Table of Contents

1 Introduction

VDG Sense uses an HTTP / XML protocol for communication of third party software. This interface is a set of http and xml-messages that are sent over a socket interface to and from one or more client applications. These messages can be instructions to obtain data like camera names, layout settings, record data and a list of connected subsystems. These messages can also obtain instructions to be executed, like layout switching and alarm activation. Live and recorded images can also be retrieved.

Below is a quick overview of functions that can be retrieved via API:

- VDG Sense settings
- Connected devices/cameras
- Device/camera settings
- Connected VDG Sense subsystems
- Layout settings
- Event definitions
- Historical record data, with various filter
- Historical event data, with various filters

A full list of functions is available under Functions.

2 Usage

Here developers can find how the API communicates and which commands can be used. Select one of the topics to get more information on that API subject.

The functions describes all the HTTP functions available. Every function describes its purpose, input and output parameters, and arguments. The HTTP request displayed is only the argument-part of the URL. For example, when the command describes a request as:

```
command=getCameraList
```

the complete request for a VDG Sense setup at localhost should be:

```
http://127.0.0.1/command?command=getCameraList
```

Every function chapter explains the following sections:

Description	This is the description of the function.
Request	This describes the request to be send.
Parameters	This is a list explaining all parameters from the request. Between [and] is the type. If the type is omitted, it is assumed to be of type string.
Response	This describes the response. In most cases, the response is an xml message. If the answer is not the default answer, it is listed here.
Tags	If the response is not the default answer, this list explains all tags from the result. Between [and] is the type. If the type is omitted, it is assumed to be of type string.
See Also	Displays a list with useful links.
Change log	Displays a list with changes for the function.
Examples	Displays some examples if available.

2.1 Authentication

The HTTP interface can be operated by sending HTTP/1.1 compliant requests and retrieving their responses. Requests consist of a single file request, with optionally HTTP/GET parameters and can contain optional HTTP/POST XML data. All responses sent by the HTTP interface will contain XML HTTP/POST data. The following three files are relevant to user authentication, where [server] is the reachable address of manager:

- http://[server]/info
- http://[server]/webservice
- http://[server]/command

In addition, some request and responses contain a UTC timestamp. These timestamps are formatted as follows: yyyy-mm-dd hh:mm:ss. For example, 2013-09-03 19:05:55 represents 5 minutes and 55 seconds past 7 p.m. on the 3rd of September 2013 in UTC time.

Retrieve server-side information

API version 2.6.1 and above contain the /info request file. When called, this request will return the following response:

```
<?xml version="1.0" encoding="UTF-8"?>
<apiinfo>
    <utc>[current time on the server]</utc>
    <version>[api version]</version>
</apiinfo>
```

For example, a /info request made on the date and time above on version 2.6.1 will return the following response in its HTTP/POST data

```
<?xml version="1.0" encoding="UTF-8"?>
<apiinfo>
    <utc>2013-09-03 19:05:55<!-- utc-->
    <version>2.6.1</version>
</utc></apiinfo>
</apiinfo>
```

Take note that earlier versions of the API will return a 404 NOT FOUND error when requesting /info. Use this to work out that the 2.6.1 features are not available. A /info request does not require authentication.

Digest authentication

The digest authentication method uses a so-called 'nonce' to identify the type of client that connects. An example message of the digest authentication method is:

```
<?xml version='1.0'?>
```

```
<AuthenticateUserDigest>
    <username>user</username>
    <nonce>AR5chsWVZagPfMpB</nonce>
    <timestamp>2013-09-04 08:38:43</timestamp>
    <digest>804a2cba7610088a6c7975777e6349daefadcdf9</digest>
</AuthenticateUserDigest>
```

In this message, the nonce identifies the client type. These nonces will be given to API integrators based on the type of connection they will use. The timestamp is the current time in UTC. The digest is a digital signature to verify the message contents. Note that there is no password field. The password is used when the digest string is created and this string contains enough information for the API to verify that the sender of the message knows the correct password.

Digest calculation

This section describes how to calculate the digest string. First, some input values need to be determined: username, password, current time and the nonce.

USR: user
PWD: password
TIME: 2013-09-04 08:38:43
NONCE: AR5chsWVZagPfMpB

With the time, username and password, the following string can be formed:

MD5(TIME)+USR+SHA1(SHA1(PWD)) =
a268f1c72dea7d9d677e365d1285fd78user2470c0c06dee42fd1618bb99005adca2ec9d1e19

In other words, this means that the resulting string is a concatenation of an MD5 hash of the time string, the username and a double SHA1 hash of the password. Note that the input for the second SHA1 calculation uses the binary result of the first one, not its string representation. This string and the nonce are then used to create the final message authentication code with HMAC-SHA1. The string should be used as key and the nonce should be used as data in the HMAC-SHA1 function.

DATA = AR5chsWVZagPfMpB
KEY = a268f1c72dea7d9d677e365d1285fd78user2470c0c06dee42fd1618bb99005adca2ec9d1e19
HMAC-SHA1 (DATA, KEY) = 804a2cba7610088a6c7975777e6349daefadcdf9

The resulting string can be used as the digest field in the authentication message. The authentication message should be sent to the /webservice request file.

PHP example of digest calculation

```
// User input, generated time and app nonce
$username  = "user";
$password  = "password";
$timestamp = gmdate("Y-m-d H:i:s"); // e.g. 2013-09-04 08:38:43
$nonce     = "AR5chsWVZagPfMpB";

// Note that the result of the first sha1 is binary.
```

```
$string = md5( $timestamp ) . $username . sha1( sha1( $password, true ) );

// Note that $nonce is used as data and $string as key.
$digest = hash_hmac( "sha1", $nonce, $string );
```

Response

If the authentication was successful, the system will respond with a session key. Include this in further requests which require authentication. The following is an example of a successful login response.

```
<?xml version="1.0" encoding="UTF-8"?>
<AuthenticateUserDigestResponse>
    <result>OK</result>
    <sessionkey>275000862</sessionkey>
    <apiversion>2.6.1</apiversion>
</AuthenticateUserDigestResponse>
```

In case of error, the system will respond with an error string as seen below.

```
<?xml version="1.0" encoding="UTF-8"?>
<AuthenticateUserDigestResponse>
    <result>ERROR</result>
    <message>Authentication failed</message>
</AuthenticateUserDigestResponse>
```

Basic authentication

Servers using an API version prior to 2.6.1 (which does not support /info) can be reached by using basic authentication. Mobile notifications are not supported using this version. A basic authentication request is also sent to the /webservice file and looks as follows:

```
<?xml version="1.0" encoding="UTF-8"?>
<AuthenticateUser>
  <username>[username]</username>
  <password>[plain-text password]</password>
</AuthenticateUser>
```

In case of success, the webservice will return a session key, which can be used for other methods. In case of error, the webservice will return an error string.

```
<?xml version="1.0" encoding="UTF-8"?>
```

```
<AuthenticateUserResponse>
    <result>OK</result>
    <sessionkey>275000862</sessionkey>
    <apiversion>2.3.1</apiversion>
</AuthenticateUserResponse>
```

Logging out

Both digest and basic authentication methods use the same logout request. A logout request is sent to the /webservice file and looks as follows:

```
<?xml version="1.0" encoding="UTF-8"?>
<DeleteSessionKey>
  <sessionkey>[sessionkey]</sessionkey>
</DeleteSessionKey>
```

If the user has been logged out successfully, the webservice will return OK. An error string is returned otherwise.

```
<?xml version="1.0" encoding="UTF-8"?>
<DeleteSessionKeyResponse>
  <result>OK</result>
</DeleteSessionKeyResponse>
```

2.2 Camera view

Some commands like selectCamera and fillLayout require a cameraview parameter or comma separated list of cameraview parameters. A cameraview parameter consists of 6 name:value pairs, separated by a semicolon (';'), see table below. Only the cameraid is required. A cameraid value of 0 will blank the corresponding panel. The order of the pairs is arbitrary.

If the server id is omitted, the server this command is send to will be used. If one of the cutout parameters is ommited, a default value of 0 is used for the left and top parameter and a default value of 1000 is used for the right and bottom parameter.

Parameters

Version	Function	Type	Remarks		
1.10.1		[serverid]		[uid]	Optional unique server id. When omitted, uses the default server.
1.10.1		cameraid		[uid]	Required unique camera id. A value of zero (0) blanks the panel. (deviceid works similar)
1.11.1		left		[0..1000]	Optional left position in promille for image cutout.
1.11.1		top		[0..1000]	Optional top position in promille for image cutout.
1.11.1		right		[0..1000]	Optional right position in promille for image cutout.
1.11.1		bottom		[0..1000]	Optional bottom position in promille for image cutout.

Query Examples

Example of the most elementary camera view:
cameraid:1

Example with serverid, fields separated by a semicolon (';'):
cameraid:1;serverid:8889181777222111211

Complete example:
cameraid:8001;serverid:1234567812345001;left:10;top:20;right:30;bottom:40

Mixed example:
top:21;right:300;serverid:1234567812345002;left:10;cameraid:80;bottom:40

Example with some missing cutout parameters.
cameraid:8001;right:500;bottom:500

Example of a simple cameraview list, cameraviews separated by a comma (','):
cameraid:1,cameraid:2,cameraid:3,cameraid:4

Example of a more complex cameraview list,
(notice left and right for cameraid 3!):
cameraid:1;left=500;top=500,cameraid:2:right=500;top=500,cameraid:3;right=250;top=250;left=750;bottom=750,cameraid:4;left=100;top=100;right=400;bottom=400

Example of a complete cameraview list:
cameraid:81;serverid:1234567812345001;left:10;top:20;right:30;bottom:41, bottom:42;
cameraid:82;serverid:1234567812345002;left:12;top:22;right:32

Cutout

The image cutout parameters left, top, right and bottom are used to zoom in on the image. They select a part of the image that is displayed on the panel. The parameter values range from 0 up to 1000, representing a promile part of the image.

The aspect ratio is kept. Because the cutout parameters are promiles, the image aspect ratio is kept if the cutout aspect ratio is 1:1. So if the cutout width and height are the same the image aspect ratio is kept. If the cutout width and height are not equal, the largest of the two is taken.

If the left value is larger then the right value, these values are switched. The same goes for top and bottom. If the values are out of bounds, they are rounded to the nearest boundary.

Example Description

The following examples all start with an image showing the original image (yellow), the image cutout (blue) and the resulting part of the image (cross pattern). Hereafter follows a calculation of the resulting image cutout. The cutout parameters are checked to see if they exceed boundaries and if they are reversed (right before left, bottom before top). The aspect ratio of the image is kept, the result has the same aspect ratio as the image (and the cutout aspect ratio after checking it is 1:1).

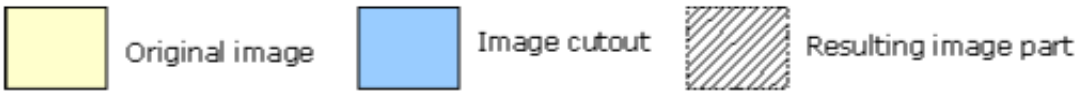
In these examples several parameters are supplied, either by the image or by the user. These parameters are in italic font. Some parameters are calculated and some use other parameters. The cutout parameters also show their corrected values after the '—->' arrow. All examples use this scheme:

Image parameters	
Width	400
Height	300
Aspect ratio	Width / Height

Cutout parameters	
Left	user —-> correction
Top	user —-> correction
Right	user —-> correction
Bottom	user —-> correction
Width	Right – Left —-> correction
Height	Bottom – Top —-> correction
Max	Maximum value of Width or Height
Aspect ratio	Width / Height —-> correction

Cutout parameters	
ZoomFactor	1000 / Max

Resulting image cutout	
Left	Cutout.Left * Image.Width / 1000
Top	Cutout.Right * Image.Height / 1000
Right	Cutout.Left + Cutout.Width
Bottom	Cutout.Top + Cutout.Height
Width	Image.Width / ZoomFactor
Height	Image.Height / ZoomFactor

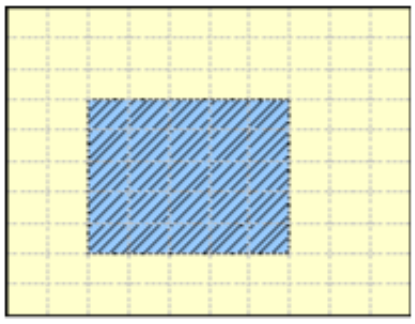


Legend of cutout example figures.

Cutout Examples

Example 1

Cutout with correct aspect ratio within borders.



Example 1

Image parameters	
Width	400
Height	300
Aspect ratio	1.3333333

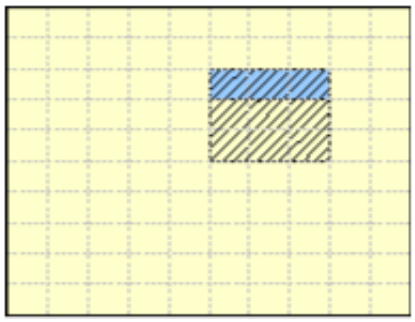
Cutout parameters	
Left	200 —> 200
Top	300 —> 300
Right	700 —> 700
Bottom	800 —> 800
Width	500 —> 500
Height	500 —> 500
Max	500

Cutout parameters	
Aspect ratio	1 —> 1
ZoomFactor	2

Resulting image cutout	
Left	80
Top	90
Right	200
Bottom	150
Width	280
Height	240
Aspect ratio	1.3333333

Example 2

Cutout with incorrect aspect ratio within borders.



Example 2

Image parameters	
Width	400
Height	300
Aspect ratio	1.3333333

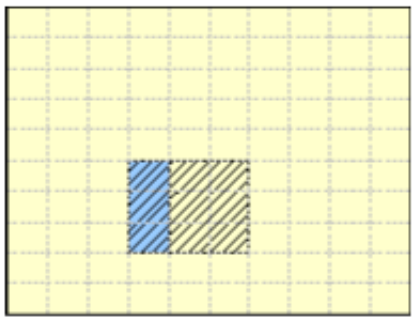
Cutout parameters	
Left	500 —-> 500
Top	200 —-> 200
Right	800 —-> 800
Bottom	300 —-> 500
Width	300 —-> 300
Height	100 —-> 300
Max	300
Aspect ratio	3 —> 1
ZoomFactor	3.3333333333333

Resulting image cutout	
Left	200
Top	60

Resulting image cutout	
Right	120
Bottom	90
Width	320
Height	150
Aspect ratio	1.3333333

Example 3

Cutout with incorrect aspect ratio within borders.



Example 3

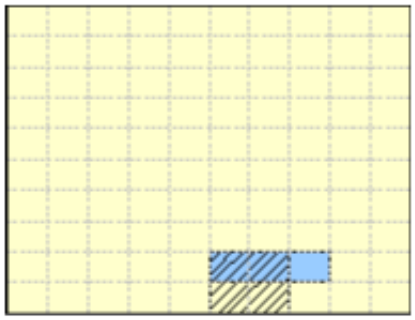
Image parameters	
Width	400
Height	300
Aspect ratio	1.3333333

Cutout parameters	
Left	300 —-> 300
Top	500 —-> 500
Right	400 —-> 600
Bottom	800 —-> 800
Width	100 —-> 100
Height	300 —-> 300
Max	300
Aspect ratio	0.333333333333 —> 1
ZoomFactor	3.3333333333333

Resulting image cutout	
Left	120
Top	150
Right	120
Bottom	90
Width	240
Height	240
Aspect ratio	1.3333333

Example 4

Cutout with incorrect aspect ratio where the resulting cutout would pass the bottom border.

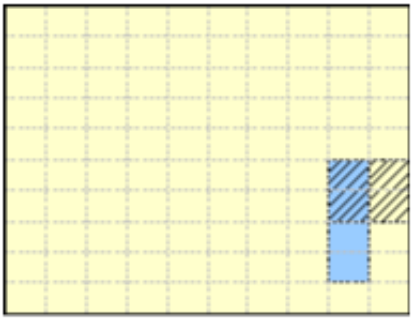


Example 4

Image parameters	
Width	400
Height	300
Aspect ratio	1.3333333
Cutout parameters	
Left	500 —-> 500
Top	800 —-> 800
Right	800 —-> 700
Bottom	900 —-> 1000
Width	300 —-> 200
Height	100 —-> 200
Max	200
Aspect ratio	3 —> 1
ZoomFactor	5
Resulting image cutout	
Left	200
Top	240
Right	80
Bottom	60
Width	280
Height	300
Aspect ratio	1.3333333

Example 5

Cutout with incorrect aspect ratio where the resulting cutout would pass the right border.



Example 5

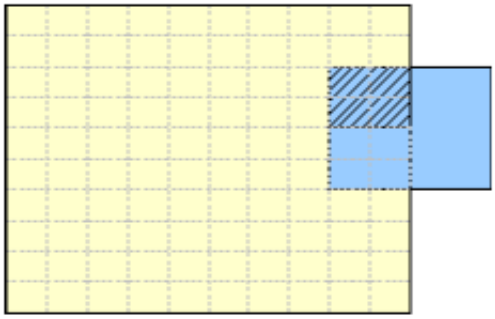
Image parameters	
Width	400
Height	300
Aspect ratio	1.3333333

Cutout parameters	
Left	800 —-> 800
Top	500 —-> 500
Right	900 —-> 1000
Bottom	900 —-> 700
Width	100 —-> 200
Height	400 —-> 200
Max	200
Aspect ratio	0.25 —> 1
ZoomFactor	5

Resulting image cutout	
Left	320
Top	150
Right	80
Bottom	60
Width	400
Height	210
Aspect ratio	1.3333333

Example 6

Cutout with correct aspect ratio that passes the right border.



Example 6

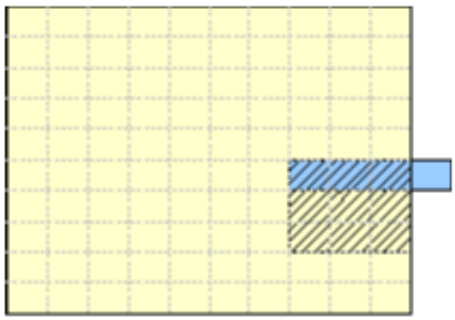
Image parameters	
Width	400
Height	300
Aspect ratio	1.3333333

Cutout parameters	
Left	800 —-> 800
Top	200 —-> 200
Right	1200 —-> 1000
Bottom	600 —-> 400
Width	400 —-> 200
Height	400 —-> 200
Max	200
Aspect ratio	1 —> 1
ZoomFactor	5

Resulting image cutout	
Left	320
Top	60
Right	80
Bottom	60
Width	400
Height	210
Aspect ratio	1.3333333

Example 7

Cutout with incorrect aspect ratio that passes the right border.



Example 7

Image parameters	
Width	400
Height	300
Aspect ratio	1.3333333

Cutout parameters	
Left	700 —-> 700
Top	500 —-> 500
Right	1100 —-> 1000
Bottom	600 —-> 800
Width	400 —-> 300
Height	100 —-> 300
Max	300
Aspect ratio	4 —> 1
ZoomFactor	3.3333333

Resulting image cutout	
Left	280
Top	150
Right	120
Bottom	90
Width	400
Height	240
Aspect ratio	1.3333333

Example 8

Cutout with correct aspect ratio that passes the bottom border.

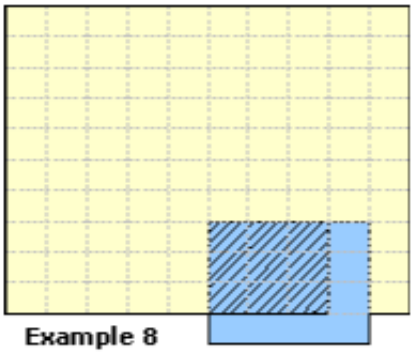


Image parameters	
Width	400
Height	300
Aspect ratio	1.3333333

Cutout parameters	
Left	500 —-> 500
Top	700 —-> 700
Right	900 —-> 800
Bottom	1100 —-> 1000
Width	400 —-> 300
Height	400 —-> 300
Max	300
Aspect ratio	1 —> 1
ZoomFactor	3.3333333

Resulting image cutout	
Left	200
Top	210
Right	120
Bottom	90
Width	320
Height	300
Aspect ratio	1.3333333

Example 9

Cutout with incorrect aspect ratio that passes the bottom border.

