

**Digifort HTTP API Interface
API 1.10.0**

For Digifort 7.4.0.0

Rev. A

Index

Part I Overview	8
1 Description.....	8
2 History.....	8
Part II Definitions	16
1 General notations.....	16
General abbreviations	16
Style conventions and general syntax of the URL	16
Part III Interface	18
1 General arguments.....	18
2 Types of variables.....	18
3 Filter masks.....	19
4 System Objects.....	21
5 Event Types.....	22
6 Event Actions.....	25
7 MIME Types.....	33
Media Types	34
Media Frame Types	35
8 Authentication.....	35
Basic HTTP authentication	35
Basic authentication with parameters	36
Safe authentication	36
Creating an authentication session.....	37
Calculating the authentication data	38
Maintaining an authentication session.....	39
9 Responses.....	39
Response in text	39
List of parameters in text.....	40
Response in XML	40
List of parameters in XML.....	41
Response in JSON	42
Default parameters of return	42
Return codes	43
Extra HTTP Headers	44
Part IV API Groups	47
1 Version.....	47
Requesting the version of API	47
2 Server.....	48
Requesting the machine code of the server	48
Requesting data about the server	49

Requesting licensing information	51
Requesting data of usage of the server	55
Requesting status of Master / Slave	56
3 Cameras.....	59
Requesting the list of cameras	59
Requesting the list of camera groups	63
Requesting the status of the cameras	65
Activating / Deactivating cameras	70
Requesting a live image (Snapshot)	71
Requesting a live media stream	73
Requesting a live video stream (MJPEG)	78
Requesting a live video stream (MP4)	82
Requesting a live audio stream	85
Sending a live audio stream	87
Playback	89
Requesting a recorded image (Snapshot).....	89
Requesting a recorded media stream.....	91
Requesting a recorded media stream (MJPEG).....	95
Requesting a recorded media stream (MP4).....	99
Requesting the timeline data.....	102
Exporting.....	106
Starting an export session.....	107
Dow nloading exported files.....	113
Closing an export session.....	114
PTZ	115
Simple	115
Relative	116
Absolute	118
Area Zoom.....	119
Simultaneous.....	121
Continuous.....	122
Auto Focus	124
Auto Iris	124
Menu control.....	125
Presets	126
List of presets	126
Call a preset	128
Set a preset	129
Call a pattern.....	130
Windshield w iper.....	131
Auxiliary	132
List of Auxiliaries	132
Call an Auxiliary	134
Patrol	135
Requesting the list of patrols	135
Requesting the status of PTZ patrol.....	137
Controlling PTZ Patrol.....	139
Supported commands.....	140
Requesting the status of PTZ	142
I/O	147
Requesting the list of input events.....	147
Requesting the status of the input ports.....	150
Requesting the status of input events.....	152
Requesting the status of output ports	154

Requesting the list of output actions.....	156
Triggering an output action.....	158
Requesting the status of virtual ports.....	159
Setting the state of a virtual port.....	161
Motion detection	162
Notifying motion detection.....	162
Manual events	163
Requesting the list of manual events.....	163
Triggering a manual event.....	167
Bookmarks	170
Adding a new bookmark.....	170
Searching bookmarks.....	171
Privacy Mode	174
Controlling privacy mode.....	174
Requesting the status of privacy mode.....	175
Recording	176
Notifying an event for event recording.....	176
Locks	177
Searching Recording Locks.....	177
Adding a new Recording Lock.....	180
Creating a new Self-Healing Job.....	181
4 I/O Devices.....	181
Requesting the list of I/O devices	181
Requesting the status of I/O devices	185
Activating / Deactivating I/O devices	187
I/O	188
Requesting the list of input events.....	188
Requesting the status of input ports.....	191
Requesting the status of input events.....	193
Requesting the status of the output ports	195
Requesting the list of output actions.....	197
Triggering an output action.....	199
Requesting the status of virtual ports.....	200
Setting the state of a virtual port.....	202
5 Users.....	203
Requesting the list of users	203
Requesting the list of groups	208
Requesting the list of current connected users	213
Blocking / Unblocking users	216
6 Screenstyles.....	217
Requesting the list of screenstyles	217
Requesting the image of a screenstyle	220
7 Screen views.....	221
Requesting the list of screen views of the user	221
Requesting the list of public screen views	223
Reading the contents of an user screen view	226
Reading the contents of a public screen view	229
8 Maps.....	232
Requesting the list of maps	232
Activating / Deactivating maps	235
9 Operational Maps.....	236
Requesting the list of Operational Maps	236

Activating / Deactivating Operational Maps	238
Requesting the list of Custom Objects	240
Updating coordinates and memo of a Custom Object	243
Activating / Deactivating Custom Objects	244
10 Global Events.....	245
Requesting the list of Global Events	245
Triggering a Global Event	249
Requesting the Event Actions of a Global Event	252
Activating / Deactivating Global Events	253
11 Scheduled Events.....	254
Requesting the list of scheduled events	254
Activating / Deactivating scheduled events	258
12 Virtual matrix.....	259
Requesting the list of active monitors	259
Requesting the list of seen (Older) monitors	264
Loading an empty screen style in a monitor	267
Displaying an object in a monitor	268
Displaying an user screen view in a monitor	269
Displaying a public screen view in a monitor	270
Removing objects from a monitor	271
Starting media playback in a monitor	272
13 Events.....	273
Searching for events records	273
Monitoring server events	277
14 LPR.....	282
Requesting the list of LPR Configurations	282
Requesting the status of LPR Configurations	284
Requesting the list of surrounding cameras of an LPR Configuration	287
Activating / Deactivating LPR configurations	289
Searching for LPR records	290
Requesting the data of an LPR record	293
Requesting the image of an LPR record	293
Requesting the data of the last LPR record	294
Requesting the image of the last LPR record	295
Triggering the recognition	296
Monitoring the LPR events	297
Processing an external image	305
Querying data from a license plate	305
LPR Record Details	306
Zones	313
Requesting the list of LPR Zones.....	313
Requesting the statistics of LPR Zones.....	319
Seaching LPR Zone Records.....	321
Activating / Deactivating LPR Zones.....	326
Zone Groups.....	327
Requesting the list of LPR Zones Groups.....	327
Requesting the statistics of LPR Zone Groups.....	333
Activating / Deactivating LPR Zone Groups.....	335
15 Analytics.....	336
Requesting the list of Analytics Configurations	336
Requesting the status of Analytics Configurations	340
Activating / Deactivating Analytics configurations	343

Requesting the value of counters in an Analytics Configuration	344
Resetting the value of counters of an Analytics Configuration	346
Searching for Analytics records	347
Requesting the metadata of a record	353
16 Web Pages.....	355
Requesting the list of Web Pages	355
Activating / Deactivating Web Pages	357
17 Audit.....	358
Searching for audit records	358
Monitoring the audit records	363
18 Failover.....	370
Requesting the list of failover monitors	370
Requesting the status of failover monitors	372
Activating / Deactivating failover monitors	375
19 Device Models.....	377
Media Devices	377
Requesting the list of supported media devices.....	377
Requesting the data of a specific media device model.....	381
I/O Devices	385
Requesting the list of supported I/O devices.....	385
Requesting the data of a specific I/O device.....	386
20 RTSP.....	388
Requesting the RTSP server settings	388
Requesting the status of RTSP server	389

Index**0**

Part

I

1 Overview

1.1 Description



This document specifies the HTTP base external programming interface of Digifort system.

