
Connection: keep-alive
```

Then audio packet transmission will perform.

Format of sound – only L16.
rate – any rational value.
channels – from 1 to 6.