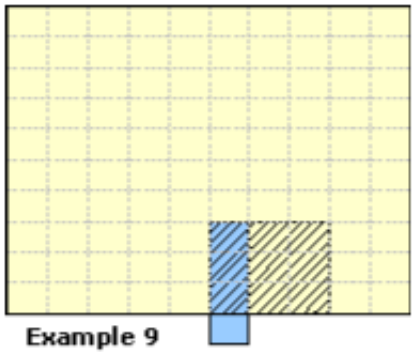


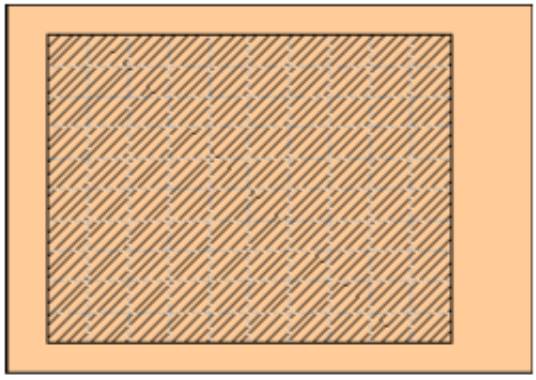
Image parameters	
Width	400
Height	300
Aspect ratio	1.3333333

Cutout parameters	
Left	500 —-> 500
Top	700 —-> 700
Right	600 —-> 800
Bottom	1100 —-> 1000
Width	100 —-> 300
Height	400 —-> 300
Max	300
Aspect ratio	0.25 —> 1
ZoomFactor	3.3333333

Resulting image cutout	
Left	200
Top	210
Right	120
Bottom	90
Width	320
Height	300
Aspect ratio	1.3333333

Example 10

Cutout with incorrect aspect ratio and reversed values that also passes all borders.



Example 10

Image parameters	
Width	400
Height	300
Aspect ratio	1.3333333

Cutout parameters	
Left	1200 —-> 0
Top	1100 —-> 0
Right	-100 —-> 1000
Bottom	-100 —-> 1000
Width	-1300 —-> 1000
Height	-1200 —-> 1000
Max	1000
Aspect ratio	1.0833333333 —> 1
ZoomFactor	1

Resulting image cutout	
Left	0
Top	0
Right	400
Bottom	300
Width	400
Height	300
Aspect ratio	1.3333333

2.3 Connection

A command is send as url by HTTP(s) (url encoded) as:
http(s)://server-ip[:port]/command?cameraid=[cameraid]& command=[command]&
The order of all arguments is arbitrary.

2.4 Important notes

- Xml tags can be supplied in arbitrary order. This can differ from the definitions and examples in this document.
- The nesting level of the xml tags and the functionality and parameters defined in this document will not change within the same major version. This only applies to official releases, not drafts.
- Between major version changes, functions and parameters can change. This is denoted in this documentation.
- Messages can be extended with tags, sub tags, also with different levels, and attributes without breaking the functionality of the command. This is denoted in this documentation.
- For debugging purpose messages can also be extended without notice.
- All xml tags are in lowercase, but the system will not be case sensitive about this.
- All http requests and commands are case insensitive.
- For the purpose of readability in this document requests are written with embedded spaces. In the real HTTP command they must be ommited.

2.5 Plugin

As of version 2.1.1 the HTTP interface and web server is implemented as a plugin. The 1.12 version (and some above) will still be available for a while. As a plugin, the web server runs in its own context, independent of the server. In this architecture, the web server doesn't need to run on a server, although this will be the default, but can run on any system attached to the video network. On large systems with multiple servers, it might be best to run the web server on a separate system. The plugin is provided as a DLL and is managed by the manager service. The manager is installed as service on every server by default.

Connecting to user manager

The web server plugin logs in the same way as the client to a central user manager. This user manager is often the same as the management server (the master server) to which the system managed servers (client servers) connect to. In case of a system with only one server, this server is also the management server and the user manager. After the web server has logged in, it obtains all rights available for that user. For the web server this basically includes the right the see certain camera images from certainservers.

One connection for all images

When the web server is logged into the user manager, it is connected to all available servers. It is now able to transmit images from all those servers. A client only has to connect to one web server to obtain video in stead of connecting to every server separately.

To obtain video from a specific server, the server id must be supplied by the command. As can be seen in the Changelog by remarks for the 2.1.1 version, a lot of commands now have serverid added to their parameter list. If the server id is ommited, the serverid from the management server is taken.

Timestamps

All timestamps from version 2.1.1 are in UTC format.

2.6 Redundancy

in the scheme for a redundant system interface, we assume that all elements, the management server, server and the client manager application, are implemented twice. The primary parts normally communicates with eachother. The secondary parts takes over when the primary fails. When the primary part comes back, it takes

Types of commands

For the further story it is important to distinguish between the two types of API commands.The first type are commands that retrieve data such as getDivaList, getAbsolutePosition etc. These commands perform no actions on the server and only request data. The second type are commands that can execute an action on the server. These are commands like setPreset, executeAction, selectCamera etc. These action commands can change the status of the server.

Primary server

The primary management server is the server who is normally active. This server is checked by the secondary (fail-over) server (connection J). The primary server can have connections to clients and managers. The server does not know in principle what kind of client is connected to it.

Secondary server

The secondary server is the fail-over server. It checks if the server(s) no longer runs (J) and takes over on failure. Then the secundary server makes sure that all of the clients are informed of this acquisition. The clients will then disconnect from the primary server (F and H) and connect to the secondary server (G and I).

Primary manager

The primary manager has a connection with the server (primary (F) or secondary (G)). This manager sends all actions to the server and sends all status information to (manager) clients (A and C). Multiple clients can send action and status commands to the manager. The manager performs them all.

Unique commands

If multiple clients are setup as redundant or failover Client, they can send the same commands to the manager (A and C). It can happen that the same action is send twice to the server. This can lead to unforeseen results. To prevent this, the commands can be provided with a unique ID by the UID parameter in the URL. This value can be anything as long as it is unique. Example:

command=selectLayout&layoutid=1&divaid=7015498111931795223&uid=AUniqueValue1234

The manager only performs the first action command. Second and subsequent action commands with the same UID will not be forwarded to the server. A normal response will be returned.

Secondary manager

The secondary manager works in failover mode. In this mode the manager connects to the primary manager (E) and asks for status updates of all handled messages. A status message is sent by the primary manager after finishing a command. Each incoming command of a client is queued by the secundary manager. If a

status message from the primary manager comes with the same UID as a command in the queue then this command is executed. This command does not sent any action servers but does give an answer and/or status information back. When commands stay too long in the queue the secundary manager will switch to normal mode. This manager then behaves more like a primary manager. Once the secundary manager starts to receive status messages from the primary manager it switches back to failover mode.

Primary client

The primary client has a normal connection to the primary manager (A). It can also be connected to the secondary manager (B). To prevent executing double action commands on the server a unique ID can be sent with each command. This UID prevents the secundary manager (in failover mode) to send actions to the server. All status messages are handled by both the primary and secondary manager. If the primary manager fails it will no longer give answers and there may be no connections anymore. Verifying if the manager is still working can be done by asking for a certain status, e.g. with getEvents.

Secondary client

The secondary client seeks to replace the primary client should it fail. The primary and secondary client must be controlled so that they send the same commands to the managers with the same UIDs. The secondary client can be omitted. When connected to multiple managers a client must send UID's with each message.

2.7 Results

XML results will be in the form of:

```
(Content-Type: "text/xml")
<result errorcode="[error code]">
  <description>
    Description for result code (0 = ok, no error)
  </description>
</result>
```

Optional tags can be present and the tag description can be ommited, depending the command issued (specifically if the result sends back information).
When a jpeg image is send, the result will be in the form of:
(Content-Type: “image/jpeg”)
(image data)

The Response section for every function describes the possible returning errorcodes.

All functions except those that return an image, will return an error code of 0 or greater in case of success and an error code less then 0 in case of failure. An error code of “0, Ok” means that the function is executed without error. An error code of “-18, General error” can be returned if an unhandled error occured. Consult the logging files for more explanation in that case.

2.8 RTSP Video

Live Video URL

The generic URL is formatted as follows:

rtsp://<serveraddress>/<serverid or description>?device=<deviceid>&<optional settings>

The figures illustrates how this information can be obtained from Sense Client.

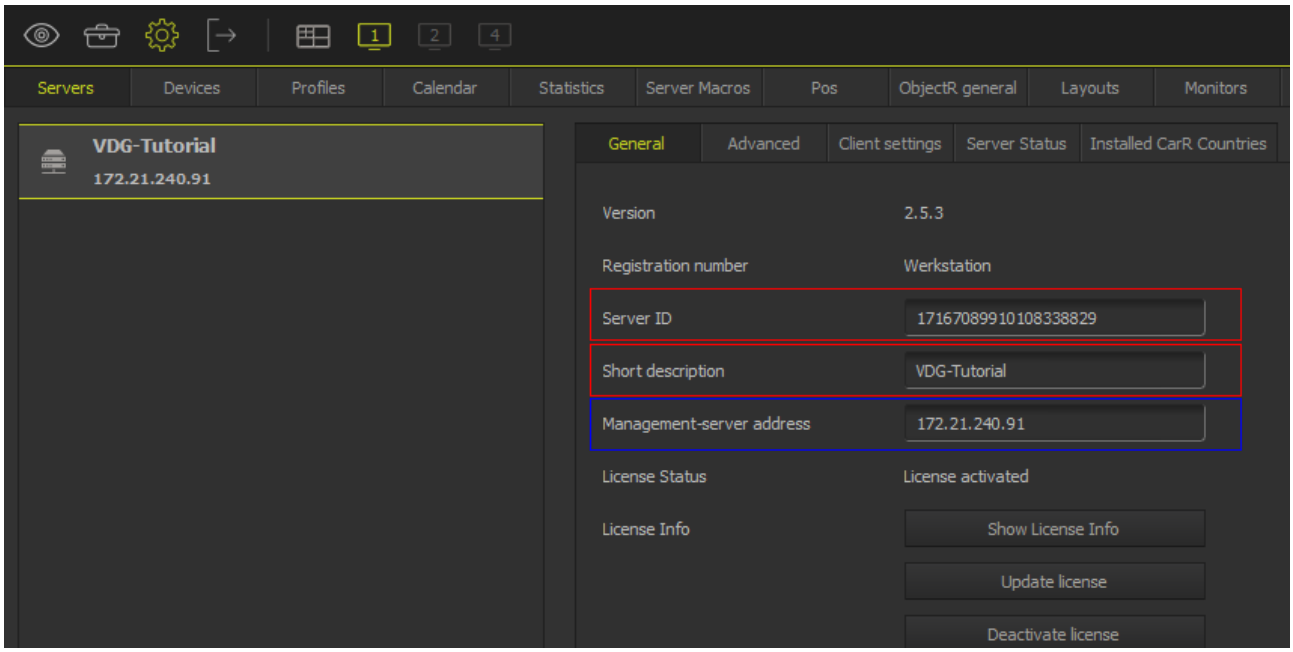


Figure 1: Server ID, Description and IP Address. (Server tab, Sense Client)

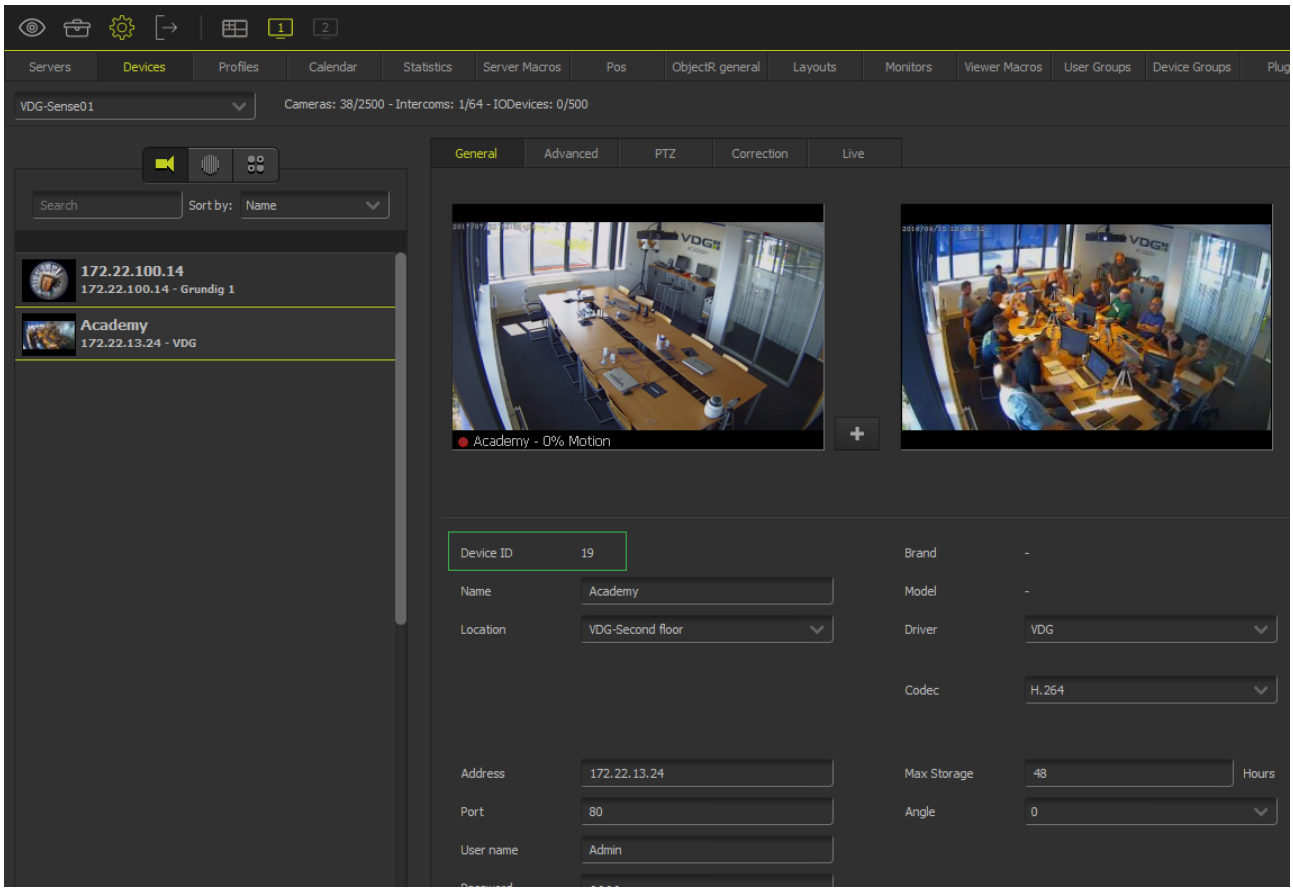


Figure 2: Device ID. (Devices tab, Sense Client)

Within the boundaries of the given example, both of the following URLs would be valid (since it supports both the server id and description):

- rtsp://172.21.240.91/VDG-Tutorial?device=19
- rtsp://172.21.240.91/17167089910108338829?device=19

The minimum request URL should always contain at least the server-address, server-ID and device-id. This will return the native video. Resolution, quality and codec are the same as the camera stream.

Video on-demand URL

In order to play recorded video, the timestamp argument can be added to the RTSP URL. This argument identifies from which point in time playback needs to commence. The date and time are formatted in a specific way:

YEAR (4 digits); MONTH (2 digits); DAY (2 digits); HOUR (2 digits); MINUTE (2 digits); SECOND (2 digits); MILLISECOND (3 digits, optional)

By this notations, the following 2 timestamps are formatted correctly:

- 20130226181722 (26-2-2013 18:17:22)
- 20130226181722455 (26-2-2013 18:17:22.455)

In addition, the following operators can be used combined with the argument:

- =
  Commences playback on exactly the given date and time.
- =>
  Commences playback on the first available data after the given date and time.
- =<
  Commences playback on the last available data before the given date and time.

Example

rtsp://172.21.240.91/VDG-Tutorial?device=3&timestamp=>20170226181722

The request above will play video from camera 3 from 26-2-2017 18:17:22 or later.

Note: Transcoding playback data is not supported!

Authentication

After initiating an RTSP stream, the server will require a username and password. These are identical to those set within the Sense Plugin Manager, please note that the Sense user needs to have sufficient permission for the camera which is requested. Both the username and password are case-sensitive. Figure 3 illustrate the login procedure within VLC media player.

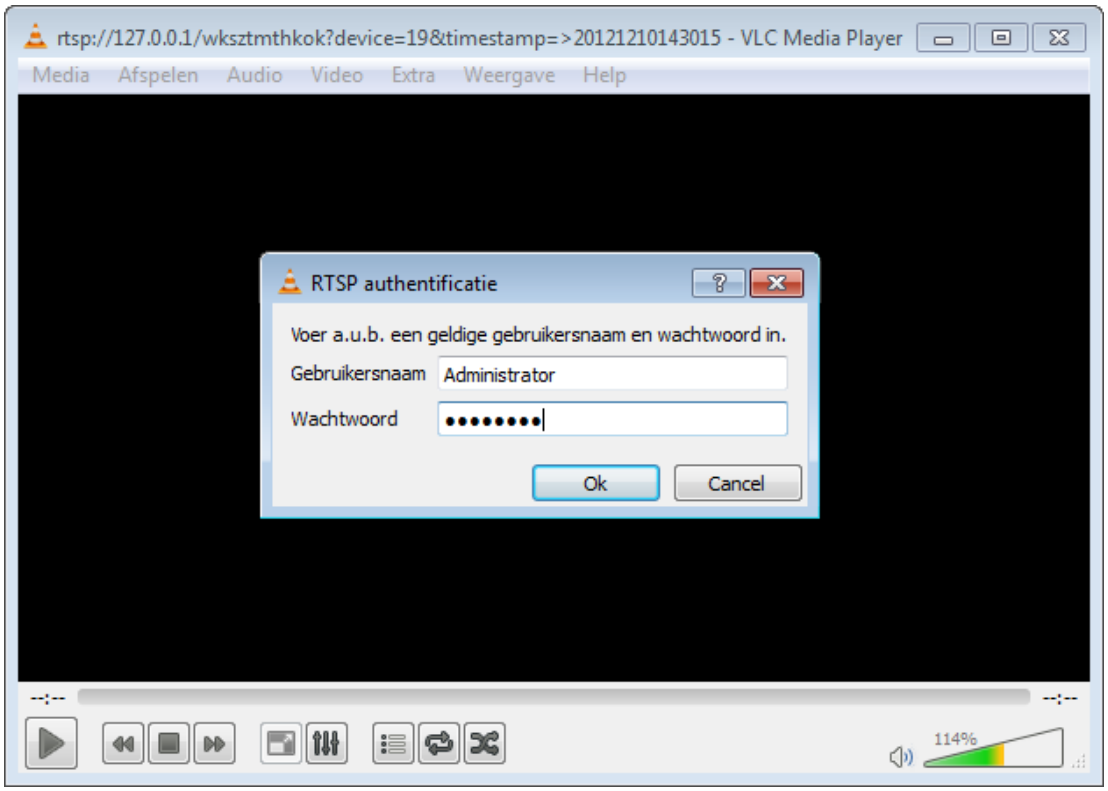


Figure 3: Logging in on the RTSP server.

Dual Streaming

By providing a width URI parameter, it's possible to make use of the dual streaming capabilities of a device.

Example

- Dual stream url:
  rtsp://<address>/<server-id>?device=<device-id>&width=320
  The width parameter is send to the server, the server determines the correct stream, based on the dual stream setting for the max panel width. No transcoding is performed.

Transcoding

The supplied video can be transcoded. In that case the video will always be supplied in JPEG format, and can be adjusted for the required height, width, quality and bandwidth, if specified. This can be accomplished by combining the transcode (do or do not transcode), width (videowidth) and height videoheight) arguments within the URL.

Note: Transcoding playback data is not supported

Whether to do or do not transcode is identified by the server using a 0 (do not) or 1 (do) value. The video's height and width are given in pixels.