The interface based on HTTP provides functionalities for accessing data of all the objects of the server, inserting data and alarms and request live and recorded images.

1.2 History

Version	Comments
1.0.0	First public version
1.1.0	<ul style="list-style-type: none"> - Added Auxiliary PTZ command support - Added SpotNumber parameter to send an object to the virtual matrix - Added server event monitoring - Updated the compatibility of the commands in order to reflect the addition of some features in Explorer and Standard versions - Added a new variable type: APITimestamp - Added new PTZ command: Area Zoom - Added command to request a live MJPEG video stream - Added session with LPR commands
1.2.0	<ul style="list-style-type: none"> - The command to create a safe authentication session no longer needs the mandatory AuthSession=0 parameter - The Speed parameter from area zoom command is no longer mandatory - Added a new camera motion detection section - The command to show an object on a virtual matrix monitor will now accept the "Analytics Configuration" and "LPR Configuration" object types - Added 2 new object types in the command to retrieve the list of active monitors of virtual matrix: Analytics Configuration and LPR Configuration - The command to monitor the server events will now accept the parameter KeepAliveInterval to control the sending of keep-alive events - The command to monitor the LPR events will now accept the parameter KeepAliveInterval to control the sending of keep-alive events - Added new command to request a live media stream without transcoding
1.2.1	<ul style="list-style-type: none"> - Added new analytics events to the command to monitor server events
1.3.0	<ul style="list-style-type: none"> - Added a new result code to the API commands. The return code 10 will now be used when the user does not have enough rights to access a given command - The command to return a list of user screen views was changed from GetScreenViews to GetUserScreenViews

	<ul style="list-style-type: none"> - Added a new command to return the list of public screen views - Added a new command to display an user screen view in the virtual matrix - Added a new command to display a public screen view in the virtual matrix - The virtual matrix commands will not require the authentication with Admin user anymore, instead, the commands will now only require the authentication of an user with virtual matrix operating rights - The command to request the list of global events will not require the authentication with Admin user anymore, instead, the command will now only require the user authentication. - The command to trigger a global event will not require the authentication with Admin user anymore, instead, the command will now only require the authentication of an user with rights to trigger global events and with rights to access the global event to be triggered. - The commands to request the list of maps and request the list of maps of an user were merged, now when requesting the list of maps the command will return only the list of maps that the user has access to and will not require the authentication with Admin user anymore. To keep compatibility with old API versions, the command <code>GetMapsViewRight</code> will still be able to be invoked, but it will be internally redirected to <code>GetMaps</code> instead. - The commands to request the list of cameras and request the list of cameras of an user were merged, now when requesting the list of cameras the command will return only the list of cameras over which the user has rights for live view or playback and will not require the authentication with Admin user anymore. To keep compatibility with old API versions, the command <code>GetCamerasViewRight</code> will still be able to be invoked, but it will be internally redirected to <code>GetCameras</code> instead. - Added a new return code to cameras. The code 10001 can be returned when calling the routines <code>GetSnapshot</code>, <code>GetMediaStream</code> and <code>GetJPEGStream</code> if the camera is under privacy mode. - Added a new command to request the data of the last LPR record - Added a new command to request the image of the last LPR record - The analytics events ANALYTICS_FOREIGN_OBJECT and ANALYTICS_MISSING_OBJECT were changed to ANALYTICS_ABANDONED_OBJECT and ANALYTICS_REMOVED_OBJECT in monitoring server events
1.4.0	<ul style="list-style-type: none"> - Added a new command to trigger the license plate recognition by using an external sensor - Added new return code for LPR - Added the new parameter <code>OverrideShowCameras</code> to the command to trigger a global event - Added a new command to request the status of the cameras - Added a new command to request the status of the I/O devices - Added new parameter containing the media profiles of the cameras, on the command to request the list of cameras - Added a new section with commands related to RTSP server - Added a new parameter containing the name of LPR configuration on LPR records
1.5.0	<ul style="list-style-type: none"> - Added a new parameter to the command to trigger global events - The analytics events ANALYTICS_SMOKE and ANALYTICS_FIRE were added to monitoring server events - The software edition is now being informed in the command to request server information - Added commands to control Manual Events from cameras - The object type LPR_LIST was changed to LPR_EVENT in monitoring server events - The event type LPR_PLATE_IN_LIST was changed to LPR in monitoring server events - The command to request a live image will not return the result "Data not ready yet" anymore, it will return the image directly when it is ready - Added a new description field to the current object of virtual matrix monitors - The types of variables APITime and APITimestamp now includes milliseconds - The following commands were affected by changes to variables APITime and

	<p><u>APITimestamp:</u></p> <p><u>Server/GetInfo</u></p> <p><u>Cameras/Playback/GetSnapshot</u></p> <p><u>LPR/GetRecordData</u></p> <p><u>LPR/GetLastRecordData</u></p> <p><u>LPR/MonitorEvents</u></p> <p><u>Events/Monitor</u></p> <ul style="list-style-type: none"> - Added new <u>types of variables</u> - Added support to <u>filters mask</u> that can be used to filter results of some API commands - The following commands now supports <u>filters mask</u>: <p><u>Cameras/IO/GetOutputActions</u></p> <p><u>Cameras/ManualEvents/GetManualEvents</u></p> <p><u>Cameras/GetCameras</u></p> <p><u>Cameras/GetStatus</u></p> <p><u>AlarmDevices/IO/GetOutputActions</u></p> <p><u>AlarmDevices/GetAlarmDevices</u></p> <p><u>AlarmDevices/GetStatus</u></p> <p><u>Users/GetUsers</u></p> <p><u>ScreenViews/ GetUserScreenViews</u></p> <p><u>ScreenViews/ GetPublicScreenViews</u></p> <p><u>Maps/GetMaps</u></p> <p><u>GlobalEvents/GetGlobalEvents</u></p> <p><u>LPR/GetLPRConfigurations</u></p> <p><u>VirtualMatrix/GetActiveMonitors</u></p> <p><u>VirtualMatrix/GetScreenMonitors</u></p> <ul style="list-style-type: none"> - Added commands to control camera bookmarks - Added new command to <u>start media playback in a monitor of virtual matrix</u> - New types of events for audio detection (AUDIO_LEVEL_LOW, AUDIO_LEVEL_HIGH) added to <u>server events monitoring</u> - Added new commands for media playback - New <u>variable type</u> Double - New information of recording hours and estimative of recording hours on <u>camera status</u> - New information with connection address and port for the <u>list of cameras</u> - New events of FAILOVER and FAILBACK in <u>server events monitoring</u> - Added values of reliability and hit to the commands to request date of an <u>LPR record</u> and data of the <u>last LPR record</u> - Added command to <u>search for LPR records</u> - Added command to <u>search for Analytics records</u>
1.6.0	<ul style="list-style-type: none"> - The type ALARM_DEVICE was changed to IO_DEVICE in <u>server events monitoring</u> - Changed all commands of alarm devices to I/O devices - Added Latitude and Longitude values to cameras in the <u>list of cameras</u> command - New event of COMMUNICATION_RESTORED in <u>server events monitoring</u> - New commands to control camera Privacy Mode - New commands to control camera PTZ Patrol - New command to <u>search for events</u>
1.7.0	<ul style="list-style-type: none"> - Added new parameters <code>OverrideEmailMessageFormat</code>, <code>OverrideOperatorMessageFormat</code> to command to <u>trigger global events</u> and <u>trigger manual events</u> - Added a new command to <u>start or end the recording by event</u> when camera is set to record by event - Added a new command to <u>request the status of LPR Configurations</u> - Added a new command to <u>request the list of Analytics Configurations</u> - Added a new command to <u>request the status of Analytics Configurations</u>