URI	Type	Description
transcode	Boolean (0|1)	If set to 1, image will be transcoded
bandwidth	Integer (Kb/s)	Set the maximum bandwidth
width	Integer (pixel)	Video width in pixels
height	Integer (pixel)	Video height in pixels
quality	Integer (percentage)	Quality of the RTSP stream (default 70)
fittopanel	Boolean (0|1)	Add black bars if requested image height/width does not correspond with video resolution

Examples

- Transcode stream url:
  rtsp://<address>/<server-id>?device=<device-id>&width=320&height=240&transcode=1
  The width, height and transcode parameters are send to the server, the server determines the correct stream, based on the dual stream setting for the max panel width. The stream is transcoded to a JPEG stream with the requested width and height, but it keeps the original aspect-ratio.
- Transcoding over limited bandwidth url:
  rtsp://<address>/<server-id>?device=<device-id>&width=320&height=240&transcode=1&bandwidth=32
  The stream is transcoded to half the requested width and height, balanced over the limited bandwidth, and keeps the original aspect-ratio.
  The resulting frame-rate will differ and be irregular compared to the original frame-rate, due to the load-balancing.

2.9 Timestamp format

All timestamps have the format: yyyy-mm-dd hh:nn:ss.zzz

Where:

- yyyy stands for 4 digits year number
- mm stands for 1 or 2 digits month number starting at 1
- dd stands for 1 or 2 digits day number starting at 1
- hh stands for 1 or 2 digits hour number starting at 0
- nn stands for 1 or 2 digits minute number starting at 0
- ss stands for 1 or 2 digits second number starting at 0
- zzz stands for 3 digits millisecond number starting at 000

2 And 3 digits numbers may have leading zero's e.q.: '02' hour.

From version 2.1.1 (plugin), all timestamps are in UTC format.

2.10 Transcoding

Transcoding can be used to reduces the image quality and resolution before the image is send over the network. It is mainly used when connecting through a WAN connection with a low bandwidth. Depending on the BandWidth value and already send images, an image is marked or rejected for sending. When marked, the image is first decompressed and rescaled using the ImageWidth and ImageHeight parameter. Then it is compressed again using the ImageQuality parameter. This way the server will optimize the resulting image before sending it. For best performance and quality the ImageWidth and ImageHeight parameters must be the same as the displayed image.

When transcoding is enabled for the web client, the ImageWidth and ImageHeight parameters can be send along every getLiveImage request. If ommited, the default ImageWidth and ImageHeight values are taken.

The current settings for transcoding can be obtained with the command getServerOptions and set with the command setParameters. WebTranscodedImagesRequested returns 1 when transcoding is active, otherwise 0. WebDefaultImageWidth and WebDefaultImageHeight are used to determine the default ImageWidth and ImageHeight values for transcoding. When transcoding is not active, these values are not used. In that case, the complete image is send, even if it is going to be displayed much smaller.

Transcoding is seperately enabled for web clients. This allows the server to send transcoded images to web clients and normal images to NMS systems. Because transcoded images are send to all clients who requested a transcoded image, the lowest transcoding parameter values for all those clients are used. This could mean that even though you request an 1024 x 786 image, you receive an 400 x 300 images because another client requested that image size. Shortely after this other client stops requesting you will receive the image at your requested size.

2.11 Using this interface

The commands of this interface can be used to obtain information like video and events, to set some parameters and to display video on panels.

A complete VDG Sense concept consists of several systems like managers, servers, and clients, all linked together to store and display video and other related data. Servers used as viewer consists of serveral objects that build up (multi)layouts. A layout contains panels that are used to display video and data like events and controls. Every object in the server that is selectable or controllable has a unique id or index. This id or index must be used with the specified command. All id's can be obtained by examining the setup and by issueing the correct commands.

To start, it is important to know how the complete VDG Sense concept is setup and what the management server is. This server is used to send all commands to (except obtaining live video, historical video and historical video data) (From the plugin version 2.1.1 on all information can be retrieved from the management server). When only 1 server is used, this server is management server, user server, video server, administrator and viewer all together. In this case, there is no need to identify or select the server.

Most commands require a cameraid and optional a serverid. This information can be retrieved by examining the servers and by obtaining the cameralist with getCameraList. This will return a list of all cameras the system uses.

Video is displayed in panels that are displayed in layouts on monitors. Panels are addressed by their index. Layouts are addressed by their id. To get the list with layouts issue the command getLayoutList. When displaying video i.e. with selectCamera, the system looks on all viewers which one's have the required panel displayed on a monitor. All servers found will try to select the panel and set the video on it. This can fail if no server is found that shows the layout, the current logged in user on the server has no rights to view the video, no panel is found to display video on or there is no camera found.

The information required to select layouts and multilayouts can be viewed in the server and retrieved with getLayoutList and getMultiLayoutList. This information can be used to display layouts on monitors with selectLayout and selectMultiLayout.

After all this information is available it is possible to query the servers for video information and to control the servers to display video. Also other commands are available like enrolling a person for face recognition and all kind of dome control commands.

For more information send an email to support@vdgsecurity.com.

3 Functions

For the usage of these functions see: Usage. Some functions may require an API license

3.1 addClip

This call adds a request to create a new video clip (in mp4 format). The server will generate the clips one by one from the 'queue' (getClips). In order to check if the clip finished, the call getClips should be used to poll for the status. When the status is finished, the call downloadClip can be used to download the clip.

Clips are unique based on serverid, deviceid, startdate, enddate, imagewidth, and imageheight. If exactly the same request is done twice, the server will return the first clip object and won't start a new request.

The parameters 'imagewidth' and 'imageheight' resize/transcode the original video to the requested resolution. The time of creating a clip will increase when transcoding is required.

- If one of the parameters is omitted, it will resize according to the original ratio.
- If both parameter are sent with the request, it will add black bars to the bottom/top or sides and keeps the ratio of the video intact.
- If both parameters are omitted, it will export the original resolution.
- The width and height will be rounded to a power of 16.
- The minimum width and height is 128 pixels.
- The maximum width and height is the original resolution of the video stream.

Clips will be deleted based on their retention size and time. By default, clips will be removed after seven days or when the clip folder exceeds 1 GB. You can set this retention size and time in the native client under 'Advanced settings' in the 'Server' tab. To manually remove a clip from a server, use the call deleteClip.

Note: This function requires VDG Sense version 2.5.4 or higher

Request

```
command=addClip&serverid=[UID]&deviceId=[UID]&description=[String]&startdate=[timestamp]&enddate=[timestamp]&vca=[string]
```

Parameter	type	remarks	
serverid		[UID]	unique server id
deviceid		[UID]	unique device id
description		[string]	a user defined description
startdate		[timestamp]	startdate of event clip
enddate		[timestamp]	enddate of event clip
imagewidth		[number]	optional for transcoding
imageheight		[number]	optional for transcoding
vca		[none, alarms, objects, all]	include or exclude VCA metadata

Response

```
<result errorcode="0">
  <clip>
    <id>[UID]</id>
    <serverid>[UID]</serverid>
```

```
    <deviceid>[UID]</deviceid>
    <startdate>[timestamp]</startdate>
    <enddate>[timestamp]</enddate>
    <description>[string]</description>
    <filesize>[number]</filesize>
    <imagewidth>[number]</imagewidth>
    <imageheight>[number]</imageheight>
    <status description="[string]">[string]</status>
  </clip>
</result>
```

node	value	remarks	
/result			
@errorcode		[number]	See error codes below
/result/clip			
id		[UID]	unique id
serverid		[UID]	unique server id
deviceid		[UID]	unique device id
startdate		[timestamp]	Clips with a (local) timestamp >= startdate are returned.
enddate		[timestamp]	Clips with a (local) timestamp < = enddate are returned.
description		[string]	A user defined description
filesize		[number]	Size of the clip in bytes
imagewidth		[number]	Width of the clip in pixels
imageheight		[number]	Height of the clip in pixels
status		[string]	'pending', 'busy', 'finished', or 'failed'

error code	description	remarks	
0		Ok	Ok, no error
-2		No device	No device found for the supplied cameraid/deviceid.
-4		Invalid Timestamp.	Invalid datetime format. See Timestamp.
-17		No server	There is no server with the supplied id.
-18		General error	An undefined error occurred.
-33		No serverID	No serverID parameter given.
-34		No deviceID	No deviceID parameter given.
-35		No startdate	No startdate parameter given.
-36		No enddate	No enddate parameter given.
-37		enddate before startdate	Enddate parameter is before the startdate parameter.
-38		Clip time to long	Enddate is >5 minutes than startdate.

See Also

Changelog

- Sense 2.5.8 – Added vca parameter
- 2.7.2 – Added

3.2 addPerson

adds a person to the database. If an expiration date is given, the date will be checked and an error is returned if the date is not compatible.

Request

```
command=addPerson&firstname=[string]&lastname=[string]&address=[string]&postalco
de=[string]&city=[string]&phone=[string]&fax=[string]&mobile=[string]&licensepla
te=[string]&lastnameprefix=[string]&reference=[string]&notes=[string]&expiration
date=[string]&enabledexpirationdate=[boolean]
```

Response

```
<result errorcode="0">
  <person>
    <id>[uid]</id>
    <firstname>[string]</firstname>
    <lastname>[string]</lastname>
    <address>[string]</address>
    <postalcode>[string]</postalcode>
    <city>[string]</city>
    <phone>[string]</phone>
    <fax>[string]</fax>
    <mobile>[string]</mobile>
    <licenseplate>[string]</licenseplate>
    <lastnameprefix>[string]</lastnameprefix>
    <reference>[string]</reference>
    <notes>[string]</notes>
    <expirationdate>[string]</expirationdate>
    <enabledexpirationdate>[boolean]</enabledexpirationdate>
  </person>
</result>
```

node	value	remarks
/result		
@errorcode	[number]	See error codes below
/result/person		
id	[UID]	unique person id
firstname	[string]	firstname of the person
lastname	[string]	lastname of the person
address	[string]	address of the person
postalcode	[string]	postal code of the person
city	[string]	city of the person
phone	[string]	phone number of the person
fax	[string]	fax number of the person
mobile	[string]	mobile number of the person

node	value	remarks	
licenseplate		[string]	license plate of the person
notes		[string]	notes added to the person
expirationdate		[string]	the date the person will be removed from the database
enabledexpirationdate		[boolean]	flag to indicate the person can be expired

error code	description	remarks	
0		Ok	Ok, no error
-4		Invalid Timestamp	Invalid datetime format. See Timestamp.

Changelog

- 2.6.1 – added

3.3 deleteClip

Delete the clip of the given id. See addClip for the usage of the all the clip methods.

Note: This function requires VDG Sense version 2.5.4 or higher

Request

```
command=deleteClip&id=[UID]&serverid=[UID]
```

Parameter	type	remarks
id	[UID]	unique clip id
serverid	[UID]	unique server id

Response

error code	description	remarks
0	Ok	Ok, no error
-40	Clip not found	Clip with given ID does not exist

See Also

- downloadClip
- getClips
- addClip
- updateClip

Changelog

- 2.7.2 – Added

3.4 deletePerson

Removes a person from the database.

Request

```
command=deletePerson&amp;amp;id=[UID]
```

Response

error code	description	remarks
0	Ok	Ok, no error

Changelog

- 2.6.1 – added

3.5 downloadClip

Download and save the clip of the given id. See addClip for the usage of the all the clip methods.

Note: This function requires VDG Sense version 2.5.4 or higher

Request

```
command=downloadClip&id=[UID]&serverid=[UID]
```

Parameter	type	remarks
id	[UID]	unique clip id
serverid	[UID]	unique server id

Response

error code	description	remarks
	OK	Binary data
-40	Clip not found	Clip with given ID does not exist

See Also

- deleteClip
- getClips
- addClip
- updateClip

Changelog

- 2.7.2 – Added

3.6 executeAction

This commands executes a specific action on a system. If a divaid is supplied, the action is send to the system with that id if available. Dependend on the system, viewer or server, and the action type, also viewer of server, the action will be executed.

A list with available actions can be obtained with the getActionList command. This command returns Ok if the syntax of the request is correct. It does not return the results of the action itself. The action to be executed should be send as POST information in XML format. Depending on the type of the action, several parameters should be applied. See examples below.

From version 2.3.1 the name and parameters can also be aplied in the url. To supply a parameter in the url, use the same name as the 'name' attribute of the parameter tag as used in the post data xml.

Request

```
command=executeAction&actionname=[string]&[parameter]=[value]
```

POST Data

```
<!--?xml version="1.0" ?-->
 <action>
  <actionname>[string]</actionname>
  <parameters>
    <parameter name="[string]">Value</parameter>
    ...
  </parameters>
 </action>
```

POST Data Tags

node	value	remarks
/action		
actionname	[string]	unique name of the action
/action/parameters		
parameter	[string]	Value of the parameter
/action/parameters/parameter		
@name	[string]	unique name of the parameter.

error code	description	remarks
0	Ok	Ok, no error
-2	No device	No device found for the supplied cameraid/deviceid.

error code	description	remarks	
-17		No server	No server found for the supplied serverid.
-18		General error	An undefined error occured
-19		No dome	The device is not a dome device.
-22		Value is out of range	The supplied value is out of range.
-23		An empty value is not allowed	This command does not accept an empty value.
-24		Invalid action	The action is not a valid action.
-26		Action name does not exist	The supplied action name does not exist.
-27		Invalid action parameters	At least one action parameter does not exist.
-31		Invalid pretime	The supplied pretime is <0

See Also

- getActionList
- Actions
- Examples

Changelog

- 2.3.17 – parameter 'name' for action name is changed to 'actionname'
- 2.3.1 – name and parameters can also be aplied in the url
- 2.2.1 – added

Examples

All examples are included in the chapter Examples. Below is a list of currently supported actions. Use the command getActionList to determine if the system you interface with actually supports the action.

- Set device settings
- Change camera name
- Disable home preset
- Set home preset timeout
- Start tagging video (Video tagging)
- Stop tagging video (Video tagging)
- Execute macro

3.7 exportEvents

This command returns a Comma-Separated Values text (CSV) of events from the database.

All parameters are optional and can be used to filter the kind of event. Filters are and-ed together, meaning the event must match all filters. The parameter types is a comma separated list of types (strings). The event must match all events that match one of the types supplied are returned (after ending together with the other filters). If this parameter is omitted, all event types are returned.

The list can be limited by supplying starttime, endtime and/or limit. Refer to chapter Result Lists for an explanation of these parameters. If no limit argument is supplied, a default of 100 records is returned. The result is always sorted in ascending time.

Request

```
command=exportEvents&serverid=[UID]&deviceid=[UID]&types=[commalist]&starttime=[timestamp]&endtime=[timestamp]&limit=[number]
```

parameter	type	remarks
serverid	[UID]	filters events from a specific server.
cameraid	[UID]	filters events from a specific device.
types	[string]	types delimited with comma can be retrieved with getEventTypes
starttime	[timestamp]	events with a (local) timestamp >= startdate are returned.
endtime	[timestamp]	events with a (local) timestamp <= enddate are returned.
limit	[number]	determines the limit of the resulting event count.

Response

A Comma-Separated Values text where columns are separated with a semi-colon ';'. The first line contains the following columnnames:

- 'timestamp'
- 'device name'
- 'event name'
- 'value'

The following lines contain the event data.

See also

- Result Lists

Changelog

- 2.6.1 – Added

3.8 exportVideoTag

Export the given VideoTag as a native export.

The parameter Divaid (What is DIVA?) is optional, but if omitted, the request will export the VideoTag from all clients logged into the server.

Note: This function requires VDG Sense version 2.5.8 or higher

Request

```
command=exportVideoTag&TagID=[uid]&ExportTo=[path]
```

Parameter	type	remarks	
tagid		[UID]	unique id of video Tag
exportTo		[path]	local storage location for export
divaid		[UID]	unique id of client (Optional)

Response

```
<result errorcode="0">
 <descripion>Ok</description>
</result>
```

error code	description	remarks	
0		Ok	Ok, no error
-10		Not enough parameters	There are not enough parameters supplied for this function
-18		General error	An undefined error occured
-26		Videotag does not exist	Videotag does not exist or cannot be found
-28		Not a Diva client	There is no client with the supplied divaid
-52		Video tag is still active	Tag cannot be exported while active

See Also

- getVideoTagList
- Examples – Video tagging

Changelog

- 2.5.8 – Added

3.9 fillLayout

Displays a list of cameras in a layout. The cameras are displayed in the panels of the layout identified by layoutid. If more cameras then panels are supplied the surplus of cameras is not displayed. If less cameras then panels are supplied the surplus of panels is not changed. To blank these panels, a cameraid of 0 must be supplied.

The system (set of servers and viewers) searches all viewers to find where the layout is displayed. If one or more displayed layouts are found, the panels on this layout, starting with panelindex 0, are used to display the video. Only panels that are set to display video are used. Other panels, like event panels, are skipped.

If divaid is given, the command will only by send to the viewer with that id.

Request

```
command=fillLayout&layoutid=[UID]&cameraviews=[camviews]&divaid=[UID]
```

parameter	type	remarks
layoutid	[UID]	Unique layout id.
cameraviews	[camviews]	Comma separated list of camera views.
divaid	[UID]	unique id.

Response

error code	description	remarks
0	Ok	Ok, no error

See Also

- getCameraList
- getLayoutList
- selectCamera
- Camviews type

Changelog

- 2.3.1 – added divaid.
- 2.1.2 – Fixed dispatching to viewers.
- 1.11.1 – CameraView extended with left, top, right and bottom parameters.
- 1.10.1 – Added.

3.10 genericEvent

The genericEvent signal with the supplied value parameter is used in the macro mechanism to trigger macros.

This signal is also stored in the database. It can be obtained with the command getEvents. Special care must be taken when including xml specific characters. These are '&', '<' en '>'. The ampersand ('&') character can not be sent by this command because it will be part of the URL. See next paragraph. The less-then ('<') and greater-then ('>') character cannot be part of the return xml message in getEvents. These characters are converted to '<' and '>' before they are stored in the database.

As the value of this event is in the query part of an URL, the ampersand character ('&') cannot be part of it. It must be URL Encoded. See i.e. http://www.w3schools.com/TAGS/ref_urlencode.asp. The best practive is to URL Encode the complete value.

This command is only available if the option XML is available.

Request

```
command=genericEvent&amp;amp;value=[string]&amp;amp;serverid=[UID]&amp;amp;deviceid=[UID]
```

Response

parameter	type	remarks	
value		[string]	The value used for the generic event.
serverid		[UID]	unique server id
deviceid		[UID]	unique device id.

error code	description	remarks	
0		Ok	Ok, no error
-1		Not implemented.	Function not implemented.
-2		No device	No device found for the supplied cameraid/deviceid.
-17		No server	There's no server with the supplied id.
-53		Device connection lost	Device has no connection.

Changelog

- 2.2.1 – Handling of xml and html specific characters.
- 1.12.1 – Added serverid and deviceid.
- 1.11.1 – Added.

3.11 getActionList

Retrieve a list of actions that can be executed with the command executeAction.

Request

```
command=getActionList
```

Response

```
<result errorcode="0">
  <actions count="[number]">
    <action>
      <name>[string]</name>
      <description>[string]</description>
      <type>[string]</type>
      <parameters count="[number]">
        <parameter>
          <name>[string]</name>
          <description>[string]</description>
          <type>[string]</type>
          <required>[boolean]</required>
          <min>[string]</min>
          <max>[string]</max>
          <default>[string]</default>
        </parameter>
        ...
      </parameters>
    </action>
    ...
  </actions>
</result>
```

node	value	remarks
/result		
@errorcode	[number]	See error codes below
/result/actions		
@count	[number]	The number of actions in this list.
/result/actions/action		
name	[string]	unique name of the action.
description	[string]	a description of the action.
type	[string]	if it is a 'viewer' or 'server' action.
/result/actions/action/parameters		
@count	[number]	the number of parameters in this list
/result/actions/action/parameters/parameter		
name	[string]	unique name of the parameter.

node	value	remarks	
description		[string]	a description of the parameter.
type		[string]	the type of the parameter.
required		[boolean]	[0|1|false|true|yes|no] whether this parameter is required.
min		[string]	optional, the minimum value for this parameter.
max		[string]	optional, the maximum value for this parameter.
default		[string]	optional, the default value for this parameter.

error code	description	remarks	
0		Ok	Ok, no error
-18		General error	An undefined error occured

See Also

- Actions
- executeAction

Changelog

- 2.2.1 – added

3.12 getCameraList

Removed as of version 2
Deprecated as of 1.11.3, use getDeviceList instead.

Obtains a list of all cameras available in the network.

Request

```
command=getCameraList
```

Response

```
[string]
...
```

node	value	remarks
/result		
@errorcode	[number]	See error codes below
/result/cameras		
@count	[number]	The number of cameras
/result/cameras/camera		
camera	[string]	the user defined camera name.
id	[UID]	unique camera id

Changelog

- 2.1.1 – Removed
- 1.11.3 – Deprecated, use getDeviceList instead.
- 1.10.3 – Reverted the changes from 1.10.1 to maintain backwards compatibility. Use getDeviceList to obtain more information.
- 1.10.1 – Replaced attribute id with subtag cameraid, added subtag serverid, replaced value of tag camera with subtag name, added subtag response.
- 1.3.7 -Renamed attribute cameraCount to count.
- 1.3.5 – Moved count attribute from tag result to tag cameras
- 1.3.1 – Added

3.13 getClips

Obtains a list of all created or pending clips. See addClip for the usage of the all the clip methods.

Note: This function requires VDG Sense version 2.5.4 or higher

Request

```
command=getClips&serverID=[UID]
```

Parameter	type	remarks
serverID	[UID]	unique server id (Optional)

Response

```
<result errorcode="0">
  <clips count="[number]">
    <clip>
      <id>[UID]</id>
      <serverid>[UID]</serverid>
      <deviceid>[UID]</deviceid>
      <startdate>[timestamp]</startdate>
      <enddate>[timestamp]</enddate>
      <description>[string]</description>
      <filesize>[number]</filesize>
      <imagewidth>[number]</imagewidth>
      <imageheight>[number]</imageheight>
      <status description="[string]">[string]</status>
    </clip>
  </clips>
</result>
```

node	value	remarks
/result		
@errorcode	[number]	See error codes below
/result/clips		
@count	[number]	The number of video clips.
/result/clips/clip		
id	[UID]	unique id
serverid	[UID]	unique server id
deviceid	[UID]	unique device id
startdate	[timestamp]	Clips with a (local) timestamp >= startdate are returned.
enddate	[timestamp]	Clips with a (local) timestamp < = enddate are returned.
description	[string]	A user defined description

node	value	remarks	
filesize		[number]	Size of the clip in bytes
imagewidth		[number]	Width of the clip in pixels
imageheight		[number]	Height of the clip in pixels
status		[string]	'pending', 'busy', 'finished', or 'failed'

error code	description	remarks	
0		Ok	Ok, no error.
-17		No server	There is no server with the supplied id.
-18		General error	An undefined error occurred.

See Also

- downloadClip
- addClip
- deleteClip
- updateClip

Changelog

- 2.7.2 – Added

3.14 getDeviceList

Obtains a list of all devices (cameras) available in the network.

Request

```
command=getDeviceList
```

Response

```
<result errorcode="0">
  <devices count="[number]">
    <device>
      <name>[string]</name>
      <serverid>[uid]</serverid>
      <deviceid>[uid]</deviceid>
      <status>[string] </status>
      <type>[string] </type>
      <brand>[string] </brand>
      <location>[string] </location>
      <ptztype>[string]</ptztype>
      <ipaddress>[ip-address]</ipaddress>
      <port>[number]</port>
    </device>
  </devices>
</result>
```

node	value	remarks
/result		
@errorcode	[number]	See error codes below
/result/devices		
@count	[number]	number of devices
/result/devices/device		
name	[string]	the name of the device
serverid	[UID]	unique server id
deviceid	[UID]	unique camera id
status	[string]	status of the device, see Camera Status
type	[string]	type of the device, i.e. "camera"
brand	[string]	brand of the device, i.e. "VDG"
location	[string]	location of the device
ptztype	[string]	ptz type of the device, i.e. "VDG"
ipaddress	[ip-address]	The ip adress of the device
port	[number]	the port of the device

error code	description	remarks

error code	description	remarks
0	Ok	Ok, no error

See Also

- Camera Status

Changelog

- 2.7.2 – Added ipaddress and port nodes
- 2.3.18 – Device list is alphabetically sorted by device name
- 1.10.1 – Added

3.15 getDivaList

Obtains a list of all servers and viewers connected to the server this request is send to.

Request

```
command=getDivaList
```

Response

```
<result errorcode="0">
  <divas count="[number]">
    <diva>
      <divaid>[UID]</divaid>
      <name>[string]</name>
      <type>[string]</type>
      <options>[string]</options>
      <status>[string]</status>
      <address>[number]</address>
    </diva>
    ...
  </divas>
 </result>
```

node	value	remarks
/result		
@errorcode	[number]	See error codes below
/result/divas		
@count	[number]	The number of servers and viewers.
/result/divas/diva		
divaid	[UID]	The unique id
name	[string]	The name or description
type	[string]	Returns 'Diva server' or 'Diva client'.
options	[string]	Comma separated list of options for this servers or viewers, like CarR, FaceR, XML etc.
status	[string]	Status of the server. Usually empty
address	[ip-address]	The ip address of the server

error code	description	remarks
0	Ok	Ok, no error

See Also

- selectLayout
- What is DIVA?

Changelog

- 2.3.1 – added name
- 1.10.2 – added

3.16 getEvents

This command returns a list of specific events for specific cameras. The parameter startid is required to return a list of events. If omitted, or startid is zero or negative, an empty list will be returned with the lastEventId value. The lastEventId value can be used to query for new events.

All other arguments are optional and can be used to filter the kind of event. Filters are and-ed together, meaning the event must match all filters. The argument types is a comma separated list of types (strings) the event must match All events that match one of the types supplied are returned (after ending together with the other filters). If this parameter is omitted, all event types are returned.

The results are always sorted in ascending time. Because of the internal working of the system, it could happen that the order of eventid's differs from the order of eventtime's. For example, the result list, ordered by eventtime, could have an eventid sequence of 1, 2, 3, 5, 4, 6.

The result will also give the last event id back. To get only the last event id, issue this command with a zero or negative start id.

Every event in the eventlist has several result fields. Refer to the Event result fields table for a list of all result fields. The value of all event result fields depends on the software version and setup.

The list can be limited by supplying starttime, endtime and/or limit. Refer to chapter Result list. for an explanation of these parameters.

The parameters deviceid and serverid can be used to get events from a specific device (from a specific server).

Request

```
command=getEvents&startid=[id]&starttime=[timestamp]&endtime=[timestamp]&types=[string]&limit=[number]&deviceid=[uid]&serverid=[uid]
```

parameter	type	remarks
startid	[id]	events with an id > startid are returned.
starttime	[timestamp]	events with a (local) timestamp >= startdate are returned.
endtime	[timestamp]	events with a (local) timestamp <= enddate are returned.
types	[string]	comma separated list of types the event must match, i.e. Alarm,SceneR.
limit	[number]	determines the limit of the resulting event count.
deviceid	[UID]	filters events from a specific device.
serverid	[UID]	filters events from a specific server.

CarRLicensePlateFound only

parameter	type	remarks
licenseplate	[string]	events with the specified license plate are returned (Case sensitive)
licenseplatedistance	[number]	the number of characters that should match with the entered licenseplate. (0 = all chars should match; 1 = one char can differ, 2 = two chars can differ etc.)

Response

```
<result errorcode="0">
  <events count="[number]" lasteventid="[UID]">
    <event>
      <eventid></eventid>
      <eventtype></eventtype>
      <eventdescription></eventdescription>
      <eventvalue></eventvalue>
      <eventtime></eventtime>
      <serverid></serverid>
      <serverdescription></serverdescription>
      <deviceid></deviceid>
      <devicedescription></devicedescription>
    </event>
    ...
  </events>
 </result>
```

node	value	remarks
/result		
@errorcode	[number]	See error codes below
/result/events		
@count	[number]	The number of events.
lastEventId	[UID]	id of the last event.
/result/events/event		
eventid		id of event.
eventype		type of event. (Available in API version < 2.7.1)
eventdescription		description of event.
eventvalue		value of event.
eventtime		Time stamp of event in UTC time.
serverid		Id of server where the event occurred.
serverdescription		Description of the server
deviceid		Id of device which triggered the event.
devicedescription		Description of the device

error code	description	remarks
0	Ok	Ok, no error
-2	No device	No device found for the supplied camareaid/deviceid
-4	Invalid timestamp	Invalid datetime format. See Timestamp.

See Also

- Timestamp format.
- Event Types
- Event Result Fields
- Result list
- getEventTypes
- streamEvents

Changelog

2.7.2	Added licenseplate and licenseplatedistance parameters for CarRLicensePlateFound events.
2.1.1	Times are in UTC times (unless CompatibilityMode1 is activated)
1.12.3	Documented if timestamps are local or UTC timestamps.1.11.3 Strict checking on starttime and endtime parameter format. Returns -4 on error
1.11.1	Corrected the attribute name for time in the documentation. It must be types
1.11.1	Added cameraid and serverid filter.
1.10.2	Allows negative limit and endtime before starttime to reverse the ordering of the result list. Corrected the attributes names in the documentation for startdate and enddate, they are starttime and endtime
1.10.1	Changed default limit from 100 to 25.
1.4.1	Added parameters type and limit.
1.3.9	Fixed arguments. They are now startid, startdate and starttime. Changed limit from 1000 to 100.
1.3.7	Moved attributed eventCount and lastEventId to tag events Renamed eventCount to count.
1.3.3	Limit to 1000 result records
1.3.2	Added count as attribute to result tag.
1.3.1	Added

Example

Retrieve the last evenid..

To get the latest event id ommit the startid or give it a zero or negative value.
Command url:

```
command=getEvents&startid=0
```

Result:

```
<result errorcode="0">
  <events count="0" lasteventid="39993299">
  </events>
 </result>
```

3.17 getEventTypes

Obtains a list of all supported event types. The lists consists of name, description and level pairs. The name can be used in getEvents.

The usability level is only to get a global indication of how often the event will occur. A value of zero will indicate a normal, most common, event. A higher number indicates a less occurring event, which possibly only occurs in special circumstances.

Request

```
command=getEventTypes
```

Response

```
<result errorcode="0">
  <eventtypes count="[number]">
    <eventtype>
      <name>[string]</name>
      <description>[string]</description>
      <level>[number]</level>
    </eventtype>
    ...
  </eventtypes>
 </result>
```

node	value	remarks
/result		
@errorcode	[number]	See error codes below
/result/eventtypes		
@count	[number]	The number of event types.
/result/eventtypes/eventtype		
name	[string]	The name, as used in getEvents.
description	[string]	A system defined description of the event type..
level	[number]	A usability level number, 0 = normal, 1 = system and user, 2 = other

See Also

- Event Types
- getEvents.

Changelog

- 1.10.2 – added

3.18 getHardwareStatus

Obtains the single status of the hardware of all servers that are linked together (by their management address). The status consists of three levels: ok, warning and error. If the status is warning or error than an additional description tag is send explaining the situation in user readable language.

This function requires the System Status plugin to be enabled. For more information about configuring the System status plugin, click here. If System Status plugin is not running, an errorcode = "0" will be given.

Request

```
command=getHardwareStatus
```

Response

```
<result errorcode="0">
  <hardwarestatus>
    <status>[Ok|Warning|Error]</status>
    <description>[string]</description>
  </hardwarestatus>
 </result>
```

node	value	remarks
/result		
@errorcode	[number]	See error codes below
/result/hardwarestatus		
status	[Ok|Warning|Error]	The status level.
description	[string]	Optional description of the status level.

error code	description	remarks
0	Ok	Ok, no error

Results

If ok, the following will be returned:

```
<hardwarestatus>
  <status>Ok</status>
</hardwarestatus>
```

If failover is active, the following will be returned:

```
<hardwarestatus>
  <status>Error</status>
  <description>failover server is active</description>
</hardwarestatus>
```

In case of a Raid failure, the following will be returned:

```
<hardwarestatus>
  <status>Error</status>
 <description>[raid-status]</description>
</hardwarestatus>
```

[Raid-status] is a textual representation of the raid status. It is different for AMCC and LSI raid systems.

AMCC status

AMCC error [raid-status] messages have the following form:

```
AMCC: Server <ip of server>, RAID unit <id of unit>: <status of unit>
AMCC: Server <ip of server>, RAID drive <id of unit>: <status of drive>
```

Where status of a unit can be:

- OK
- Verifying
- Initializing
- Degraded
- Rebuilding
- Recovering
- Migrating
- Inoperable
- Unknown

And where status of drive can be

- OK
- Offline
- Offline JBOD
- Uncoverted DCB
- Unsupported DCB
- DCB Data check
- DCB Orphan
- DCB Read failure

- DCB Read timeout
- Unsupported
- Unknown

Note that not every status generates an error!
These values are taken from the official AMCC specifications

LSI status

LSI error [raid-status] messages have the following form:

```
LSI: Server <ip of server>, RAID unit <id of unit>: <status of unit>
LSI: <status of raid>
```

Where status of raid can be:

- degraded
- unconfigured-good
- unconfigured-bad
- hot-spare
- offline
- failed
- rebuild
- online
- system
- UNCONFIGURED-SHIELDED
- HOTSPARE-SHIELDED
- CONFIGURED-SHIELDED
- rebuilding
- no connection
- invalid response

These values are taken from the official LSI MIB specifications

Example

A degraded AMCC unit will generate the following response:

```
<hardwarestatus>
   <status>Error</status>
   <description>AMCC: Server 10.0.0.10, RAID unit 2: Degraded</description>
</hardwarestatus>
```

Only the last unit or drive that has a failure is reported in the XML response. If an error occurs the systems raid configuration software should be consulted for more information a repair.

See also

- Knowledge base: Configuring system status plugin

Changelog

- 2.4.2 – added

3.19 getHistoricalImage

This command is used to obtain the historical image. Parameter time is used to reference the image timestamp. Parameter type is used to tell the system what image to obtain. If this parameter is omitted or as an undefined value, the system assumes the image at the specified time is requested. If the image is not found, an error result is returned.

Request

```
command=getHistoricalImage&serverid=[UID]&cameraid=[UID]&time=[timestamp]&type=[type]
```

parameter	type	remarks	
serverid		[UID]	unique server id.
cameraid		[UID]	unique camera id
time		[timestamp]	UTC timestamp of image to look for (format: yyyy-mm-dd hh:nn:ss.zzz)
type		[type]	functionality of time parameters:
		before	returns the first image before the time given.
		atorbefore	returns the image at or the first image before the time given.
		at	(default) returns the image exactly at the time given.
		atorafter	returns the image at or the first image before the time given.
		after	returns the first image after the time given.
		keyframe	returns the corresponding keyframe (requires 2.5.4 or higher)

Response

```
jpeg image (or result in case of error)
```

error code	description	remarks	
0		Ok	Ok, no error
-2		No device	No device found for the supplied cameraid/deviceid.
-4		Invalid timestamp	Invalid datetime format. See Timestamp
-11		No image	There is no image found to return.

See Also

- Timestamp format

Changelog

- 2.1.2 – Fixed leap year bug, which returned a historical image one day in the future (which results in no image if an image of the current day was requested).
- 2.1.1 – Added serverid.
- 1.12.3 – Documented if timestamps are local or UTC timestamps.
- 1.11.3 – Strict checking on starttime and endtime parameter format. Returns -4 on error (see Result Codes).
- 1.3.6 – Added

3.20 getHistoricalImageData

This command is used to obtain the data concerning a historical image. Parameter time is used to reference the image. Realtime will be set at the time of the first image at or before the supplied time parameter. This function is to be used after a call to getHistoricalImage to retrieve data from the returned image. The time parameter must be exactly the same for both calls. If the time parameter doesn't match, their will be an error returned. If no previous or next image is found, they will not be present in the response.

Request

```
command=getHistoricalImageData&serverid=[UID]&cameraid=[UID]&time=[timestamp]
```

parameter	type	remarks
serverid	[UID]	unique server id.
cameraid	[UID]	unique camera id
time	[timestamp]	UTC date and time of image to get data for.

Response

```
<result errorcode="0">
  <previousimage>
    <time>[timestamp]</time>
  </previousimage>
  <realimage>
    <time>[timestamp]</time>
  </realimage>
  <nextimage>
    <time>[timestamp]</time>
  </nextimage>
 </result>
```

node	value	remarks
/result		
@errorcode	[number]	See error codes below
/result/previousimage		
time	[timestamp]	local timestamp of the previous image
/result/realimage		
time	[timestamp]	local timestamp of the current image
/result/nextimage		
time	[timestamp]	local timestamp of the next image

error code	description	remarks

error code	description	remarks	
0		Ok	Ok, no error
-2		No device	No device found for the supplied camaraid/deviceid
-4		Invalid timestamp	Invalid datetime format. See Timestamp
-6		Error historical Data	Error retrieving historical image data

See Also

- Timestamp format

Changelog

- 2.1.1. – Added serverid
- 1.12.3 – Documented if timestamps are local or UTC timestamps.
- 1.11.3 – Strict checking on starttime and endtime parameter format. Returns -4 on error (see Result Codes).
- 1.4.1. – Added

3.21 getImageSequence

This command is used to obtain a list of timestamps of recorded images between given dates. The result is a limited list starting with records from starttime to endtime. Optional filters (e.g. motion) can be included in later releases. A maximum of 100 records is returned by default if limit is not supplied.

As of version 1.10.2, the endtime can lie before the starttime and the limit can be negative. Refer to chapter Result Lists for an explanation of these parameters.

Request

```
command=getImageSequence&serverid=[UID]&cameraid=[UID]&starttime=[timestamp]&end
time=[timestamp]&limit=[number]
```

parameter	type	remarks
serverid	[UID]	unique server id.
cameraid	[UID]	unique camera id
starttime	[timestamp]	images at or past this timestamp are returned
endtime	[timestamp]	images before or at this timestamp are returned
limit	[number]	determines the limit of the resulting record count

Response

```
<result errorcode="0">
  <imagesequence count="[number]">
    <img>
      <time>[timestamp]</time>

    ...
  </imagesequence>
 </result>
```

node	value	remarks
/result		
@errorcode	[number]	See error codes below
/result/imagesequence		
@count	[number]	The number of images.
/result/imagesequence/image		
time	[timestamp]	local time of this image

error code	description	remarks
0	Ok	Ok, no error

error code	description	remarks
-2	No device	No device found for the supplied cameraid/deviceid.
-4	Invalid timestamp	Invalid datetime format. See Timestamp.

See Also

- Result Lists
- Timestamp format

Changelog

- 2.1.1 – Added serverid
- 1.12.3 – Documented if timestamps are local or UTC timestamps
- 1.11.3 – Strict checking on starttime and endtime parameter format. Returns -4 on error (see Result Codes).
- 1.11.3 – Default values are taken for starttime (earliest possible time) and endtime (current time).
- 1.10.2 – Allows negative limit and endtime before starttime to reverse the ordering of the result list.
- 1.3.9 – Limited the returning record count to 100 records by default. Can be adjusted by parameter limit.
- 1.3.6 – Added.

3.22 getLayoutList

Obtains a list of all layouts. Layouts are used to display video on.

Request

```
command=getLayoutList
```

Response

```
<result errorcode="0">
  <layouts count="[number]">
    <layout>
      <layoutid>[UID]</layoutid>
      <name>[string]</name>
      <panels count="[number]">
        <panel>
          <panelid>[UID]</panelid>
          <name>[string]</name>
          <deviceid>[UID]</deviceid>
          <serverid>[UID]</serverid>
          <type>[string]</type>
          <top>[number]</top>
          <left>[number]</left>
          <width>[number]</width>
          <height>[number]</height>
        </panel>
        ...
      </panels>
    </layout>
    ...
  </layouts>
 </result>
```

node	value	remarks
/result		
@errorcode	[number]	See error codes below
/result/layouts		
@count	[number]	the number of layouts
/result/layouts/layout		
layoutid	[UID]	unique id of the layout
name	[string]	the user defined name of the layout
/result/layouts/layout/panels		
count	[number]	the number of panels
/result/layouts/layout/panels/panel		
panelid	[UID]	unique id of the panel

node	value	remarks	
name		[string]	the user defined name of the panel
deviceid		[UID]	unique id of the device assinged to the panel (0 is no device assigned)
serverid		[UID]	unique id of the server of the device assigned to the panel (0 is no device is assigned).
type		[string]	description of the type of the panel
top		[number]	top coordinate of the position of the panel
left		[number]	left coordinate of the position of the panel
width		[number]	width of the panel
height		[number]	height of the panel

error code	description	remarks
0	Ok	Ok, no error

See Also

- selectCamera
- fillLayout
- getLayoutStatus

Changelog

- 2.3.1 – Changed panels tag from count to subtag with more information
- 1.10.1 – Added

3.23 getLayoutStatus

Obtains a list with the current layout status for the requested server. This list includes the monitors, the name of the layout that is selected on the monitor, and the devices that are visible on Live or Playback panels.