	<ul style="list-style-type: none"> - Added a new command to load an empty screen style in a monitor of Virtual Matrix - Added SourceStorage parameter for all playback commands - Updated return codes - Added a new command to read the contents of an user screen view - Added a new command to read the contents of a public screen view - Added audio support to the command to request live media stream from a camera. A new video parameter was added to this command as well. - Added a new command to request live audio from a camera - Added a new command to send live audio to a camera for two-way audio communication - Added a new command to retrieve the value of counters of analytics configurations - Added a new command to reset the value of counters of analytics configurations - Added EDITION field to GetAPIVersion - Added support for H.265 frames - Added new commands to query supported device models data - Added new command to set a camera preset using current camera position - Added new commands to activate / deactivate Cameras, I/O devices, Global Events, Maps, LPR Configurations and Analytics Configurations - Added new command to block / unblock users - Added support for JPEG playback with rendered metadata
1.8.0	<ul style="list-style-type: none"> - Added new commands for exporting media in .MP4 format - Possibility to add extra HTTP headers to responses - Added ACTIVE field to the following commands: <ul style="list-style-type: none"> - Requesting list of cameras - Requesting list of global events - Requesting list of I/O devices - Requesting list of maps - Requesting list of analytics configurations - Requesting list of LPR configurations - Added new commands to manage scheduled events - Added new commands to manage failover monitors - Added new command to return the status of all PTZ cameras - Added new command to return the list of PTZ auxiliaries from a camera - Added new command to return the list of connected users - Added new command to search for audit records - Added new command to monitor audit records - The command to return the image of a screenstyle now supports AUTO and TIMER - Added new parameter DoNotReplaceSameObj to Virtual Matrix ShowObject - Added new parameter DoNotReloadSameStyle to Virtual Matrix LoadScreenStyle - Added new fields ACTIVETIME, INACTIVETIME, RECORDINGFPS, USEDIDISKSPACE to the command to retrieve the status of cameras - Added new field DATA to the command to retrieve the list of screen styles - Added a new command to retrieve the timeline data of recorded footage - Added new field OBJECTRIGHTS to the command to retrieve the list of users - Added new field STATE to the command to retrieve the status of PTZ Patrol - Added new action STOP to the command to control PTZ Patrol - Added new argument LPRConfiguration to the commands of retrieving the last LPR record and last LPR image - Added SERVERID parameter to the command to retrieve the server's machine code - Added parameters PROFILE and CUSTOMPROFILE to the commands to retrieve a camera snapshot and live video stream in MJPEG - Added new fields CURRENTSCREENSTYLEDATA and CURRENTSCREENVIEWOBJECTS to the command to retrieve the list of virtual matrix monitors

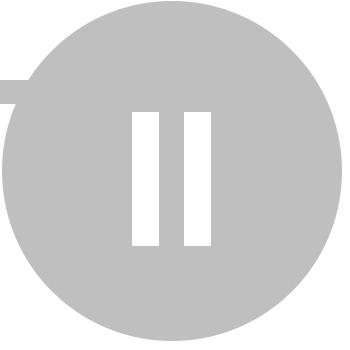
	<ul style="list-style-type: none"> - Added new command to retrieve the configured actions of a global event - Requesting the settings of RTSP server does not need Admin authentication anymore
1.9.0	<ul style="list-style-type: none"> - Added JSON response format - Added new System Object Types - Added many new Event Types - Added MxPEG frames to MIME Types - Added new Return codes - Removed the requirement of Admin login for almost all commands throughout the API. User authentication with specific rights will be required in lieu of Admin authentication - The commands for screenstyles now works on all editions - Added new APILatLn type for Latitude / Longitude representation throughout the API - Added Latitude/Longitude fields to List of Global Events - Added Latitude/Longitude fields to List of Scheduled Events - Added Latitude/Longitude fields to List of Maps - Added Latitude/Longitude fields to List of I/O Devices - Added ShowSnapshots to Event Actions - Added SendAudioClip to Event Actions - Added OverrideShowSnapshots to Trigger Global Events - Added OverrideLatitude and OverrideLongitude to Trigger Global Events - Added OverrideShowSnapshots to Trigger Global Manual - Added OverrideLatitude and OverrideLongitude to Trigger Global Manual - Added command to Request the list of User Groups - Added list of groups to Request the list of Users - Added Memo field to Request the list of Users - Added Memo field to Request the list of Cameras - Added command to Request the list of Camera Groups - Added Group field to Cameras List - Added Actions field to List of Global Events - Added Actions field to List of Manual Events - Added Actions field to List of Scheduled Events - Added command to retrieve the list of I/O Input Events of a Camera - Added command to retrieve the list of I/O Input Events of an I/O Device - Added command to search for Recording Locks - Added command to add Recording Locks - Added new analytics types to Event Types and Analytics Records - Added command to retrieve information about server licensing - SOUNDINDEX parameter of PlaySound from EventActions was changed for FILENAME and is now a string value containing the name of the sound file - LPR_CONFIGURATION and ANALYTICS_CONFIGURATION will now trigger COMMUNICATION_FAILURE and COMMUNICATION_RESTORED events - Added TimestampHeader parameter to commands for requesting media stream, video stream (JPEG) and audio stream - Added many new fields to the list of supported media devices - Added new command to query the list of supported I/O devices - Added Audio and FilenameFormat parameters for video exporting - Added DateTime, UTCDateTime and ServerType to Server Info - Added CAMERA_NOT_WORKING status to LPR and Analytics Status - Added command to retrieve the status of Master / Slave - Added RTSPS and RTSPSPORT fields to RTSP Config - Added new object types to Audit - Added new command to retrieve the list of Web Pages - Added new field MetadataPresent to the command to search analytics records