Request

Note: divaid is optional.

```
command=getLayoutStatus&divaid=[UID]
```

Response

```
<result errorcode="0">
  <layoutstatus>
    <divas count="[number]">
      <diva>
        <divaid>[UID]</divaid>
        <name>[string]</name>
        <addresses>[number]</addresses>
        <monitors count="[number]">
          <monitor>
            <index>[number]</index>
            <monitorid>[UID]</monitorid>
            <name>[string]</name>
            <active>[boolean]</active>
            <layout>
              <layoutid>[UID]</layoutid>
              <name>[string]</name>
              <panels count="[number]">
                <panel>
                  <panelid>[UID]</panelid>
                  <name>[string]</name>
                  <serverid>[UID]</serverid>
                  <deviceid>[UID]</deviceid>
                  <status>
                    <video>[string]</video>
                  </status>
                </panel>
                ...
              </panels>
            </layout>
          </monitor>
          ...
        </monitors>
      </diva>
      ...
    </divas>
```

```
 </layoutstatus>
</result>
```

node	value	remarks
/result		
@errorcode	[number]	See error codes below
/result/layoutstatus		
/result/layoutstatus/divas		
@count	[number]	the number of viewers in the list.
/result/layoutstatus/divas/diva		
divaid	[UID]	unique id (What is DIVA?)
name	[string]	name/description
addresses	[number]	the address of the server
/result/layoutstatus/divas/diva/monitors/monitor		
index	[number]	index number of the monitor
monitorid	[UID]	unique id of the monitor
name	[string]	name of the monitor
active	[boolean]	True or false indicating if the monitor is active
/result/layoutstatus/divas/diva/monitors/monitor/layout		
layoutid	[UID]	unique id of the layout
name	[string]	the name of the layout
/result/layoutstatus/divas/diva/monitors/monitor/layout/panels		
@count	[number]	the number of panels in this list
/result/layoutstatus/divas/diva/monitors/monitor/layout/panels/panel		
panelid	[UID]	unique id of the panel
name	[string]	the name of the panel
serverid	[UID]	the unique 64 bit server id of the camera displayed on this panel (empty or 0) if no device is selected on this panel
deviceid	[UID]	the unique 32 bit device id of the camera displayed on this panel (empty or 0) if no device is selected on this panel.
/result/layoutstatus/divas/diva/monitors/monitor/layout/panels/panel/status		
video	[string]	The status for the video stream (i.e. Ok, NoVideo, ConnectionLost, ServerLost, NotActive)

error code	description	remarks
0	Ok	Ok, no error

See Also

- getLayoutList
- getDivaList

Changelog

- 2.2.1 – added

3.24 getLicensePlate

Returns the last scanned licenseplate for the requested device. When this function is called without the timeout parameter it will try to find a scanned licenseplate within the time (in ms) set in the skiplicensetimeout parameter. This parameter is configured in the HTTP Server plugin Advanced menu. A value of 1000 milliseconds means that when calling this function it will search for scanned license plates in the past 1000 ms. As soon as one or more license plates are found the functions returns a result.

The parameter timeout is the number of milliseconds the function will look for license plates. As soon as the duration of the function is longer than the timeout, the function will return and there will be no license plates found.

Note!: The maximum process time for license plate detection per image is adjustable in the software per device with the parameter CarR Max Process Time. This setting could influence the results, especially if timeout is smaller.

Triggered mode: If the device profile parameter Activate CarR is set to Macro this function works in triggered mode. In this mode only images are scanned when the call to this function is made. This allows for fast response times.

Continuous mode: If the device profile parameter Activate CarR is set to Recorded Images then license plate detection is continuously performed on images with motion. This allows to return license plates that are read between calls to this function.

Remarks:

It could be possible that the system is still scanning an image when the timeout occurs and the function returns without a result. If a licenseplate is found in this last image and stored in the Database, it can be retrieved with a call to the function getEvents with proper parameters.

The result lastsampletime is about the time stamp of the last image that is processed.

The resulting message contains the last scanned license plate.

If the CarR options is not available, this function returns error "not implemented".

If the Sense database is not running, this function will not return results.

Request

```
command=getLicensePlate&deviceid=[UID]&serverid=[UID]&timeout=[0..3600000]
```

parameter	type	remarks
deviceid	[UID]	Unique device id
serverid	[UID]	Unique server id
timeout	[0..3600000]	Maximum number of milliseconds to wait until a licenseplate is found. Defaults to 1000

Response

```
<result errorcode="0">
  <lastsampletime>[timestamp]</lastsampletime>
  <licenseplates count="[number]">
    <licenseplate>
      <registration>[string]</registration>
      <confidence>[1..100]</confidence>
      <country>[abbreviation]</country>
      <time>[timestamp]</time>
      <serverid>[UID]</serverid>
      <deviceid>[UID]</deviceid>
    </licenseplate>
    ...
  </licenseplates>
 </result>
```

node	value	remarks
/result		
@errorcode	[number]	See error codes below
/result		
lastsampletime	[timestamp]	The UTC timestamp the last image was taken
/result/licenseplates		
licenseplates		List of licenseplates
@count	[number]	The number of licenseplates
/result/licenseplates/licenseplate		
registration	[string]	The license plate registration number
confidence	[1..100]	Percentage of the reliability of the license plate
country	[abbreviation]	Abbreviation of the country to which the license plate belongs
time	[timestamp]	The UTC timestamp the image was take
serverid	[UID]	Unique id of the server
deviceid	[UID]	Unique id of the device

error code	description	remarks
0	Ok	Ok, no error
-1	Not implemented	Function not implemented
-2	No device	No device found for the supplied cameraid/deviceid
-20	No CarR	System or device not set for license plate recognition
-21	Device not active	The device is not active in the current profile and cannot produce any results

See Also

- Timestamp format

Changelog

- 1.12.3 – Documented if timestamps are local or UTC timestamps
- 1.11.3 – Added

3.25 getLiveImage

The system can send live images of a certain camera. The images will be in JPEG format. imagewidth and imageheight only work if transcoding is enabled.

Request

```
command=getLiveImage&serverid=[UID]&cameraid=[UID]&imagewidth=[number>0]&imagehe
ight=[number>0]&fittopanel=[bool]
```

parameter	type	remarks	
serverid		[UID]	unique server id.
cameraid		[UID]	unique camera id
imagewidth		[number > 0]	optional image width for transcoding
imageheight		[number > 0]	optional image height for transcoding
fittopanel		[boolean]	optional scale image, fit to panel. Values: true, false

Response

```
jpeg image (or result in case of error)
```

error code	description	remarks	
0		Ok	Ok, no error
-2		No device	No device found for the supplied cameraid/deviceid
-11		No image	There is no image found to return
-18		General error	An undefined error occured
-21		Device not active	The device is not active in the current profile and cannot produce any results

Changelog

- 2.1.1 – Added parameter serverid
- 1.4.1 – Added imagewidth and imageheight
- 1.3.1 – Added

3.26 getMacroList

This function is deprecated use getServerMacroList or getViewerMacroList instead.

Obtains a list of all viewer macro's. These macro's can be executed with the executeMacro command.

Request

```
command=getMacroList
```

Response

```
<result errorcode="0">
  <macros count="[number]">
    <macro>
      <macroid>[UID]</macroid>
      <name>[string]</name>
      <type>[string]</type>
    </macro>
    ...
  </macros>
 </result>
```

node	value	remarks
/result		
@errorcode	[number]	See error codes below
/result/macros		
@count	[number]	The number of macros
/result/macros/macro		
macroid	[UID]	unique id of the macro
name	[string]	the unique user defined name of the macro
type	[string]	type of the macro 'viewer' or 'server'

See Also

- executeAction
- getServerMacroList
- getViewerMacroList

Changelog

- 2.2.1 – added

3.27 getMultiLayoutList

Obtains a list of all selectable multilayouts from the system. These multilayouts can be selected to display layouts on the monitors attached to the system.

Request

```
command=getMultiLayoutList
```

Response

```
<result errorcode="0">
  <multilayouts count="[number]">
    <multilayout>
      <multilayoutid>[UID]</multilayoutid>
      <name>[string]</name>
    </multilayout>
    ...
  </multilayouts>
 </result>
```

node	value	remarks
/result		
@errorcode	[number]	See error codes below
/result/multilayouts		
@count	[number]	The number of multilayouts
/result/multilayouts/multilayout		
multilayoutid	[UID]	unique id of the multilayout
description	[string]	user defined name

error code	description	remarks
0	Ok	Ok, no error
-18	General error	An undefined error occured

See Also

- selectMultiLayout

Changelog

- 2.2.1 – added

3.28 getOptions

Returns the camera possibilities for the positioning options: pan, tilt, zoom, focus and iris. And returns a list with the preset names.

- Relative true means the camera supports the parameter with the command setRelativePosition
- Absolute true means the camera supports the parameter with the command setAbsolutePosition.
- Position true means the camera supports the parameter with the command getPosition.
- Stop true means the camera supports the parameter with the command holdPosition.
- Auto true means the camera supports the parameter with the command setAbsolutePosition.

This command always return the available options, even when the device is not a dome. In that case, instead of returning a 'No Dome' message, this command will return the results with all positioning values false.

Request

```
command=getOptions&serverid=[UID]&cameraid=[UID]
```

parameter	type	remarks
serverid	[UID]	unique server id.
cameraid	[UID]	unique camera id

Response

```
<result errorcode="0">
  <options>
    <positioning>
      <pan>...</pan>
      <tilt>...</tilt>
      <zoom>...</zoom>
      <focus>...</focus>
      <iris>...</iris>
    </positioning>
    <presets count>
      <presets number>...</presets number>
    </presets count>
 ...
 </options>
 </result>
```

node	value	remarks
/result		
@errorcode	[number]	See error codes below
/result/positioning		
pan	[false|true]	options for relative, absolute,

node	value	remarks	
tilt		[false|true]	position and stop options for relative, absolute, position and stop
zoom		[false|true]	options for relative, absolute, position, stop and auto
focus		[false|true]	options for relative, absolute, position, stop and auto
iris		[false|true]	options for relative, absolute, position, stop and auto
/result/presets			
@count		[number]	The number of presets
/result/presets/preset*			
@number		[number]	The index number of the preset
		[string]	The value of the preset

error code	description	remarks	
0		Ok	Ok, no error
-2		No device	No device found for the supplied cameraid/deviceid
-18		General error	An undefined error occured

Changelog

- 2.1.1 – Added serverid parameter
- 1.3.3 – Added

3.29 getPersonFields

Returns a list of all available persons fields.

Request

```
command=getPersonFields
```

Response

```
<result errorcode="0">
  <personfields>
    <field>firstname</field>
    <field>lastname</field>
    <field>address</field>
    <field>postalcode</field>
    <field>city</field>
    <field>phone</field>
    <field>fax</field>
    <field>mobile</field>
    <field>licenseplate</field>
    <field>lastnameprefix</field>
    <field>notes</field>
    <field>reference</field>
    <field>expirationdate</field>
    <field>enableexpirationdate</field>
  </personfields>
</result>
```

node	value	remarks
/result		
@errorcode	[number]	See error codes below
/result/personfields		
field	[string]	name of the field in the database

error code	description	remarks
0	Ok	Ok, no error
-18	General error	An undefined error occured

Changelog

- 2.6.1 – added

3.30 getPersonList

Obtains a list of all persons stored in the database.

Request

```
command=getPersonList
```

Response

```
<result errorcode="0">
  <persons count="[number]">
    <person>
      <id>[uid]</id>
      <firstname>[string]</firstname>
      <lastname>[string]</lastname>
      <address>[string]</address>
      <postalcode>[string]</postalcode>
      <city>[string]</city>
      <phone>[string]</phone>
      <fax>[string]</fax>
      <mobile>[string]</mobile>
      <licenseplate>[string]</licenseplate>
      <lastnameprefix>[string]</lastnameprefix>
      <reference>[string]</reference>
      <notes>[string]</notes>
      <expirationdate>[string]</expirationdate>
      <enabledexpirationdate>[boolean]</enabledexpirationdate>
    </person>
  </persons>
</result>
```

node	value	remarks
/result		
@errorcode	[number]	See error codes below
/result/persons		
@count	[number]	number of persons
/result/persons/person		
id	[UID]	unique person id
firstname	[string]	firstname of the person
lastname	[string]	firstname of the person
address	[string]	address of the person
postalcode	[string]	postal code of the person
city	[string]	city of the person
phone	[string]	phone number of the person
fax	[string]	fax number of the person
mobile	[string]	mobile number of the person

node	value	remarks	
licenseplate		[string]	license plate of the person
notes		[string]	notes added to the person
expirationdate		[string]	the date the person will be removed from the database
enabledexpirationdate		[boolean]	flag to indicate the person can be expired

error code	description	remarks	
0		Ok	Ok, no error
-18		General error	An undefined error occured

Changelog

- 2.6.1 – added

3.31 getPluginOptions

Retrieve a list of actions that can be executed with the command executeAction.

Request

```
command=getPluginOptions
```

Response

```
<result errorcode="0">
  <pluginoptions>
    <version>[string]</version>
    <apiversion>[string]</apiversion>
    <skiplicensetimeout>[string]</skiplicensetimeout>
  </pluginoptions>
 </result>
```

Tag

node	value	remarks
/result		
@errorcode	[number]	See error codes below
/result/pluginoptions		
version	[string]	version of the Plugin
apiversion	[string]	apiversion of the Plugin
skiplicensetimeout	[string]	shows time to skip the license (ms)

error code	description	remarks
0	Ok	Ok, no error
-1	Not implemented	Function not implemented
-18	General error	An undefined error occured

Changelog

- 2.2.1 – added

3.32 getPosition

This command returns the current position of the dome.

Request

```
command=getPosition&serverid=[UID]&cameraid=[UID]
```

parameter	type	remarks	
serverid		[UID]	unique server id
cameraid		[UID]	unique camera id

Response

```
<result errorcode="0">
  <pan>900</pan>
  <tilt>-450</tilt>
  <zoom>0</zoom>
  <focus>0</focus>
  <iris>0</iris>
</result>
```

node	value	remarks	
/result			
@errorcode		[number]	See error codes below
pan		[-1800 – 1800]	pan position
tilt		[-1800 – 1800]	tilt position
zoom		[-1 – 1000]	zoom position, -1 = auto zoom
focus		[-1 – 1000]	focus position, -1 = auto focus
iris		[-1 – 1000]	iris position, -1 = auto iris

error code	description	remarks	
0		Ok	Ok, no error
-2		No device	No device found for the supplied cameraid/deviceid.
-18		General error	An undefined error occured
-19		No dome	The device is not a dome device
-21		Device not active	The device is not active in the current profile and cannot produce any results

Changelog

- 2.1.1 – added serverid
- 1.7.1 – added

3.33 getPresetImage

Get the preset images of a dome, a preset image is a represented image of the located preset position.

Request

```
command=getpresetimage&serverid=[UID]&deviceid=[UID]&preset=[number]& imagewidth
=[number >40&number <2560]&imageheight=[number>40&number<1600]
```

parameter	type	remarks	
serverid		[UID]	unique server id.
deviceid		[UID]	unique device id.
preset		[number]	the number of the preset
imagewidth		[number > 40 & number < 2560]	optional image width for transcoding
imageheight		[number > 40 & number < 1600]	optional image height for transcoding

Response

```
jpeg image (or result in case of error)
```

error code	description	remarks	
-2		No device	No device found for the supplied cameraid/deviceid
-11		No image	There is no image found to return
-21		Device not acive	The device is not active in the current profile and cannot produce any results

See Also

- setPreset
- gotoPreset

Changelog

- 2.1.1 – Added

3.34 getServerList

Removed as of version 2
Deprecated as of 1.11.3, use getDivaList instead.

Obtains a list of all servers in a master slave configuration. The first entry will be of the master server, the server that handles the request.

Request

```
command=getServerList
```

Response

```
<result errorcode="0">
  <servers count="[number]">
    <server id="[UID]">[ip-address]</server>
    ...
  </servers>
 </result>
```

Tag

node	value	remarks
/result		
@errorcode	[number]	See error codes below
/result/servers		
@count	[number]	The number of servers
server	[ip-address]	The ip adress of the server
/result/servers/server		
@id	[UID]	unique id of the server

error code	description	remarks
0	Ok	Ok, no error
-18	General error	An undefined error occured

Changelog

- 2.1.1 – Removed
- 1.11.3 – Deprecated, use getDivaList instead.
- 1.4.1 – Added

3.35 getServerMacroList

Obtains a list of all server macro's. These macro's can be executed with the executeMacro command.

Request

```
command=getServerMacroList
```

Response

```
<result errorcode="0">
  <servermacros count="[number]">
    <macro>
      <id>[UID]</id>
      <name>[string]</name>
      <type>[string]</type>
      <valid>[boolean]</valid>
      <status>[string]</status>
    </macro>
    ...
  </servermacros>
 </result>
```

node	value	remarks
/result		
@errorcode	[number]	See error codes below
/result/servermacros		
@count	[number]	The number of macros
/result/servermacros/macro		
id	[UID]	unique id of the macro
name	[string]	the unique user defined name of the macro
type	[string]	type of the macro 'viewer' or 'server'
valid	[boolean]	
status	[string]	'Active' or 'Inactive'

See Also

- executeAction
- getMacroList
- getViewerMacroList

Changelog

- 2.2.1 – added

3.36 getServerOptions

Returns server settings. See chapter Transcoding for an explanation about Transcoding.

Because the system is recording most of the time, the oldest and newest stored image time can vary between successive calls to this function.

When is serverid is omitted, the info of the management server will be returned.

Request

```
command=getServerOptions&amp;amp;amp;serverid=[UID]
```

serverid	[UID]	unique server id (optional)

Response

```
<result errorcode="0">
  <serveroptions>
    <serverid>[id]</serverid>
    <description>[string]</description>
    <title>[string]</title>
    <version>[no.no.no.no]</version>
    <registrationnumber>[string]</registrationnumber>
    <cameracount>[number]</cameracount>
    <servercount>[number]</servercount>
  </serveroptions>
</result>
```

Tag

node	value	remarks
/result		
@errorcode	[number]	See error codes below
/result/serveroptions		
serverid	[id]	the id of the server
description	[string	description of the system.
title	[string]	title of the system.
version	[no.no.no.n]	version of the system
registrationnumber	[string]	registration number of the system
cameracount	[number]	maximum number of cameras for this server.
servercount	[number]	number of servers added to the system.

error code	description	remarks
0	Ok	Ok, no error
-18	General error	An undefined error occurred

See Also

- Timestamp format

Changelog

- 2.7.2 – Changed documentation of servercount to “number of servers added to the system.”
- 2.1.1 – Added
- 1.11.3 – Fixed documentation error. Timestamps for this function are UTC timestamps
- 1.6.1 – Added oldest and newest stored image time fields
- 1.4.1 – Added

3.37 getServerStatus

Returns information about the status of a specific server. If the parameter serverid is omitted, the information of the local server is returned. Only taggedvideo is documented (2.4.1). Tagged video is not deleted until released which must be done by a user in the system.

Request

```
command=getServerStatus&serverid=[UID]
```

Response

```
<result errorcode="0">
  <result>
    <serverstatus>
      <id>[UID]</id>
      <taggedvideo>
        <percentage>[0..100]</percentage>
        <status>[string]</status>
      </taggedvideo>
    </serverstatus>
  </result>
</result>
```

Tag

node	value	remarks
/result		
@errorcode	[number]	See error codes below
/result/result/serverstatus		
id	[UID]	unique id of the server
/result/result/serverstatus/taggedvideo		
percentage	[0..100]	percentage of video storage that is tagged.
status	[string]	indicates if tagging is active ('active') or not ('inactive')

error code	description	remarks
0	Ok	Ok, no error
-17	No Server	There is no server with the suplied id.
-18	General error	An undefined error occured

See Also

- executeAction
- getSystemStatus
- getActionList

Changelog

- 2.4.2 – Added status by taggedvideo.
- 2.4.1 – Added

3.38 getStatusTypes

Obtains a list of all supported status types. The lists consists of name and description pairs. The name can be used in streamStatus.

Request

```
command=getStatusTypes
```

Response

```
<result errorcode="0">
  <statustypes count="[number]">
    <statustype>
      <name>[string]</name>
      <description>[string]</description>
    </statustype>
    ...
  </statustypes>
 </result>
```

node	value	remarks
/result		
@errorcode	[number]	See error codes below
/result/statustypes		
@count	[number]	The number of status types
/result/statustypes/statustype		
name	[string]	the name, as used in streamStatus
description	[string]	a system defined description of the status type

error code	description	remarks
0	Ok	Ok, no error
-18	General error	An undefined error occured

See Also

- streamStatus

Changelog

- 2.5.1 – Added

3.39 getSystemStatus

This function obtains the status of the "whole" Sense VMS system, while getServerStatus will only return the status of one single server.

Request

Note: serverid is optional.

```
command=getSystemStatus&serverid=[UID]
```

Response

```
<result errorcode="0">
    <systemstatus>
        <id>[UID]</id>
        <license>[number]</license>
        <devicestatuslist count="[number]"></devicestatuslist>
        <storage>
            <info>[string]</info>
            <storagesize>[number]</storagesize>
            <blocksize>[number]</blocksize>
        </storage>
        <health>
            <memory>
                <info>[string]</info>
                <load>[number]</load>
                <totalphys>[number]</totalphys>
                <availphys>[number]</availphys>
                <totalpagefile>[number]</totalpagefile>
                <availpagefile>[number]</availpagefile>
                <totalvirtual>[number]</totalvirtual>
                <availvirtual>[number]</availvirtual>
            </memory>
            <processmemory>
                <workingsetsize>[number]</workingsetsize>
            </processmemory>
            <cpu>
                <cores>[number]</cores>
                <load>[number]</load>
            </cpu>
            <network>
                <inputbandwidth>[number]</inputbandwidth>
                <outputbandwidth>[number]</outputbandwidth>
            </network>
            <harddisk>
                <readbandwidth>[number]</readbandwidth>
                <writebandwidth>[number]</writebandwidth>
            </harddisk>
```

```
        </health>
        <systeminfo>
            <oem-id>[UID]</oem-id>
            <numofprocessors>[number]</numofprocessors>
            <page-size>[number]</page-size>
            <processor-type>[number]</processor-type>
        </systeminfo>
        <serversettings>
            <managementserveraddress>[number]</managementserveraddress>
        </serversettings>
        <statusevents>
            <managementconnectionstatus>[string]</managementconnectionstatus></p
re>
        </statusevents>
        <divaconnections>
            <connection>
                <IPAddress>[ip-address]</IPAddress>
                <type>[string]</type>
                <address>[ip-address]</address>
                <username>[string]</username>
                <divaid>[number]</divaid>
                <description>[string]</description>
            </connection>
        </divaconnections>
        <taggedvideo>
            <percentage>[number]</percentage>
            <status>[string]</status>
        </taggedvideo>
    </systemstatus>
</result>
```

node	value	remarks
/result		
@errorcode	[number]	See error codes below
/result/systemstatus		
id	[id]	unique id of the systemstatus
license	[number]	license number
devicestatuslist	[number]	the number of devicestatus objects
/result/systemstatus/storage		
info	[string]	information about the storage values
storagesize	[number]	the number of storagesize
blocksize	[number]	the number of blocksize
/result/systemstatus/storage/health		
/result/systemstatus/health/memory		
info	[string]	information about the memory values
load	[number]	the number of the load memory
totalphys	[number]	the number of the total physical memory

node	value	remarks
availphys	[number]	the number of available physical memory
totalpagefile	[number]	the number of the total page files
availpagefile	[number]	the number of available page files.
totalvirtual	[number]	the number of the total virtual memory.
availvirtual	[number]	the number of available virtual memory
/result/systemstatus/health/processmemory		
workingsetsize	[number]	the size of the working sets of the processmemory
/result/systemstatus/health/cpu		
cores	[number]	cpu cores value
load	[number]	cpu load value
/result/systemstatus/health/network		
inputbandwidth	[number]	width of inputband of the network
outputbandwidth	[number]	width of outputband of the network
/result/systemstatus/health/harddisk		
readbandwidth	[number]	width of readband of the harddisk
writebandwidth	[number]	width of writeband of the harddisk
/result/systemstatus/systeminfo		
oem-id	[number]	unique id of the mainboard
numofprocessors	[number]	the number of processors
page-size	[number]	the number of pages.
processor-type	[number]	the type of the processor.
/result/systemstatus/serversettings		
managementserveraddress	[number]	the address of the managementserver
/result/systemstatus/statusevents		
managementconnectionstatus	[string]	Connection status with management server
/result/systemstatus/divaconnections		
/result/systemstatus/divaconnections/connection		
IPaddress	[ip-address]	The IP address of the DIVA server
type	[string]	type of server
address	[ip-address]	The address of the server
username	[string]	The current logged in user
divaid	[uid]	Unique ID of the server
description	[string]	server description
/result/systemstatus/taggedvideo		
percentage	[number]	the amount of tagged video in percentages
status	[string]	status of the tagged video (active, inactive)

error code	description	remarks
0	Ok	Ok, no error

See Also

- getHardwareStatus

Changelog

- 2.4.2 – Added

3.40 getVideoTagList

Obtains a list of all created Video tags.

Request

```
command=getVideoTagList
```

Response

```
<result errorcode="0">
 <videotaglist count="[Number]">
  <videotag>
   <id>[UID]</id>
   <serverid>[UID]</serverid>
   <deviceid>[UID]</deviceid>
   <startdate>[timestamp]</startdate>
   <enddate>[timestamp]</enddate>
   <title>[string]</title>
   <comment>[string]</comment>
   <status>[string]</status>
  </videotag>
 </videotaglist>
</result>
```

node	value	remarks
/result		
@errorcode	[number]	See error codes below
/result/videotaglist		
@count	[number]	The number of videotags.
/result/videotaglist/videotag		
id	[UID]	unique id
serverid	[UID]	unique server id
deviceid	[UID]	unique device id
startdate	[timestamp]	Tags with a (local) timestamp >= startdate are returned.
enddate	[timestamp]	Tags with a (local) timestamp < = enddate are returned.
title	[string]	A user defined title
comment	[string]	A user defined comment
status	[string]	'Active' or 'Inactive'

error code	description	remarks
0	Ok	Ok, no error
-18	General error	An undefined error occured

See Also

- updateVideoTag
- releaseVideoTag
- Examples – Video tagging

Changelog

- 2.4.3 – Added

3.41 getViewerMacroList

Obtains a list of all viewer macro's. These macro's can be executed with the executeMacro command.

Request

```
command=getViewerMacroList
```

Response

```
<result errorcode="0">
  <macros count="[number]">
    <macro>
      <macroid>[UID]</macroid>
      <name>[string]</name>
      <type>[string]</type>
      <valid>[boolean]</valid>
      <status>[string]</status>
    </macro>
    ...
  </macros>
 </result>
```

node	value	remarks
/result		
@errorcode	[number]	See error codes below
/result/macros		
@count	[number]	The number of macros
/result/macros/macro		
macroid	[UID]	unique id of the macro
name	[string]	the unique user defined name of the macro
type	[string]	type of the macro 'viewer' or 'server'
valid	[boolean]	
status	[string]	'Active' or 'Inactive'

See Also

- executeAction
- getMacroList

Changelog

- 2.2.1 – added

3.42 gotoPreset

Moves the dome to a stored preset position.

Request

```
command=gotoPreset&serverid=[UID]&cameraid=[UID]&preset=[number>=0]
```

parameter	type	remarks	
serverid		[UID]	unique server id
cameraid		[UID]	unique camera id
preset		[number>=0]	number of the preset to move the dome to

Response

See chapter Results for result syntax and result codes.

node	value	remarks	
/result			
@errorcode		[number]	See error codes below

error code	description	remarks	
0		Ok	Ok, no error
-2		No device	No device found for the supplied cameraid/deviceid.
-18		General error	An undefined error occured
-19		No dome	The device is not a dome device
-21		Device not active	The device is not active in the current profile and cannot produce any results

Changelog

- 2.1.1 – added serverid
- 1.6.2 – added

3.43 grabLicensePlate

Grabs an image from the live image stream and puts it in the license plate engine buffer. The image is scanned asynchroniously for a licenseplate. If one is found, a CarRLicensePlateEvent is fired.

If the CarR option is not available this function returns 'not implemented'.

Request

```
command=grabLicensePlate&cameraid=[UID]&serverid=[UID]
```

parameter	type	remarks
cameraid	[UID]	unique camera id.
serverid	[UID]	unique server id.

Response

See chapter Results for result syntax and result codes.

error code	description	remarks
0	Ok	Ok, no error
-1	Not implemented	Function not implemented.
-2	No device	No device found for the supplied cameraid
-21	Device not active	The device is not active in the current profile and cannot produce results

See also

getLicensePlate
Event result values

Changelog

- 1.10.0 – added

3.44 holdPosition

This function is deprecated. To stop dome movement, the function setRelativePosition should be used with the value zero (0) for all PTZ-parameters.

This command can be issued to selectively stop dome control. All values are optional, when not supplied, or the value is zero (0), the current value is maintained

Request

```
command=holdPosition&serverid=[UID]&cameraid=[UID]&pan=[0|1]&tilt=[0|1]&zoom=[0|1]&focus=[0|1]&iris=[0|1]
```

parameter	type	remarks	
serveid		[UID]	unique server id.
cameraid		[UID]	unique camera id
pan		[0|1]	0 means do nothing, 1 means stop pan changing
tilt		[0|1]	0 means do nothing, 1 means stop tilt changing
zoom		[0|1]	0 means do nothing, 1 means stop zoom changing
focus		[0|1]	0 means do nothing, 1 means stop focus changing
iris		[0|1]	0 means do nothing, 1 means stop iris changing

Response

error code	description	remarks	
0		Ok	Ok, no error
-2		No device	No device found for the supplied cameraid/deviceid
-18		General error	An undefined error occured
-19		No dome	Ok, no erro
-21		Device no active	The device is not active in the current profile and cannot produce any results

Changelog

- 2.2.1 – Added serverid
- 1.7.1 – Added parameters focus and iris
- 1.3.4 – Added parameters pan, tilt and zoom
- 1.3.2 – Added

3.45 releaseVideoTag

Release the videotag with the supplied id.

Request

```
command?command=releaseVideoTag&id=[UID]
```

Response

```
<result errorcode="0">
 <descripion>Ok</description>
</result>
```

error code	description	remarks
0	Ok	Ok, no error
-18	General error	An undefined error occured
-26	Videotag does not exist	Videotag does not exist or cannot be found

See Also

- getVideoTagList
- updateVideoTag
- Examples – Video tagging

Changelog

- 2.4.3 – Added

3.46 selectCamera

Selects a camera. The camera is displayed in the panel which is identified by panelid in the layout which is identified by layoutid. The system (set of servers and viewers) searches all viewers to find the place where the layout is displayed. Use parameter cameraview to define the camera, this includes the serverid and deviceid. If divaid is given, the command will only by send to the system with that id (What is DIVA?).

The use of panelindex is deprecated. This value only counts panels which are used for displaying video. Other panels are skipped. I.e. if there are 4 panels of which the second panel is an event panel and panelindex = 2, the fourth panel (with panelindex 3) is used. Note that panelid and panelindex cannot be used simultaneously. The use of panelindex is deprecated. Use panelid instead.

Request

```
command=selectCamera&cameraview=[camview]&layoutid=[UID]&panelid=[UID]&panelinde
x=[0..]&divaid=[UID]
```

parameter	type	remarks	
cameraview		[camview]	Camera view
layoutid		[UID]	unique layout id
panelid		[UID]	panel id
panelindex		[0..]	panel index
divaid		[UID]	unique id (What is DIVA?)
keeppanelselection		[boolean]	true or false to keep current panel selected

Response

error code	description	remarks	
0		Ok	Ok, no error
-2		No device	No device found for the supplied cameraid/deviceid
-14		No Layout	No layout found for the supplied layoutid
-18		General error	An undefined error occured

See Also

- getDeviceList
- getLayoutList
- selectLayout
- fillLayout

Changelog

- 2.3.18 – Added panelid
- 2.3.1 – Added divaid
- 2.1.2 – Fixed dispatching to viewers, and using serverid
- 1.11.1 – CameraView extended with left, top, righ and bottom parameters
- 1.10.1 – Added

3.47 selectLayout

Selects a layout. A layout is a set of panels that is displayed on a monitor. The layout is identified by layoutid. The layout is displayed on the monitor identified by monitorindex on the system identified by divaid. If the divaid is omitted, this command is sent to all viewers that will all switch this layout if possible..

Request

```
command=selectLayout&divaid=[UID]&monitorindex=[0..]&layoutid=[UID]
```

parameter	type	remarks
divaid	[UID]	Unique diva id. Determines the viewer, if omitted or -1 all viewers are addressed
monitorindex	[0..]	index of the monitor
layoutid	[UID]	unique layout id

Response

error code	description	remarks
0	Ok	Ok, no error
-18	General error	An undefined error occured

See Also

- getDivaList
- GetLayoutList
- fillLayout
- selectCamera

Changelog

- 2.1.2 – Fixed dispatching to viewers
- 1.10.2 – Added

3.48 selectMultiLayout

This command selects a specific multilayout. A multilayout is a set of layouts that are displayed on all monitors attached to a viewer. The viewer itself is identified by the divaid. If this id is supplied and the viewer exists, that viewer will be selected for further commands. Otherwise, the local system the request is send to will be selected. The multilayout with the supplied id or name on the selected system will be displayed if it exists. If both an ID and name is given, the ID will have priority. The name parameter is case sensitive.

Request

```
command=selectMultiLayout&divaid=[UID]&multilayoutid=[UID]
```

parameter	type	remarks	
divaid		[UID]	unique id
multilayoutid		[UID]	Unique multi layout id (priority over name)
name		[string]	User given name of multi layout (case sensitive)

Response

error code	description	remarks	
-1		Not implemented	Function not implemented

See Also

- getDivaList
- getMuliLayoutList

Changelog

- 2.5.10 – Added name parameter
- 2.2.1 – Added

3.49 setAbsolutePosition

Sends a dome to a specific location. All values are optional. When not supplied the current value is maintained. The command returns as soon as the PTZ is started.

Request

```
command=setAbsolutePosition&serverid=[UID]&cameraid=[UID]&pan=[-1800..1800]&tilt
=[-1800..1800]&zoom=[0..1000]&focus=[-1..1000]&iris=[-1..1000]
```

parameter	type	remarks	
serverid		[UID]	unique server id.
cameraid		[UID]	unique camera id
pan		[-1800..1800]	absolute pan position in 0.1 degrees
tilt		[-1800..1800]	absolute tilt position in 0.1 degrees
zoom		[0..1000]	0..1000 value in 0.1 percentage
focus		[-1..1000]	-1 for autofocus, 0..1000 value in 0.1 percentage
iris		[-1..1000]	-1 for autoiris, 0..1000 value in 0.1 percentage

Response

error code	description	remarks	
0		Ok	Ok, no error
-2		No device	No device found for the supplied cameraid/deviceid
-19		No dome	The device is not a dome device
-21		Device not active	The device is not active in the current profile and cannot produce any results

Changelog

- 2.1.1 – Added serverid
- 1.7.1 – Added

3.50 setParameters

This command can be issued to set several system wide parameters. All arguments are optional. Transcoding Bandwidth only works with live images.

Request

```
command=setParameters&transcodingquality=[0..99]&transcodingenabled=[0|1]&imagew
idth=[number>0]&imageheight=[number>0]
```

parameter	type	remarks
transcodingenabled	[0|1]	Determines if images are transcoded (1) or not (0).
transcodingquality	[0..99]	Allowed maximum transcoding quality in percentage (0, worst – 99, best).
imagewidth	[number > 0]	Used in decompressing and rescaling transcoded images.
imageheight	[number > 0]	Used in decompressing and rescaling transcoded images.

Response

error code	description	remarks
0	Ok	Ok, no error
-18	General error	An undefined error occured

Changelog

- 2.7.1 – Parameters updated.
  - Removed TranscodingBandwidth
  - WebTranscodedImagesRequested changed to transcodingenabled
  - WebDefaultImageWidth changed to imagewidth
  - WebDefaultImageHeight changed to imageheigth
- 1.3.6 – Added

3.51 setPlaybackDateTime

Sets the playback timestamp on a specific viewer or on all viewers.

This function checks if the supplied time parameter is a valid timestamp. This value is not check to see if it is within the available recording time space. The viewer that executes this function will show the supplied UTC timestamp as local time for that particular system.

Request

```
command=setPlaybackDateTime&divaid=[uid]&time=[timestamp]
```

parameter	type	remarks	
divaid		[UID]	determines the viewer, if ommitted or -1 all viewers are adressed.
time		[timestamp]	UTC timestamp for the playback datetime.

Response

error code	description	remarks	
0		Ok	Ok, no error
-4		Invalid timestamp	Invalid datetime format. See Timestamp.
-18		General error	An undefined error occured

See Also

- Timestamp format
- GetServerOptions

Changelog

- 1.12.3 – Documented if timestamps are local or UTC timestamps.
- 1.11.3 – Strict checking on starttime and endtime parameter format. Returns -4 on error (see Result Codes).
- 1.11.1 – Added.

3.52 setPlaybackSpeed

This function sets the playback speed on the selected client or on all clients.

The playback speed can be taken from a fixed list of values. If the speed is positive, the client(s) will play forwards. If the speed is negative, the client(s) will play backwards.

Request

```
command=setPlaybackSpeed&divaid=[uid]&speed=[value]
```

parameter	type	remarks	
divaid		[UID]	Unique diva id. Determines the viewer, if omitted or -1 all viewers are addressed
speed		[value]	Sets the playback speed at a specific value. Allowed values are: 0.5, 1, 2, 4, 8, 16 for forward playback and -0.5, -1, -2, -4, -8, -16 for backward playback

Response

```
<result errorcode="0">
 <description>Ok</description>
</result>
```

error code	description	remarks	
0		Ok	Ok, no error
-1		Not implemented	The function requires an api license
-12		No Diva	There's no system found with the supplied divaid
-18		General error	An undefined error occurred
-22		Value is out of range	The speed parameter value is not in the list of allowed values
-23		An empty value is not allowed	The speed parameter is missing
-28		Not a Diva client	The system with the supplied divaid is not a client

See Also

- startPlayback
- stopPlayback

- setPlaybackDateTime

Changelog

- 2.4.5 – Added

3.53 setPreset

Stores the current position of a dome as a preset.

Request

```
command=setPreset&serverid=[UID]&cameraid=[UID]&preset=[number>=0]
```

Response

error code	description	remarks	
0		Ok	Ok, no error
-2		No device	No device found for the supplied cameraid/deviceid
-18		General error	An undefined error occured
-19		No dome	The device is not a dome device.
-21		Device not active	The device is not active in the current profile and cannot produce any results.

Changelog

- 2.1.1 – Added serverid.
- 1.6.2 – Added.

3.54 setRelativePosition

Sends a dome to a relative position. All values are optional. When not supplied, the current value is maintained. The values range from -1000 to 1000. The value determines the speed relative to the maximum speed the dome can handle. To stop dome movement, use this function with the value zero (0) for all PTZ-parameters.

Request

```
command=setRelativePosition&serverid=[UID]&deviceid=[UID]&pan=[-1000..1000]&tilt
=[-1000..1000]&zoom=[-1000..1000]&focus=[-1000..1000]&iris=[-1000..1000]
```

parameter	type	remarks	
serveid		[UID]	unique server id.
deviceid		[UID]	unique device id
pan		[-1000..1000]	-1000 to pan left, 1000 to pan right.
tilt		[-1000..1000]	-1000 to tilt up, 1000 to tilt down.
zoom		[-1000..1000]	-1000 to zoom out, 1000 to zoom in (at max speed).
focus		[-1000..1000]	-1000 to focus out, 1000 to focus in (at max speed).
iris		[-1000..1000]	-1000 for smaller iris, 1000 for bigger iris (at max speed).

Response

error code	description	remarks	
-2		No device	No device found for the supplied cameraid/deviceid.
-18		General error	An undefined error occured
-19		No dome	The device is not a dome device.
-21		Device not active	The device is not active in the current profile and cannot produce any results.

Changelog

- 2.1.1 – Added serverid
- 1.3.1 – Added

3.55 startPlayback

This function starts the playback on the selected client or on all clients.