	<ul style="list-style-type: none"> - Added new command to Request the metadata of an analytics record - Added ObjectDUID parameter to Display an Object in the Virtual Matrix command - Added commands to Manage Virtual Ports of I/O Devices - Added commands to Manage Virtual Ports of Cameras - Added new command to Create a New Edge Recording Self-Healing Job - Added PositionMonitoring field to Supported PTZ Commands - Added Position Monitoring for Camera PTZ Status which allows for the retrieval of the current position (Pan/Tilt/Zoom) of the camera - Added fields ConfiguredToRecord and WrittingToDisk to Camera Status - Added new parameter KeepAspectRatio to the commands to Get Live Snapshot, Get Live MJPEG Stream, Get Playback Snapshot, Get Playback MJPEG and Start Export - Added many more fields to LPR Record Details - Added IncludePlateDetails, IncludeImage, IncludePlateCropImage parameters to Monitoring the LPR Events - Added PlateCrop parameter to the command to Request the Image of an LPR Record and Request the Image of the Last LPR Record - Added new command to Process an External Image on LPR - Added new command to Query License Plate Data on LPR - Added DeviceEvent and ServerEvent types to Event Search - Added SourceStorage=Edge parameter for Edge Playback to the commands of Requesting a recorded Media Stream and Requesting a recorded JPEG Stream
1.9.1	<ul style="list-style-type: none"> - Added AAC support for Requesting a live audio stream - Added support for audio for Requesting Live JPEG Stream - Added support for audio for Requesting a recorded JPEG Stream - Added new field Active to the List of Operational Maps
1.10.0	<ul style="list-style-type: none"> - BREAKING CHANGE: XMLHttpRequest event action has been replaced by XMLHttpRequests, containing a list of HTTP Actions - Added new Event Action to Show Playback Loop - Added SuppressRepeatedPlates and WithLPRBridgeData parameter for Searching for LPR Records command - Added new EventAction for SendPushNotification - Added command to Clear a Monitor of the Virtual Matrix - Added LPR Zone and Group of LPR Zone object types to the commands to Show an Object in the Virtual Matrix, Request list of users, Request list of groups and Request list of current connected users, Searching for Events Records - Added VehicleModel and WithLPRBridgeData to LPR commands - Added LPR_BRIDGE module to Server Licensing Information - Changed LPR Middleware field for LPR Bridge on LPR Record Details - Added RuleName to Searching for Analytics Records - Activation of Operational Maps now also accepts object names, not only DUIDs - Added Activation for Web Pages - The path of the API to create a new Download Job for Self-Healing has been changed from /Cameras/EdgeRecording/CreateDownloadJob to /Cameras/Recording/SelfHealing/CreateDownloadJob - Added Operational Map Custom Objects commands - Some Virtual Matrix commands now support sending the command to multiple monitors at the same time - Added ObjectDUID to Event Actions - Added Object DUID to Event Monitoring - Added AnalyticsEvents field to List of Analytics Configurations - Added Metadata, RenderInfo, WatermarkText, WatermarkColor, WatermarkSize and WatermarkPosition parameters to Live Image (Snapshot)

	<ul style="list-style-type: none"> - Added Metadata, RenderInfo, WatermarkText, WatermarkColor, WatermarkSize and WatermarkPosition parameters to Live Video Stream (MJPEG) - Added Metadata, RenderInfo, WatermarkText, WatermarkColor, WatermarkSize and WatermarkPosition parameters to Recorded Image (Snapshot) - Added RenderInfo, WatermarkText, WatermarkColor, WatermarkSize and WatermarkPosition parameters to Recorded Media Stream (MJPEG) - Added Metadata, RenderInfo, WatermarkText, WatermarkColor, WatermarkSize and WatermarkPosition parameters to Media Exporting - Added TriggerGlobalEvents and SendToVirtualMatrix actions on Event Actions - Added ObjectTypeID field to some Event Actions - Added Metadata field to Events Monitoring. Metadata will contain the list of variables with extra information on the event - Added camera connection information to Camera Status - Added analytics record sorting by Rule Name on Analytics Records Search - Added DUID support for Global Events - Added OverrideShowPlaybackLoop to Triggering Global Events and Triggering Manual Events - Added command to retrieve an MP4 stream for Live and Playback - Added commands for LPR Zones and LPR Zone Groups
--	---

API Version	System Version
1.0.0	6.3.0.0
1.1.0	6.4.0.0
1.2.0	6.5.0.0 - 6.5.0.1
1.3.0	6.6.0.0
1.4.0	6.7.0.0
1.5.0	7.0.0.0
1.6.0	7.1.0.0
1.7.0	7.2.0.0
1.8.0	7.2.1.0
1.9.0	7.3.0.0
1.9.1	7.3.0.2
1.10.0	7.4.0.0

Part



II

2 Definitions

2.1 General notations

2.1.1 General abbreviations

The following abbreviations are used in this document

Abbreviation	Description
N/A	Not applicable - The feature/parameter/value is not used in the specified task
URL	Uniform Resources Location (URL) is a compact string that represents a feature available in Internet. The RFC 1738 describes the syntax and semantics for a URL
URI	Uniform Resource Identifier (URI) is a compact string of characters for identifying an abstract or physical feature. The RFC 3986 describes the generic syntax of a URI

2.1.2 Style conventions and general syntax of the URL

In URL syntax and in the descriptions of the arguments, text in italics with greater than and less than signs denote a content that must be substituted by a value or a string. When substituting the text, the greater than and less than signs must also be substituted. For example, the name of a camera is denoted by *<cameraname>* in description of URL syntax. In the example of URL syntax, *<cameraname>* is substituted by the string mycamera.

The URL syntax is written with the word "**Syntax:**" in bold face, accompanied by a box with the syntax of the command.

Syntax:

```
http://<server_address>/Interface/<object>[ /<subobject>... ] /<command>
[ ?<argument=value>[ &<argument=value>... ] ] [ &<general\_argument>... ]
```

Example 1: Requests the version of the API with response in XML

```
http://192.168.0.1:8601/Interface/GetAPIVersion?ResponseFormat=XML
```

Part



III

3 Interface

3.1 General arguments

All of the commands of the programming interface will accept some general arguments, which are:

Arguments of basic authentication by parameters:

Argument	Valid values	Description
AuthUser	String	User for authentication
AuthPass	String	User password for authentication

Arguments of safe authentication:

Argument	Valid values	Description
AuthSession	Integer	ID of the authentication session returned by the command to create the authentication session
AuthData	String	Calculated authentication data based on the parameters returned by the command to create the authentication session

Arguments for the format of the response:

Argument	Valid values	Description								
ResponseFormat	Text XML	<p>Format of the parameters of the response</p> <table border="1" style="margin-left: 20px;"> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>Text</td> <td>Response with parameters in text</td> </tr> <tr> <td>XML</td> <td>Response with parameters in XML</td> </tr> <tr> <td>JSON</td> <td>Response with parameters in JSON</td> </tr> </tbody> </table>	Value	Description	Text	Response with parameters in text	XML	Response with parameters in XML	JSON	Response with parameters in JSON
Value	Description									
Text	Response with parameters in text									
XML	Response with parameters in XML									
JSON	Response with parameters in JSON									

3.2 Types of variables

This programming interface implements some formats of proper variables. The following are the variables and proper formats:

Variable	Type of data	Description
APIDate	Date values	<p>This format of variable is applied where data values are passed directly in the URL.</p> <p>The format of date must be YYYY.MM.DD</p> <p>Ex: 2009, November, 15 2009.11.15</p>
APITime	Time values	<p>This format of variable is applied where time values are passed directly in the URL.</p> <p>The format of time must be HH.MM.SS</p> <p>Ex: 15:30:45 15.30.45</p>
APITimestamp	Time stamp values	This variable format is applied to return date and time values.