Request

```
command=startPlayback&divaid=[uid]
```

parameter	type	remarks
divaid	[UID]	Unique diva id. Determines the viewer, if omitted or -1 all viewers are addressed

Response

```
<result errorcode="0">
 <description>Ok</description>
</result>
```

error code	description	remarks
0	Ok	Ok, no error
-1	Not implemented	The function requires an api license
-12	No Diva	There's no system found with the supplied divaid
-18	General error	An undefined error occurred
-28	Not a Diva client	The system with the supplied divaid is not a client

See Also

- stopPlayback
- setPlaybackSpeed
- setPlaybackDateTime

Changelog

- 2.4.5 – Added

3.56 startVideoTag

Starts tagging video so recordings will not be deleted. Redirect to reference example

3.57 stopPlayback

This function stops the playback on the selected client or on all clients.

Request

```
command=stopPlayback&divaid=[uid]
```

parameter	type	remarks
divaid	[UID]	Unique diva id. Determines the viewer, if omitted or -1 all viewers are addressed

Response

```
<result errorcode="0">
 <description>Ok</description>
</result>
```

error code	description	remarks
0	Ok	Ok, no error
-1	Not implemented	The function requires an api license
-12	No Diva	There's no system found with the supplied divaid
-18	General error	An undefined error occurred
-28	Not a Diva client	The system with the supplied divaid is not a client

See Also

- startPlayback
- setPlaybackSpeed
- setPlaybackDateTime

Changelog

- 2.4.5 – Added

3.58 stopVideoTag

Stops tagging video. Redirect to reference example

3.59 streamEvents

When an event occurs in the system, it will be send (pushed) to the client. The system keeps pushing events until the connection is closed or another request is send over the current connection. The events are send as part of a multipart HTTP stream. All arguments are optional and can be used to filter the kind of event.The parameters deviceid and serverid can be used to get events from a specific device (from a specific server). The types parameter is a semicolon (;) seperated list of event types that should be included in the results. If this parameter is present, only events of matching types are included in the results. All other event types are excluded.

Event types are not grouped, as used in the command getEvents. Instead, every event type of an event that occured should match partially one (or more) of the event types of the supplied parameter before it is send to the client. Partially means that if i.e. types 'alarm,motion' are requested, all events with the text 'alarm' or 'motion' somewhere in the event type of the event that occured will be send to the client. In this example events with the following event types will be send: 'AlarmTrigger1', 'AlarmTrigger2', 'AlarmTrigger3', 'USBAlarmInput', 'USBAlarmInputLow', 'Motion' and so on. By using the same event types as described in the chapter Event Types the same kind of grouping can be made as is used in the command getEvents. See also 'Event result values' in this chapter for a list of event types that can occur.

This command will return immediately with a multipart/x-mixed-replace content type and specific boundary. When an event occures, it will be send as xml message as denoted below, together with the boundary indicating the end of the multipart. If another request is send from the client, the ending boundary will be send.

Request

```
command=streamEvents&types=[string]&deviceid=[uid]&serverid=[uid]
```

parameter	type	remarks
types	[string]	semicolon (;) separated list of event types that should be included in the results. All other types are excluded.
deviceid	[uid]	filters events from a specific device.
serverid	[uid]	filters events from a specific server.

Response

```
<event>
  <id>[uid]</id>
  <name>[string]</name>
  <value>[string]</value>
  <time>[timestamp]</time>
  <serverid>[uid]</serverid>
  <deviceid>[uid]</deviceid>
 </event>
```

node	value	remarks	
/event			
id		[uid]	id of event.
name		[string]	name of event. (API version 2.7.1+)
type		[string]	type of event. (<= API version 2.6.1)
value		[string]	value of event.
time		[timestamp]	Time stamp of event in UTC time.
serverid		[uid]	Id of server where the event occurred.
deviceid		[uid]	Id of device which triggered the event, could be ommited or zero if it wasn't a device related event (i.e. SystemEvent).

error code	description	remarks	
0		Ok	Ok, no error
-18		General error	An undefined error occured

See Also

- Timestamp format.
- Event Types.
- Result list.
- getEvents.
- getEventTypes.

Changelog

- 2.7.1 – Replaced <type> tag with <name>
- 2.3.1 – Added

3.60 streamStatus

When a status changes in the system, it will be send (pushed) to the client. The system keeps pushing status changes until the connection is closed or another request is send over the current connection. The status changes are send as part of a multipart HTTP stream.

This command will return immediately with a multipart/x-mixed-replace content type and specific boundary. When a status changes, it will be send as xml message as denoted below, together with the boundary indicating the end of the multipart. If another request is send from the client, the ending boundary will be send. The 'type' parameter can have the following values: layout, hardware, system. Multiple values must be delimited by a comma.

Request

```
command=streamStatus&type=[string]
```

parameter	type	remarks
type	layout	sends layout status changes of the clients
type	hardware	sends hardware status changes of the RAID configuration of the servers
type	system	sends the system status of the servers in regular intervals

Changelog

- 2.5.1 – Added

3.61 streamVideo

When a connection is setup, the system will stream live jpeg images to the client until the connection is closed. The images are send as part of a multipart HTTP stream. The parameters deviceid and serverid must be used to determine the device. If serverid is omitted, the localhost server will be used. This command will return immediately with a multipart/x-mixed-replace content type and specific boundary. If no timestamp parameter is given, the stream will stream live footage. When a new image is available it will be send as image/jpeg together with the boundary.

Request

```
command=streamVideo&amp;deviceid=[uid]&amp;serverid=[uid]&amp;imagewidth=[number
&gt;0]&amp;imageheight=[number&gt;0]&amp;time=[timestamp]
```

parameter	type	remarks
deviceid	[uid]	unique device id.
serverid	[uid]	unique server id.
imagewidth	[number > 0]	optional image width.
imageheight	[number > 0]	optional image height.
time	[timestamp]	start playback from the given UTC timestamp (optional). If omitted, Stream will be live.

Response

```
Empty body. The header contains the multipart boundary.
```

error code	description	remarks
0	Ok	Ok, no error
-2	No device	No device found for the supplied cameraid/deviceid.
-21	Device not active	The device is not active in the current profile and cannot produce any results.

See Also

- getLiveImage
- streamEvents
- Timestamp format

Changelog

- 2.3.1 – Added

3.62 updateClip

Update the description of a clip of the given id. See addClip for the usage of the all the clip methods.

Note: This function requires VDG Sense version 2.5.4 or higher

Request

```
command=updateClip&id=[UID]&serverid=[UID]
```

Parameter	type	remarks
id	[UID]	unique clip id
serverid	[UID]	unique server id
description	[string]	description of the clip

Response

```
<result errorcode="0">
  <clip>
    <id>[UID]</id>
    <serverid>[UID]</serverid>
    <deviceid>[UID]</deviceid>
    <startdate>[timestamp]</startdate>
    <enddate>[timestamp]</enddate>
    <description>[string]</description>
    <filesize>[number]</filesize>
    <imagewidth>[number]</imagewidth>
    <imageheight>[number]</imageheight>
    <status description="[string]">[string]</status>
  </clip>
</result>
```

node	value	remarks
/result		
@errorcode	[number]	See error codes below
/result/clip		
id	[UID]	unique id
serverid	[UID]	unique server id
deviceid	[UID]	unique device id
startdate	[timestamp]	Clips with a (local) timestamp >= startdate are returned.
enddate	[timestamp]	Clips with a (local) timestamp < = enddate are returned.
description	[string]	A user defined description
filesize	[number]	Size of the clip in bytes

node	value	remarks	
imagewidth		[number]	Width of the clip in pixels
imageheight		[number]	Height of the clip in pixels
status		[string]	'pending', 'busy', 'finished', or 'failed '

error code	description	remarks	
0		Ok	Ok, no error
-40		Clip not found	Clip with given ID does not exist

See Also

- deleteClip
- getClips
- addClip
- downloadClip

Changelog

- 2.7.2 – Added

3.63 updatePerson

Update a person in the database. If an expiration date is given, the date will be checked and an error is returned if the date is not compatible.

Request

```
command=updatePerson&id=[UID]&firstname=[string]&lastname=[string]
```

Response

error code	description	remarks
0	Ok	Ok, no error
-4	Invalid Timestamp	Invalid datetime format. See Timestamp.
-18	General error	An undefined error occured

Changelog

- 2.6.1 – added

3.64 updateVideoTag

Update the video tag title and/or comment field. If a field is omitted, it's value remains unchanged.
Returns the changed videotag.

Request

```
command=updatevideotag&id=[UID]&title=[String]&comment=[String]
```

Response

```
<result errorcode="0">
 <videotag>
  <id>[UID]</id>
  <serverid>[UID]</serverid> <!-- if zero tag is for all devices -->
  <deviceid>[UID]</deviceid> <!-- if zero tag is for all devices -->
  <startdate>[timestamp]</startdate>
  <enddate>[timestamp]</enddate>
  <title>[String]</title>
  <comment>[String]</comment>
 </videotag>
</result>
```

node	value	remarks
/result		
@errorcode	[number]	See error codes below
/result/videotag		
id	[UID]	unique id
serverid	[UID]	unique server id
deviceid	[UID]	unique device id
startdate	[timestamp]	Tags with a (local) timestamp >= startdate are returned.
enddate	[timestamp]	Tags with a (local) timestamp < = enddate are returned.
title	[string]	A user defined title
comment	[string]	A user defined comment

error code	description	remarks
0	Ok	Ok, no error
-18	General error	An undefined error occured
-26	Videotag does not exist	Videotag does not exist or cannot be found

See Also

- getVideoTagList
- releaseVideoTag
- Examples – Video tagging

Changelog

- 2.4.3 – Added

4 References

References and examples

To look up specific commands or results, you can consult the information on this page. You will also find examples of various commands which can be given through the API.

4.1 Actions

This chapter gives a summary of the actions and their parameters as available at the moment of the creation of this document. It's purpose is to demonstrate some of the possibilities of VDG Sense. Actions can be executed on the system with the executeAction command.

To determine which actions are supported by a particular server the command getActionList should be executed. This command returns a xml structure with all supported actions and their parameters.

The commmands executeAction and getActionList are available as of version 2.2.1.

OverView

This table gives a brief summary of the actions. Refer to the section below for details.

parameter	type	remarks
camera.setDeviceSetting		Sets one device setting.
	deviceid	The id of the device.
	name	The name of a device setting to change.
	serverid	The id of the server where the camera is located.
	value	The new value for the device setting.
camera.sendDeviceCommand		Requests a URL from a device, but does not return any contents.
	deviceid	The id of the device.
	serverid	The id of the server where the device is located.
	url	The url of the device command to call.
server.tagVideoStart		Starts tagging video. Tagged video is not deleted until manually released.
	serverid	The id of the server where the device is located.
	deviceid	The id of the device.
	pretime	The number of seconds the start time will be moved in the past.
	title	The title of the VideoTag.
	comment	The comment of the VideoTag.
server.tagVideoStop		Stops tagging video. Tagged video is not deleted until manually released
	deviceid	The id of the device.
	serverid	The id of the server where the device is located.
viewer.executeMacro		Executes a macro on a specific viewer.
	divaid	The id of the viewer where the macro should be executed (What is DIVA?)

parameter	type	remarks
	name	The (unique) name of the macro to be executed.
server.executeMacro		Executes a macro on a specific viewer.
	divaid	The id of the server where the macro should be executed (What is DIVA?)
	name	The (unique) name of the macro to be executed.
viewer.selectPanel		

parameter	type	remarks

4.2 Camera Status

This table lists the results of the tag status from the command getCameraList.

Value	Description
CameraNormal	Camera operates normally.
ConnectionLost	The server has lost connection to the camera.
CameraBlocked	The camera doesn't send xml events anymore (doesn't apply to http!).
CameraInactive	The camera is deactivated in the current profile.
NoRecording	The camera is not recording.
NotActive	On demand streaming is enabled, and the camera is not streaming.

4.3 Events

Below is a summary of the events types that can be given to the command getEvents. These event types are the values for the comma separated string list of the tag types. All events that match one of the supplied types are returned (if they also match the other filters). If the parameter types is omitted from the request, all events are returned.

Use getEventTypes to obtain the server's actual supported event types list.

Version	Category	Remarks	Event types
1.3.2	Alarm	Alarm input events.	AlarmInput1, AlarmInput2, AlarmInput3, AlarmInput4, AlarmInputHigh, AlarmInputLow, CamAlarmInputHigh, CamAlarmInputLow
1.3.2	CarR	License plate found & match events.	CarRLicensePlateFound, CarRMatch
1.3.1	Connection	Connection loss and restore events.	ConnectionLost, Reconnect, ServerConnectionLost, VideoLost, VideoRestored
1.3.2	FaceR	Face found & match events.	FaceRFaceFound, FaceR Match
1.10.1	Generic	The generic event	GenericEvent
1.3.1	Motion	Motion events	Motion
1.3.1	ObjectR	All ObjectR events	ObjectRRule1, ObjectRRule2, ObjectRRule3, ObjectRRule4, ObjectRRule5, ObservionEvent
1.10.1	Other	Uncategorized events	StateRequest, UserButtonClick, UserButtonDblClick
1.3.2	Pos	Point of sale events (gas stations)	POSNozzleOut, POSNozzleIn, POSFillingStarted, POSFillingEnded
1.3.8	SceneR	All SceneR events	SceneRAngleChange, SceneRSceneLoss, SceneRAngleChangeReference, SceneRSceneLossReference
1.10.1	System	System specific events	NoRecording, DivaNotClosedGracefully, SystemStateOk, SystemStateError, ReRecording,

Version	Category	Remarks	Event types
			CameraBlocked, CameraNormal, SystemEvent, ClientConnectionLost, ClientConnectionRestored

Event Result Fields

Below is a summary of the events fields as result of command getEvents.

Version	Name	Remarks	
1.3.1		eventid	Unique event id.
1.3.1		eventtype	Number indicating the event type.
1.3.2		eventdescription	Description of the event.
1.3.1		eventvalue	Value of the event, like motion or alarm input.
1.3.8		eventtime	Time stamp of event.
1.3.2		serverid	Id of server where the event occurred.
1.3.2		serverdescription	Description of the server.
1.3.1		deviceid	Id of device which triggered the even.
1.3.2		devicedescription	Description of the device.

Event Result Values

Below is a summary of possible values for the event result fields. The eventvalue column describes the values for the event. Multiple values are separated by a comma. If a comma is part of the value, that value will be a double quoted (“value”) value.

Double quotes in a quoted value will be doubled (“value with ””quotes””, great!”).

Version	eventtype	eventdescription	eventvalue
1.3.0	0	AlarmInput1	n/a
1.3.0	1	AlarmInput2	n/a
1.3.0	2	AlarmInput3	n/a
1.3.0	3	AlarmInput4	n/a
1.3.0	4	ConnectionLost	n/a
1.3.0	5	Motion	Percentage of motion 0% – 100%
1.3.0	6	ObjectRRule1	Event text
1.3.0	7	ObjectRRule2	Event text
1.3.0	8	ObjectRRule3	Event text
1.3.0	9	ObjectRRule4	Event text
1.3.0	10	ObjectRRule5	Event text
1.3.0	11	SceneRCameraAngleChange	n/a
1.3.0	12	SceneRSceneLoss	n/a

Version	eventtype	eventdescription	eventvalue
1.3.0	13	SceneRCameraAngleChangeRef	n/a
1.3.0	14	SceneRSceneLossRef	n/a
1.3.0	15	FaceRMatch	Percentage of recognition 0% – 100% , Person name
1.3.0	16	Reconnect	n/a
1.3.0	17	FaceRFaceFound	Reliability of found face 0% – 100%
1.3.0	18	CarRLicensePlateFound	License plate, country, confidence (0 – 100%)
1.3.0	19	CarRMatch	Person name
1.3.0	20	reserved	
1.3.0	21	USBAlarmInput	Alarm number
1.3.0	22	USBAlarmInputHigh	n/a
1.3.0	23	USBAlarmInputLow	n/a
1.3.0	24	reserved	
1.3.0	25	reserved	
1.3.0	26	reserved	
1.3.0	27	reserved	
1.3.0	28	reserved	
1.3.0	29	reserved	
1.3.0	30	reserved	
1.3.0	31	reserved	
1.3.0	32	POSNozzleOut	Pump number
1.3.0	33	POSNozzleIn	Pump number
1.3.0	34	POSFillingStarted	Pump number
1.3.0	35	reserved	
1.3.0	36	reserved	
1.3.0	37	reserved	
1.3.0	38	reserved	
1.3.0	39	reserved	
1.3.0	40	reserved	
1.3.0	41	reserved	
1.3.0	42	DivaNotClosedGracefully	Timestamp
1.3.0	43	NoRecording	Timestamp of last recorded image
1.4.1	44	XMLKeepAlive	n/a
1.4.1	45	CameraBlocked	n/a
1.4.1	46	CameraNormal	n/a
1.4.1	47	StateRequest	n/a
1.6.1	48	ServerConnectionLost	n/a
1.6.1	49	AlarmInputHigh	n/a
1.6.1	50	AlarmInputLow	n/a
1.6.1	51	VideoLost	n/a
1.6.1	52	VideoRestored	n/a
1.6.1	53	SystemEvent	Error number
1.6.1	54	reserved	
1.6.1	55	reserved	
1.6.1	56	reserved	
1.6.1	57	reserved	
1.6.1	58	reserved	

Version	eventtype	eventdescription	eventvalue
1.6.1	59	reserved	
1.6.1	60	reserved	
1.6.1	61	reserved	
1.6.1	62	reserved	
1.8.1	63	SystemStateOk	n/a
1.8.1	64	SystemStateError	n/a
1.8.1	65	reserved	
1.8.1	66	reserved	
1.8.1	67	ReRecording	n/a
1.9.1	68	CamAlarmInputHigh	Alarm number
1.9.1	69	CamAlarmInputLow	Alarm number
1.10.1	70	GenericEvent	n/a
1.11.1	71	ObservisionEvent	shape
1.11.1	72	reserved	n/a
1.11.1	73	reserved	n/a
2.3.1	74	SystemFailover	n/a
2.3.1	75	MacroExecute	n/a
2.3.1	76	UserLoggedInEvent	n/a
2.3.1	77	OperatorEvent	n/a
2.3.1	78	TraficonEvent	n/a
2.3.1	79	PluginRule	n/a
2.6.1	80	reserved	n/a
2.6.1	81	SiquraVCA	triggered shape name
2.6.1	82	SystemRestore	information about the restored system
2.6.1	83	ServerConnectionRestored	
2.6.1	84	reserved	n/a
2.6.1	85	reserved	n/a
2.7.1	86	ClientConnectionLost	
2.7.1	87	ClientConnectionRestored	

4.4 Examples – Change camera name

The name of a camera can be changed with the executeAction command. This command executes a particular action on the server. The command requires a XML message as post data. This XML message holds all parameters for the action.

Set camera name

Sets the camera name of camera with serverid 8712348723918719871 and deviceid 3 to "MyCamera".

Command url:

```
http://<server>/command?command=executeAction
```

Post data:

```
<!--?xml version="1.0"?-->
 <action>
  <name>camera.setDeviceSetting</name>
  <parameters>
    <parameter name="serverid">8712348723918719871</parameter>
    <parameter name="deviceid">3</parameter>
    <parameter name="name">CameraName</parameter>
    <parameter name="value">MyCamera</parameter>
  </parameters>
 </action>
```

Result:

```
<result errorcode="0">
  <description>Ok</description>
 </result>
```

Check camera name

To see if the camera name is set correctly issue the command getDeviceList.

Command url:

```
http://<server>/command?command=getDeviceList
```

Result:

```
<result errorcode="0">
  <devices count="4">
    <device>
      <name>Lab</name>
      <serverid>7015498111931795223</serverid>
      <deviceid>1</deviceid>
      <status>ConnectionLost</status>
      <type>camera</type>
    </device>
    <device>
      <name>Dome</name>
      <serverid>7015498111931795223</serverid>
      <deviceid>2</deviceid>
      <status>CameraNormal</status>
      <type>camera</type>
    </device>
    <device>
      <name>192.168.203.80</name>
      <serverid>8712348723918719871</serverid>
      <deviceid>1</deviceid>
      <status>CameraNormal</status>
      <type>camera</type>
    </device>
    <device>
      <name>MyCamera</name>
      <serverid>8712348723918719871</serverid>
      <deviceid>3</deviceid>
      <status>CameraNormal</status>
      <type>camera</type>
    </device>
  </devices>
 </result>
```

Non existing device

What happens when we address a device that doesn't exists?

Command url:

```
http://<server>/command?command=executeAction
```

Post data:

```
<!--?xml version="1.0"?--> <action>
  <name>camera.setDeviceSetting</name>
```

```
  <parameters>
    <parameter name="serverid">8712348723918719871</parameter>
    <parameter name="deviceid">4</parameter>
    <parameter name="name">CameraName</parameter>
    <parameter name="value">MyCamera</parameter>
  </parameters>
 </action>
```

Result:

```
<result errorcode="-2">
  <description>No device</description>
 </result>
```

4.5 Examples – Determine if a device is a dome device

To see if a device has some kind of PTZ control use the command getOptions. We need to do this for every device we've obtained with the command getDeviceList.

Get options

For this example we only inspect the device with the name "Dome". We see that this device is a dome device and supports relative positioning.

Command url:

```
http://<server>/command?command=getOptions&serverid=7015498111931795223&deviceid
=2
```

Result:

```
<result errorcode="0">
  <options>
  <positioning>
    <pan>
      <relative>true</relative>
      <absolute>false</absolute>
      <position>false</position>
      <stop>true</stop>
    </pan>
    <tilt>
      <relative>true</relative>
      <absolute>false</absolute>
      <position>false</position>
      <stop>true</stop>
    </tilt>
    <zoom>
      <relative>true</relative>
      <absolute>false</absolute>
      <position>false</position>
      <stop>true</stop>
      <auto>false</auto>
    </zoom>
    <focus>
      <relative>true</relative>
      <absolute>false</absolute>
      <position>false</position>
      <stop>true</stop>
     <auto>true</auto>
    </focus>
    <iris>
      <relative>true</relative>
      <absolute>false</absolute>
```

```
      <position>false</position>
      <stop>false</stop>
      <auto>true</auto>
    </iris>
  </positioning>
  </options>
 </result>
```

4.6 Examples – Disable home preset

Most dome devices automatically goto a home preset position after some time of inactivity. We can set this timeout with the HomePresetTimeout device settting. A value of zero (0) will turn this functionality off.

Disable home preset

Disable the home preset for the device with the name "Dome".

Command url:

```
http://<server>/command?command=executeAction
```

Post data:

```
<!--?xml version="1.0"?-->
 <action>
  <name>camera.setDeviceSetting</name>
  <parameters>
    <parameter name="serverid">7015498111931795223</parameter>
    <parameter name="deviceid">2</parameter>
    <parameter name="name">homePresetTimeout</parameter>
    <parameter name="value">0</parameter>
  </parameters>
 </action>
```

Result:

```
<result errorcode="0">
  <description>Ok</description>
 </result>
```

What if the device isn't a dome

What would happen if we accidentally use a device that isn't a dome?

Command url:

```
http://<server>/command?command=executeAction
```

Post data:

```
<!--?xml version="1.0"?-->
 <action>
  <name>camera.setDeviceSetting</name>
  <parameters>
    <parameter name="serverid">7015498111931795223</parameter>
    <parameter name="deviceid">1</parameter>
    <parameter name="name">homePresetTimeout</parameter>
    <parameter name="value">0</parameter>
  </parameters>
 </action>
```

Result:

```
<result errorcode="-19">
  <description>No dome</description>
 </result>
```

4.7 Examples – Execute macro

The virtual matrix (macros), allows the system to perform some actions based on one or more signals. See virtual matrix in chapter Introduction. One signal, or a combination of signals, triggers a macro. This macro executes one or more actions.

The command executeAction has one action viewer.executeMacro which allows the client to execute a specific viewer macro on a specific viewer station.

Get the list with viewer macros.

First we need to get the list with defined macros. Only viewer macros are returned.

Command url:

```
http://<server>/?command=getMacroList
```

Result:

```
<result errorcode="0">
  <macros count="10">
    <macro>
      <macroid>8</macroid>
      <name>Events.Panel2</name>
      <type>viewer</type>
    </macro>
    <macro>
      <macroid>9</macroid>
      <name>Operator</name>
      <type>viewer</type>
    </macro>
    <macro>
      <macroid>10</macroid>
      <name>Events Bookmark.Panel2</name>
      <type>viewer</type>
    </macro>
    <macro>
      <macroid>11</macroid>
      <name>Events Bookmark.Panel14</name>
      <type>viewer</type>
    </macro>
    <macro>
      <macroid>12</macroid>
      <name>Layout  Events bookm</name>
      <type>viewer</type>
    </macro>
    <macro>
      <macroid>13</macroid>
      <name>Layout empty</name>
```

```
      <type>viewer</type>
    </macro>
    <macro>
      <macroid>16</macroid>
      <name>Layout 1.toLayout2</name>
      <type>viewer</type>
    </macro>
    <macro>
      <macroid>17</macroid>
      <name>Layout 2.toLayout1</name>
      <type>viewer</type>
    </macro>
    <macro>
      <macroid>18</macroid>
      <name>Switch Layout Layout1</name>
      <type>viewer</type>
    </macro>
    <macro>
      <macroid>19</macroid>
      <name>Switch Layout Layout2</name>
      <type>viewer</type>
    </macro>
  </macros>
 </result>
```

Executing a macro.

getMacroList in this example, we see the last two macros are used to switch to Layout 1 and Layout 2 respectively.

In this example we are going to execute the macro to switch the Layout to “Layout1” on the viewer with id 1237718987111238112.

This is only an example of course. Switching a layout could also be done with the command selectLayout. See the example “Select a camera on a server” for an example of this command.”>To execute a macro we must know the name of the macro to be executed. From the results of the getMacroList command we might be able to guess what that macro will do.

In real life situations it is best to give meaningful names to the macros. These names can be given by a defined convention. This allows the system and/or user to determine what a macro will do based on the name of the macro. These conventions should be made before configuring the system.

For example the convention used here is that a macro name that starts with “Switch Layout” is used to switch the layout of the server. The macro name ends with the name of the layout. In the results ofgetMacroList in this example, we see the last two macros are used to switch to Layout 1 and Layout 2 respectively.

This is only an example of course. Switching a layout could also be done with the command selectLayout. See the example “Select a camera on a server” for an example of this command.

Command url:

```
http://<server>/command?command=executeAction
```

Post data:

```
<!--?xml version="1.0"?-->
 <action>
  <name>viewer.executeMacro</name>
  <parameters>
    <parameter name="divaid">1237718987111238112</parameter>
    <parameter name="name">Switch Layout Layout1</parameter>
  </parameters>
 </action>
```

Result:

```
<result errorcode="0">
  <description>Ok</description>
 </result>
```

4.8 Examples – Select a camera on a server

This large example shows which commands to send to the management server to get a specific camera from a server displayed on a specific monitor on some viewer station. A typical network consists of a management server, one or more managed servers and one or more viewer stations (NMS), and of course one or more cameras and possible one or more I/O devices. Camera images are displayed in panels. Panels are part of a layout. Layouts can be displayed on one of the monitors attached to a viewer station. To see a particular camera on a server we first have to select a layout on a monitor of that server.

To select a camera on a panel we use the command selectCamera. This will show a particular camera on the selected panel on all servers that have the layout with layoutid selected. To use this command we need layoutid , serverid and deviceid . These can be obtained with the commands getLayoutList and getDeviceList.

To select a particular layout on a particular server we use the command selectLayout. This command requires a layoutid and a divaid. These can be obtained with the commands getLayoutList and getDeviceList.

Here are all the steps in the correct order.

Get list

First we start with obtaining a List. This gives us a list of all servers in the network with their divaid's. This id will be used further on to select the layout on the server with the command selectLayout.

Command url:

```
http://<server>/command?command=getDivaList
```

Result:

```
<result errorcode="0">
  <divas count="4">
    <diva>
      <divaid>7015498111931795223</divaid>
      <type>NVDR</type>
      <options></options>
    </diva>
    <diva>
      <divaid>8712348723918719871</divaid>
      <type>NVDR</type>
      <options></options>
    </diva>
    <diva>
      <divaid>1237718987111238112</divaid>
      <type>NMS</type>
      <options></options>
    </diva>
    <diva>
```

```
      <divaid>9890179877273119875</divaid>
      <type>NMS</type>
      <options></options>
    </diva>
  </divas>
 </result>
```

Get server options

We might want to get extra information from the servers, i.e. to see what description it has. This step is not necessary, but can be used to give the user extra information. You need to issue this command for each server.

Command url:

```
http://<server>/command?command=getServerOptions&serverid=7015498111931795223
```

Result:

```
<result errorcode="0">
  <serveroptions>
    <description>VDG Sense</description>
    <title>NVSP 1016</title>
    <version>1.14.5</version>
    <registrationnumber>vdg-sense</registrationnumber>
    <cameracount>16</cameracount>
    <servercount>1</servercount>
    <transcodingbandwidth>deprecated</transcodingbandwidth>
    <transcodingquality>99</transcodingquality>
    <webtranscodedimagesrequested>False</webtranscodedimagesrequested>
    <webdefaultimagewidth>640</webdefaultimagewidth>
    <webdefaultimageheight>480</webdefaultimageheight>
    <oldeststoredimagetime>deprecated</oldeststoredimagetime>
    <neweststoredimagetime>deprecated</neweststoredimagetime>
  </serveroptions>
 </result>
```

Get Layout list

Then we get the list with layouts. This gives us the layoutid's. This id will be used in the commands selectLayout and selectCamera

Command url:

```
http://<server>/command?command=getLayoutList
```

Result:

```
<result errorcode="0">
  <layouts count="3">
    <layout>
      <layoutid>1</layoutid>
      <name>Full screen Live</name>
      <panels>1</panels>
    </layout>
    <layout>
      <layoutid>3</layoutid>
      <name>Quad screen Live</name>
      <panels>4</panels>
    </layout>
    <layout>
      <layoutid>4</layoutid>
      <name>Playback</name>
      <panels>3</panels>
    </layout>
  </layouts>
 </result>
```

Get device list

We also need to get the device list. This gives us the serverids and deviceids. These will be used in the command selectCamera.

Command url:

```
http://<server>/command?command=getDeviceList
```

Result:

```
<result errorcode="0">
  <devices count="4">
    <device>
      <name>Lab</name>
      <serverid>7015498111931795223</serverid>
      <deviceid>1</deviceid>
      <status>ConnectionLost</status>
      <type>camera</type>
    </device>
```

```
    <device>
      <name>Dome</name>
      <serverid>7015498111931795223</serverid>
      <deviceid>2</deviceid>
      <status>CameraNormal</status>
      <type>camera</type>
    </device>
    <device>
      <name>192.168.203.80</name>
      <serverid>8712348723918719871</serverid>
      <deviceid>1</deviceid>
      <status>CameraNormal</status>
      <type>camera</type>
    </device>
    <device>
      <name>YourCamera</name>
      <serverid>8712348723918719871</serverid>
      <deviceid>3</deviceid>
      <status>CameraNormal</status>
      <type>camera</type>
    </device>
  </devices>
 </result>
```

Select a layout

Now we have all required information. We can select a particular layout on a monitor. This layout can be i.e. a full screen video, or a quad screen. Layouts can be freely configured in the system. We will use the first NMS from the results of getDivaList, the first monitor and we will use the Full screen Live layout.

Command url:

```
http://<server>/command?command=selectLayout&divaid=123771898711123811 2&monitori
ndex=0&layoutid=1
```

Result:

```
<result errorcode="0">
  <description>Ok</description>
 </result>
```

Select a camera

Finally we can select the camera. We want to see camera with name 'Dome' on the full screen live panel we

just selected on the first monitor of some server. The selectCamera command requires a Camera view parameter.

Command url:
Result:

```
http://<server>/command?command=selectCamera&cameraview=serverid:701549811193179
5223;deviceid:1&layoutid=1&panelindex=0
```

Result:

```
<result errorcode="0">
  <description>Ok</description>
 </result>
```

4.9 Examples – Set home preset timeout

To enable homing of a dome device and/or set the timeout period we use the same action as the example above, only use a value for the timeout in seconds.

Enable home preset

Goto home after 60 seconds inactivity

Command url:

```
http://<server>/command?command=executeAction
```

Post data:

```
<!--?xml version="1.0"?-->
 <action>
  <name>camera.setDeviceSetting</name>
  <parameters>
    <parameter name="serverid">7015498111931795223</parameter>
    <parameter name="deviceid">2</parameter>
    <parameter name="name">homePresetTimeout</parameter>
    <parameter name="value">60</parameter>
  </parameters>
 </action>
```

Result:

```
<result errorcode="0">
  <description>Ok</description>
 </result>
```

4.10 Examples – Video tagging

Normally the system will delete the oldest video and metadata to make place for new video. When deleted this video is lost. To prevent this in case something happens the video (and metadata) can be tagged so it will not be deleted.

The actions server.tagVideoStart and server.tagVideoStop respectively starts and stop tagging video. Once the video is tagged it will not be deleted. The user should use the system to untag the video's manually. Untagged video will then be deleted normally. These actions can be send to the system with the executeAction command

Start tagging video

Starts tagging video so recordings will not be deleted. Without optional parameters serverid and deviceid, all servers will tag their recorded video (and accompanying metadata). An extra parameter pretime can be added, this value will move the start time of the videotag backwards in time, with a maximum of 24 hours. The pretime value needs to be defined in seconds.
Tagged data will not be deleted by the system. The user is responsible to untag the data on systems.

While the systems becomes full with tagged video several "storage full" SystemEvents will be fired so the client can be notified of this. Events will be fired on 30, 60, 90 and 100 percent of locked storage space.

Command url:

```
http://<server>/command?command=executeAction
```

Post data:

```
<!--?xml version="1.0"?-->
 <action>
  <name>server.tagVideoStart</name>
<parameters>
   <parameter name="serverid">8712348723918719871</parameter>
   <parameter name="deviceid">3</parameter>
   <parameter name="pretime">60</parameter>
</parameters>
</action>
```

or

```
http://<server>/command?command=executeaction&actionname=server.tagVideoStart&se
rverid=8712348723918719871&deviceid=3&pretime=60
```

(if parameters serverid and deviceid are not added, the video of all cameras will be tagged)

Result:

```
<result errorcode="0">
  <description>Ok</description>
 </result>
```

Stop tagging video

Stops tagging video. The untagged video and metadata will be normally deleted (oldest images first during the cyclical deleting scheme). The previously tagged data will remain tagged and will not be deleted by the system. The user is responsible to untag the data on the system.

Command url:

```
http://<server>/command?command=executeAction
```

Post data:

```
<!--?xml version="1.0"?-->
 <action>
  <name>server.tagVideoStop</name>
<parameters>
   <parameter name="serverid">8712348723918719871</parameter>
   <parameter name="deviceid">3</parameter>
</parameters>
</action>
```

or

```
http://<server>/command?command=executeaction&actionname=server.tagVideoStop&ser
verid=8712348723918719871&deviceid=3
```

Result:

```
<result errorcode="0">
  <description>Ok</description>
 </result>
```

4.11 License Functions

Some functions require an API license option. This option is available in the pro-version of our license model.

4.12 Result Codes

This table list the possible result codes as returned from http or xml commands. A value of 0 means no error. Negative values means some kind of error, stated in the table below. The meaning of positive return values is defined by the command the answer belongs to.

Error code	Code message	Description	
0		Ok	Ok, no error.
-1		Not implemented	Function not implemented.
-2		No device	No device found for the supplied cameraid/deviceid.
-3		Error historical image	Error retrieving historical image.
-4		Invalid Timestamp.	Invalid datetime format. See Timestamp.
-5		Error live image	Error retrieving live image.
-6		Error historical Data	Error retrieving historical image data.
-7		Preset out of range	Preset number out of range. Valid numbers are [1..64].
-8		No FaceR	System or device not set for face recognition.
-9		No record	The record search is not found
-10		Not enough parameters	There are not enough parameters supplied for this function.
-11		No image	There is no image found to return.
-12		No Diva	There is no server with the supplied id (What is DIVA?).
-13		No monitor	There is no monitor with the supplied id.
-14		No layout	There is no layout with the supplied id.
-15		No panel	There is no panel with the supplied id.
-16		No multiLayout	There is no multilayout with the supplied id.
-17		No server	There is no server with the supplied id.
-18		General error	An undefined error occured
-19		No dome	The device is not a dome device.
-20		No CarR	System or device not set for license plate recognition.
-21		Device not active	The device is not active in the current profile and cannot produce any results.
-22		Value is out of range	The supplied value is out of range.
-23		An empty value is not allowed	The supplied value is out of range.
-24		Invalid action	The action is not a valid action.
-25		No macro	There is no macro with the supplied id.
-26		Action name does not exist	The supplied action name does not exist.
-27		Invalid action parameters	At least one action parameter

Error code	Code message	Description	
			does not exist.
-28		Not a Diva client	There is no client with the supplied divaid.
-29		User is not allowed to perform command	The (api) user logged in to the server does not have sufficient rights to perform the command.
-30		VideoTag does not exist	The supplied videotag id does not exist.
-31		Invalid Pretime	
-32		No id parameter found	
-33		No serverId parameter found	No serverID parameter given.
-34		No deviceID parameter found	No deviceID parameter given.
-35		No startdate parameter found	No startdate parameter given.
-36		No enddate parameter found	No enddate parameter given.
-37		Enddate before startdate error	Enddate parameter is before the startdate parameter.
-38		Time difference above limit	Enddate is >5 minutes than startdate.
-39		File not found	
-40		Clip not found	Clip with given ID does not exist
-41		Start time in the future	Startdate exceeds enddate parameter
-42		No response from server	
-43		No description parameter found	
-44		Clip is pending or busy	
-45		Database offline	The database cannot be reached
-46		Image size is not valid	
-47		Failed after multiple tries to create clip	
-48		No video data for timestamp	No video data for the given timestamp
-49		Output size exceeds retention size	The exported clip filesize is bigger than the maximum retention size
-50		Storage location doesn't exist	The given Filepath parameter refers to an invalid location.
-51		Clip build failed	
-52		Video tag is still active	Tag cannot be exported while active
-53		Device connection lost	Device has no connection

4.13 Result Lists

Some commands like getEvents and getImageSequence are used to obtain a list with recorded data from the server. This list can be limited by supplying a starttime, an endtime and a limit argument in the http command url.

A maximum of 25 events will be returned by default. This value can be changed by the limit parameter. A positive limit will return the first found records. A negative limit will return the last found records.

If the endtime lies before the starttime, the resulting list will run from starttime back to endtime. This is exactly the same result as when the command was sent with the start and end time reversed and the limit negative.

If the endtime lies before the starttime and a negative limit is supplied, the result is the same as if the endtime lies past the starttime and the limit is positive.

The results are always sorted in ascending time. Because of the internal working of the server, it could happen that the order of eventid's differs from the order of eventtime's. For example, the ordered by eventtime result list could have an eventid sequence of 1, 2, 3, 5, 4, 6.

Examples:

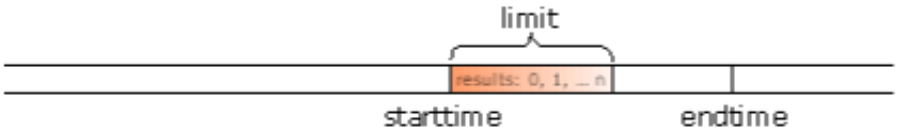


Figure 1: Endtime past starttime, positive limit

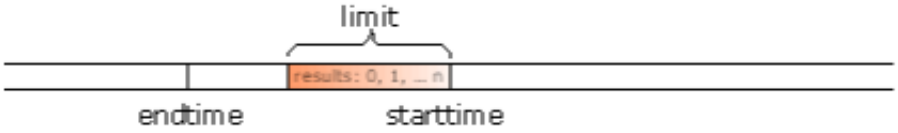


Figure 2: Endtime before starttime, positive limit

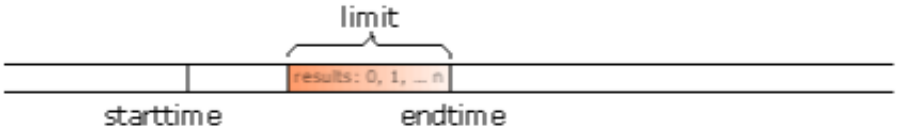


Figure 3: Endtime past starttime, negative limit

Figure 4: Endtime before starttime, negative limit

Figure 5: StartId with positive limit

Figure 6: StartId with negative limit