		The format of this variable is YYYY-MM-DD HH:MM:SS Ex: 2009, November, 15, 15:30:45 2009-11-15 15:30:45
APIMask	Text mask	Learn more in Filter Masks
API_masks	Text masks	Set of masks to filter results from API. Each mask must be delimited by comma character. Ex: ABC*,123*,??6G[1-0]
APIDouble	Double Value	The variable APIDouble is used to represent fractional numbers and its format must follow the US standard format, ie, the decimal separator character is the period (".") and the thousands separator character is the comma (","). The thousadans separator character (",") is not required. Ex: 1,223,302.45, 1223302.45
DUID	Object Identifier	Unique object identifier. DUID has the same format of a GUID without the surrounding braces. Ex: A6981C1B-DF35-427B-A4E6-9ED8AA31B626 Ex: 00000000-0000-0000-0000-000000000000 If a DUID is empty, it will be all zeroes
APILatLng	Latitude or Longitude value	The variable APILatLng is used to represent latitude and longitude values The decimal separator is DOT ("."). The value can contain up to 6 decimals. Ex: 23.167292, -10.392023
APIColor	Red Yellow Blue Navy Aqua Green Lime Fuchsia Purple Maroon Olive Teal	The variable APIColor is used to represent a color throughout the API.

3.3 Filter masks

Some API commands will allow the filtering of results using a filter mask.

The filter mask will compare the results with the specified mask, keeping only the valid results. The mask consists of literal characters, sets and wildcards.

Each literal character must match a single character in the string. The comparison to literal characters is case-insensitive.

Each set begins with an opening bracket ([) and ends with a closing bracket (]). Between the brackets are the elements of the set. Each element is a literal character or a range. Ranges are specified by an initial value, a dash (-), and a final value. Do not use spaces or commas to separate the elements of the set. A set must match a single character in the string. The character matches the set if it is the same as one of the literal characters in the set, or if it is in one of the ranges in the set. A character is in a range if it matches the initial value, the final value, or falls between the two values. All comparisons are case-insensitive. If the first character after the opening bracket of a set is an exclamation point (!), then the set matches any character that is not in the set.

Wildcards are asterisks (*) or question marks (?). An asterisk matches any number of characters. A question mark matches a single arbitrary character.

Examples:

Mask: ABC*

Result: Filter records starting with ABC.

Examples: ABCD, ABC123, ABCXYZ

Mask: ABC*123

Result: Filter records starting with ABC and ending with 123

Examples: ABCD123, ABC123, ABCXYZ123

Mask: ABC?123

Result: Filter records starting with ABC, containing any single character and ending with 123

Examples: ABCD123, ABCX123, ABCY123

Mask: ABC??23

Result: Filter records starting with ABC, containing any two characters and ending with 23

Examples: ABCD123, ABCXR23, ABCY923

Mask: ABC[XYZ]123

Result: Filter records starting with ABC, containing a single character in the set of (X, Y or Z) and ending with 123

Examples: ABCX123, ABCY123, ABCZ123

Mask: ABC[!XYZ]123

Result: Filter records starting with ABC, containing a single character not matching the set of (X, Y or Z) and ending with 123

Examples: ABCD123, ABCE123, ABCF123

Mask: ABC[D-G1-3]

Result: Filter records starting with ABC and containing a single character in the set of (D to G) or (1 to 3)

Examples: ABCD, ABC3, ABCF

Mask: ABC[D-G1-3]??[!ABC1-3]XYZ*

Result: Filter records starting with ABC, containing a single character in the set of (D to G) or (1 to 3), containing any two characters, containing the literals XYZ and ending with any chain of characters

Examples: ABCD12UXYZ, ABC2Y1UXYZ12345

3.4 System Objects

Some commands throughout the API will return IDs for object types in the system. Below is a list of all object types and its corresponding ID.

The ID can be either Integer or Typed string

ID	Description
1	Camera
2	Audio output device group
3	I/O Device
4	Alert Contact
5	Alert Group
6	User
7	User Group
8	Screen Style
9	Map
10	Global Event
11	Scheduled Event
12	Analytics Configuration
13	LPR Configuration
14	LPR List
15	User Screen View
16	Public Screen View
17	LPR Event
18	LPR Plate
19	Web Page
20	Failover monitor
21	LPR Plate Category Group
22	Camera Group
23	Operational Map
24	Digifort Server
25	LPR Zone
26	LPR Zone Group
27	Operational Map Custom Object

ID	Description
CAMERA	Camera
AUDIO OUTPUT DEVICE GROUP	Audio output device group
IO DEVICE	I/O Device
ALERT CONTACT	Alert Contact
ALERT GROUP	Alert Group
USER	User
USER GROUP	User Group
SCREEN STYLE	Screen Style

MAP	Map
GLOBAL EVENT	Global Event
SCHEDULED EVENT	Scheduled Event
ANALYTICS CONFIGURATION	Analytics Configuration
LPR CONFIGURATION	LPR Configuration
LPR LIST	LPR List
USER SCREEN VIEW	User Screen View
PUBLIC SCREEN VIEW	Public Screen View
LPR EVENT	LPR Event
LPR PLATE	LPR Plate
WEBPAGE	Web Page
FAILOVER MONITOR	Failover monitor
LPR PLATE CATEGORY GROUP	LPR Plate Category Group
CAMERA GROUP	Camera Group
OPERATIONAL MAP	Operational Map
SERVER	Digifort Server
LPR ZONE	LPR Zone
LPR ZONE GROUP	LPR Zone Group
OPERATIONAL MAP CUSTOM OBJECT	Operational Map Custom Object

3.5 Event Types

Some commands throughout the API will return IDs for event types in the system. Below is a list of all object types and its corresponding ID.

Event type	Description
ALARM_INPUT	Alarm input event. Triggered by: CAMERA, ALARM DEVICE
MANUAL	Manual event. Triggered by: CAMERA
TIMER	Timer event. Triggered by any object
COMMUNICATION_FAILURE	Device communication failure event. Triggered by: CAMERA, ALARM_DEVICE, ANALYTICS_CONFIGURATION, LPR_CONFIGURATION
COMMUNICATION_RESTORED	Device communication restored event. Triggered by: CAMERA, ALARM_DEVICE, ANALYTICS_CONFIGURATION, LPR_CONFIGURATION
RECORDING_FAILURE	Camera recording failure event. Triggered by: CAMERA
RECORDING_RESTORED	Camera recording restored event. Triggered by CAMERA
SERVER_HIGH_CPU	Server CPU usage is over the configured limit. Triggered by SERVER
SERVER_CPU_RESTORED	Server CPU usage returned to below configured limit. Triggered by SERVER
SERVER_HIGH_RAM	Server RAM usage is over the configured limit. Triggered by SERVER
SERVER_RAM_RESTORED	Server RAM usage returned to below configured limit. Triggered by SERVER
MOTION	Camera motion detection event. Triggered by: CAMERA
SCHEDULED	Scheduled event. Triggered by:

	SCHEDULED EVENT
GLOBAL	Global event. Triggered by: GLOBAL_EVENT
AUDIO_LEVEL_LOW	Audio level is low. Triggered by CAMERA
AUDIO_LEVEL_HIGH	Audio level is high. Triggered by CAMERA
FAILOVER	Failover event indicating that a server being monitored is not working. Triggered by FAILOVER SERVER MONITOR
FAILBACK	Failover event indicating that a server being monitored is working again. Triggered by FAILOVER SERVER MONITOR
KEEP_ALIVE	HTTP Connection Keep-Alive message. Sent every 5 seconds to keep the TCP / HTTP connection
DEVICE_DISK_FAILURE	Disk of device has failed. Triggered by CAMERA
DEVICE_VIDEO_LOSS	Analog video loss detected on device channel. Triggered by CAMERA
DEVICE_INTERCOM_BUTTON	Device intercom button pressed. Triggered by CAMERA
LIVE_WITNESS_SESSION_STARTED	Live Witness Session has started. Triggered by CAMERA
LIVE_WITNESS_SESSION_ENDED	Live Witness Session has ended. Triggered by CAMERA
MOBILE_CAMERA_SESSION_STARTED	Mobile Camera Session has started. Triggered by CAMERA
MOBILE_CAMERA_SESSION_ENDED	Mobile Camera Session has ended. Triggered by CAMERA
ANALYTICS_PRESENCE	Analytics presence rule event. Triggered by: ANALYTICS_CONFIGURATION
ANALYTICS_ENTER	Analytics enter rule event. Triggered by: ANALYTICS_CONFIGURATION
ANALYTICS_EXIT	Analytics exit rule event. Triggered by: ANALYTICS_CONFIGURATION
ANALYTICS_APPEAR	Analytics appear rule event. Triggered by: ANALYTICS_CONFIGURATION
ANALYTICS_DISAPPEAR	Analytics disappear rule event. Triggered by: ANALYTICS_CONFIGURATION
ANALYTICS_STOP	Analytics stop rule event. Triggered by: ANALYTICS_CONFIGURATION
ANALYTICS_LOITER	Analytics loitering rule event. Triggered by: ANALYTICS_CONFIGURATION
ANALYTICS_DIRECTION	Analytics direction filter rule event. Triggered by: ANALYTICS_CONFIGURATION
ANALYTICS_SPEED	Analytics speed filter rule event. Triggered by: ANALYTICS_CONFIGURATION
ANALYTICS_TAILGATING	Analytics tailgating filter rule event. Triggered by: ANALYTICS_CONFIGURATION
ANALYTICS_COUNTING_LINE_A	Analytics counting line rule event for direction A: Triggered by: ANALYTICS_CONFIGURATION
ANALYTICS_COUNTING_LINE_B	Analytics counting line rule event for direction B: Triggered by: ANALYTICS_CONFIGURATION
ANALYTICS_TAMPER	Analytics camera tampering rule event. Triggered by: ANALYTICS_CONFIGURATION
ANALYTICS_ABANDONED_OBJECT	Analytics abandoned object rule event. Triggered by: ANALYTICS_CONFIGURATION

ANALYTICS_REMOVED_OBJECT	Analytics removed object rule event. Triggered by: ANALYTICS_CONFIGURATION
ANALYTICS_SMOKE	Analytics smoke detection rule event. Triggered by: ANALYTICS_CONFIGURATION
ANALYTICS_FIRE	Analytics fire detection rule event. Triggered by: ANALYTICS_CONFIGURATION
ANALYTICS_FACE_DETECTION	Analytics face detection rule event. Triggered by: ANALYTICS_CONFIGURATION
ANALYTICS_OBJECT_CONDITION_CHANGE	Analytics object condition change rule event. Triggered by ANALYTICS_CONFIGURATION
ANALYTICS_FOLLOWING_ROUTE	Analytics object following route rule event. Triggered by ANALYTICS_CONFIGURATION
ANALYTICS_SIMILARITY	Analytics similarity rule event. Triggered by ANALYTICS_CONFIGURATION
ANALYTICS_OCCUPANCY	Analytics occupancy rule event. Triggered by ANALYTICS_CONFIGURATION
ANALYTICS_VIDEOSynopsis	Analytics Video Synopsis rule event. Triggered by ANALYTICS_CONFIGURATION
ANALYTICS_DETECTION	Analytics detection rule event. Triggered by ANALYTICS_CONFIGURATION
ANALYTICS_NON_DETECTION	Analytics non-detection rule event. Triggered by ANALYTICS_CONFIGURATION
ANALYTICS_SOUND_CLASSIFICATION	Analytics sound classification rule event. Triggered by ANALYTICS_CONFIGURATION
ANALYTICS_QUEUE	Analytics queue rule event. Triggered by ANALYTICS_CONFIGURATION
ANALYTICS_WRONG_DIRECTION	Analytics wrong direction rule event. Triggered by ANALYTICS_CONFIGURATION
ANALYTICS_ILLEGAL_PARKING	Analytics illegal parking rule event. Triggered by ANALYTICS_CONFIGURATION
ANALYTICS_ILLEGAL_CONVERSION	Analytics illegal conversion rule event. Triggered by ANALYTICS_CONFIGURATION
ANALYTICS_VEHICLE_DETECTION	Analytics vehicle detection rule event. Triggered by ANALYTICS_CONFIGURATION
ANALYTICS_VEHICLE_OVER_CROSS_LANE	Analytics vehicle over cross lane rule event. Triggered by ANALYTICS_CONFIGURATION
ANALYTICS_LANE_CHANGE	Analytics lane change rule event. Triggered by ANALYTICS_CONFIGURATION
ANALYTICS_LOGICAL_RULE	Analytics logical rule event. Triggered by ANALYTICS_CONFIGURATION
ANALYTICS_THERMAL	Analytics thermal event. Triggered by ANALYTICS_CONFIGURATION
ANALYTICS_FACE_RECOGNITION	Analytics face recognition event. Triggered by ANALYTICS_CONFIGURATION
ANALYTICS_SCENE_CHANGE	Analytics scene change event. Triggered by ANALYTICS_CONFIGURATION
ANALYTICS_PEDESTRIAN_DETECTION	Anaytics pedestrian detection event. Triggered by ANALYTICS_CONFIGURATION
ANALYTICS_ROAD_BLOCKED	Analytics road blocked event. Triggered by ANALYTICS_CONFIGURATION
ANALYTICS_COUNTER	Analytics counter event. Triggered by ANALYTICS_CONFIGURATION
ANALYTICS_COUNTING_LINE	Analytics counting line event. Triggered by ANALYTICS_CONFIGURATION

	ANALYTICS_CONFIGURATION
ANALYTICS_OBJECT_FILTER	Analytics object filter event. Triggered by ANALYTICS_CONFIGURATION
ANALYTICS_COLOR_FILTER	Analytics color filter event. Triggered by ANALYTICS_CONFIGURATION
ANALYTICS_CONTINUOUSLY	Analytics continuous event. Triggered by ANALYTICS_CONFIGURATION
ANALYTICS_PREVIOUS	Analytics previous event. Triggered by ANALYTICS_CONFIGURATION
ANALYTICS_DEEP_LEARNING	Analytics deep learning event. Triggered by ANALYTICS_CONFIGURATION
ANALYTICS_SMART_TRACKING	Analytics smart tracking event. Triggered by ANALYTICS_CONFIGURATION
ANALYTICS_SOCIAL_DISTANCING	Analytics social distancing event. Triggered by ANALYTICS_CONFIGURATION
ANALYTICS_ILLEGAL_VEHICLE_DETECTION	Analytics illegal vehicle detection event. Triggered by ANALYTICS_CONFIGURATION
ANALYTICS_TRAFFIC_ACCIDENT	Analytics traffic accident detection event. Triggered by ANALYTICS_CONFIGURATION
ANALYTICS_IRREGULAR_CONSTRUCTION	Analytics irregular construction event. Triggered by ANALYTICS_CONFIGURATION
ANALYTICS_IRREGULAR_PARKING	Analytics irregular parking event. Triggered by ANALYTICS_CONFIGURATION
ANALYTICS_PARKING_STATE	Analytics parking state event. Triggered by ANALYTICS_CONFIGURATION
ANALYTICS_TRAFFIC_CONGESTION	Analytics traffic congestion event. Triggered by ANALYTICS_CONFIGURATION
LPR	LPR event. Triggered by: LPR EVENT

3.6 Event Actions

Some commands throughout the API might return a list of actions from an event, these actions are called "Event Actions". Whenever this list must be returned it will have the following format:

Parameters of return for SendEmail:

Parameter	Type	Description
ACTIVE	Boolean	Action is active
EMAILGROUP	String	Contact group to send the e-mail
MESSAGETEXT	String	E-mail message
SMSFORMAT	Default Personalized	Type of SMS message

AttachCameralImage	Type	Description
ACTIVE	Boolean	Attach camera image is active
COUNT	Integer	Amount of objects in the list

Parameter in object list	Type	Description
OBJECTTYPE	Integer	System object ID
OBJECTTYPEID	String	System object ID in string format
OBJECTDUID	DUID	DUID of the object
OBJECTNAME	String	Name of the object

AddPlaybackLink	Type	Description
ACTIVE	Boolean	Send a link file attached to the e-mail for easy playback

CUSTOMSERVER	Type	Description
USECUSTOMSERVER	Boolean	Use a custom server address
ADDRESS	String	Server address
PORT	Integer	Server port
AUTOLOGIN	Boolean	Auto login to the server
USERNAME	String	Username for authentication

Parameters of return for ShowObjects:

Parameter	Type	Description
ACTIVE	Boolean	Action is active
COUNT	Integer	Amount of objects in the list

Parameter in object list	Type	Description
OBJECTTYPE	Integer	System object ID
OBJECTTYPEID	String	System object ID in string format
OBJECTDUID	DUID	DUID of the object
OBJECTNAME	String	Name of the object

Parameters of return for ShowPlaybackLoop:

Parameter	Type	Description
ACTIVE	Boolean	Action is active
COUNT	Integer	Amount of objects in the list
Parameter in object list		
OBJECTTYPE	Integer	System object ID
OBJECTTYPEID	String	System object ID in string format
OBJECTDUID	DUID	DUID of the object
OBJECTNAME	String	Name of the object

Parameters of return for ShowSnapshots:

Parameter	Type	Description
ACTIVE	Boolean	Action is active
SECONDSBEFORE	Integer	Number of seconds before the event to start playback
SECONDSAFTER	Integer	Number of seconds after the event to end playback
COUNT	Integer	Amount of objects in the list
Parameter in object list		
OBJECTTYPE	Integer	System object ID
OBJECTTYPEID	String	System object ID in string format
OBJECTDUID	DUID	DUID of the object
OBJECTNAME	String	Name of the object

Parameters of return for PlaySound:

Parameter	Type	Description
ACTIVE	Boolean	Action is active
FILENAME	String	Name of the audio file
PLAYTIMESECONDS	Integer	Amount of seconds to play the file

Parameters of return for SendMessage:

Parameter	Type	Description
ACTIVE	Boolean	Action is active
MESSAGETEXT	String	Message to send to the operator

Parameters of return for RequestAck:

Parameter	Type	Description
ACTIVE	Boolean	Action is active
MANDATORY	Boolean	Request acknowledge is mandatory

Parameters of return for TriggerGlobalEvents:

Parameter	Type	Description
ACTIVE	Boolean	Action is active
COUNT	Integer	Amount of objects in the list
Parameter in object list		
DUID	DUID	DUID of the Global Event
NAME	String	Name of the Global Event

Parameters of return for SendPushNotification:

Parameter	Type	Description
ACTIVE	Boolean	Action is active
GROUP	String	Group to send the notification
CUSTOMMESSAGE	String	Customized message to send in the notification

Parameters of return for SendAudioClip:

Parameter	Type	Description
ACTIVE	Boolean	Action is active
FILENAME	String	Name of the file to play
LOOP	Boolean	Loop the playback
LOOPCOUNT	Integer	Number of loops

Devices	Type	Description
COUNT	Integer	Number of devices in the list
<hr/>		
Parameter in object list	Type	Description
OBJECTTYPE	Integer	System object ID
OBJECTTYPEID	String	System object ID in string format
OBJECTDUID	DUID	DUID of the object
OBJECTNAME	String	Name of the object
<hr/>		

DeactivateObjects	Type	Description
COUNT	Integer	Number of objects in the list
<hr/>		
Parameter in object list	Type	Description
OBJECTTYPE	Integer	System object ID
OBJECTDUID	DUID	DUID of the object
OBJECTNAME	String	Name of the object
<hr/>		

Parameters of return for CallPreset:

Parameter	Type	Description
ACTIVE	Boolean	Action is active
COUNT	Integer	Amount of presets in the list
<hr/>		
Parameter in object list	Type	Description
CAMERA	String	Camera name
PRESETNUMBER	Integer	Preset ID

Parameters of return for ActivateIOOutput:

Parameter	Type	Description
ACTIVE	Boolean	Action is active
COUNT	Integer	Amount of actions in the list

Parameter in object list	Type	Description
DEVICENAME	String	Name of the parent device
DEVICETYPE	String	System object ID
OUTPUTSCRIPTNAME	String	Name of the output script

Parameters of return for SendToVirtualMatrix:

Parameter	Type	Description
ACTIVE	Boolean	Action is active
COUNT	Integer	Amount of actions in the list

Parameter in object list	Type	Description
NAME	String	Name of the action
DESCRIPTION	String	Description of the action

Monitors	Type	Description
COUNT	Integer	Number of monitors where object will be sent

Parameter in object list	Type	Description
NAME	String	Name of the monitor

ACTIONTYPE	SINGLE_OBJECT VIEW	Identifies the type of action
		Name
		SINGLE_OBJECT
		Only a single object will be sent to virtual matrix
		Description
		VIEW
		A saved view will be sent to virtual matrix

Parameter when ActionType is SINGLE_OBJECT	Type	Description
SYSTEMOBJECTDUID	DUID	DUID of the object that will be sent to the Virtual Matrix
SYSTEMOBJECTNAME	String	Name of the object that will be sent to the Virtual Matrix
SYSTEMOBJECTTYPE	Object Type String	Type of the object that will be sent to the Virtual Matrix
SCREENSTYLEID	Integer	ID of the layout to be set on virtual matrix
SELECTEDSPOT	Integer	Spot ID in the layout

Parameter when ActionType is VIEW	Type	Description
VIEWNAME	String	Name of the view that will be sent to the Virtual Matrix
SCREENSTYLEID	Integer	ID of the layout to be set on virtual matrix

Parameters of return for ObjectActivation:

Parameter	Type	Description
ACTIVE	Boolean	Action is active
ActivateObjects		
COUNT	Integer	Number of objects in the list
Parameter in object list	Type	Description
OBJECTTYPE	Integer	System object ID
OBJECTTYPEID	String	System object ID in string format
OBJECTDUID	DUID	DUID of the object
OBJECTNAME	String	Name of the object

Parameter	Type	Description
ACTIVE	Boolean	Action is active
DeactivateObjects		
COUNT	Integer	Number of objects in the list
Parameter in object list	Type	Description
OBJECTTYPE	Integer	System object ID
OBJECTTYPEID	String	System object ID in string format
OBJECTDUID	DUID	DUID of the object
OBJECTNAME	String	Name of the object

Parameters of return for HTTPRequests:

Parameter	Type	Description
ACTIVE	Boolean	Action is active
COUNT	Integer	Amount of actions in the list
Parameter in object list		
NAME	String	Name of the action
DESCRIPTION	String	Description of the action
METHOD	GET POST PUT PATCH DELETE SUBSCRIBE UNSUBSCRIBE RENEW	Identifies which type of HTTP Request method is configured
URL	String	URL to call
USERNAME	String	Username for authentication
DATA	String	HTTP data (If present)
EXTRAHEADERS	String	Comma-separated list of extra HTTP headers

Parameter	Type	Description
METHOD	GET POST	Identifies which type of HTTP Request method is configured
URL	String	URL to call
USERNAME	String	Username for authentication
DATA	String	HTTP data (If present)

Parameters of return for CreateBookmark:

Parameter	Type	Description
ACTIVE	Boolean	Action is active
CAMERAS	String	Comma-delimited list of cameras
STARTTIMESECONDSBEFORE	Integer	Amount of seconds before current time to start bookmark
TITLE	String	Bookmark title
COLOR	String	Bookmark color text

Parameters of return for DownloadMedia:

Parameter	Type	Description
ACTIVE	Boolean	Action is active
SECONDS	Integer	Amount of seconds to download
CAMERAS	String	Comma-delimited list of cameras to download media from

Parameters of return for CreateTimer:

Parameter	Type	Description
ACTIVE	Boolean	Action is active
COUNT	Integer	Amount of timers created
TIMER		
NAME	String	Timer name
DESCRIPTION	String	Description of the timer
TIMETOACTIVATE	Integer	Activate timer in X seconds
CANCELONEVENT	Boolean	Cancel timer in the occurrence of another event
Event List		
COUNT	Integer	Amount of events in the list
Parameter in object list		
EVENTTYPE	String	Event Type
OBJECTTYPE	String	System object ID in string format
OBJECTDUID	DUID	DUID of the object
OBJECTNAME	String	Name of the object
Actions		
This section will contain all actions of this timer event, including sub-timers		

Example of return in text:

```
ACTIONS_HTTPREQUEST_ACTIVE=TRUE
ACTIONS_HTTPREQUEST_METHOD=GET
ACTIONS_HTTPREQUEST_URL=127.0.0.1
ACTIONS_HTTPREQUEST_USERNAME=admin
ACTIONS_DOWNLOADMEDIA_ACTIVE=TRUE
ACTIONS_DOWNLOADMEDIA_SECONDS=3600
ACTIONS_DOWNLOADMEDIA_CAMERAS=01,02,03,04,05,
```

Example of return in XML:

```
<Actions>
  <HTTPRequest>
    <Active>TRUE</Active>
    <Method>GET</Method>
    <URL>127.0.0.1</URL>
    <Username>admin</Username>
  </HTTPRequest>
  <DownloadMedia>
    <Active>TRUE</Active>
    <Seconds>3600</Seconds>
    <Cameras>01,02,03,04,05,</Cameras>
  </DownloadMedia>
</Actions>
```

Example of return in JSON:

```
"Actions": {
  "HTTPRequest": {
    "Active": true,
    "Method": "GET",
    "URL": "127.0.0.1"
    "Username": "admin"
  },
  "DownloadMedia": {
    "Active": true,
    "Seconds": 3600,
    "Cameras": "01,02,03,04,05,"
  }
}
```

3.7 MIME Types

Media Frame Type:

MIME	Description
image/jpeg	JPEG image
image/wavelet	WAVELET image
video/mpeg	MPEG-4 video
video/h263	H.263 video
video/h264	H.264 video
video/h265	H.265 video
video/mjpeg	MJPEG video
audio/L24	L-PCM
audio/basic	G.711 audio
audio/G726-16	G.726 audio in 16kbps
audio/G726-24	G.726 audio in 24kbps

audio/G726-32	G.726 audio in 32kbps
audio/G726-40	G.726 audio in 40kbps
audio/AAC	AAC audio
application/octet-stream	Unrecognized format
Integer	Specify the size (in bytes) of the media frame
Integer	Frame number
APIDate	Frame date in APIDate format
APITime	Frame time in APITime format
jpeg	JPEG image
jpeg-2000	JPEG-2000 image
h.263/I-Frame	H.263 frame of type I
h.263/P-Frame	H.263 frame of type P
h.263/PB-Frame	H.263 frame of type PB
h.263/B-Frame	H.263 frame of type B
h.263/EI-Frame	H.263 frame of type EI
h.263/EP-Frame	H.263 frame of type EP
mpeg-4/I-Frame	MPEG-4 frame of type I
mpeg-4/P-Frame	MPEG-4 frame of type P
mpeg-4/B-Frame	MPEG-4 frame of type B
h.264/I-Frame	H.264 frame of type I
h.264/P-Frame	H.264 frame of type P
h.264/B-Frame	H.264 frame of type B
h.265/I-Frame	H.265 frame of type I
h.265/P-Frame	H.265 frame of type P
h.265/B-Frame	H.265 frame of type B
mxpeg/jpeg	MxPEG frame of type JPEG
mxpeg	MxPEG frame
audio/L24	L-PCM audio chunk
audio/basic	G.711 audio chunk
audio/G726-16	G.726 audio chunk in 16kbps
audio/G726-24	G.726 audio chunk in 24kbps
audio/G726-32	G.726 audio chunk in 32kbps
audio/G726-40	G.726 audio chunk in 40kbps
audio/AAC	AAC audio chunk
application/octet-stream	Unrecognized format

3.7.1 Media Types

MIME	Description
image/jpeg	JPEG image
image/wavelet	WAVELET image
video/mpeg	MPEG-4 video
video/h263	H.263 video
video/h264	H.264 video
video/h265	H.265 video
video/mxpeg	MxPEG video
audio/L24	L-PCM
audio/basic	G.711 audio
audio/G726-16	G.726 audio in 16kbps
audio/G726-24	G.726 audio in 24kbps
audio/G726-32	G.726 audio in 32kbps
audio/G726-40	G.726 audio in 40kbps
audio/AAC	AAC audio

application/octet-stream	Unrecognized format
--------------------------	---------------------

3.7.2 Media Frame Types

MIME	Description
jpeg	JPEG image
jpeg-2000	JPEG-2000 image
h.263/I-Frame	H.263 frame of type I
h.263/P-Frame	H.263 frame of type P
h.263/PB-Frame	H.263 frame of type PB
h.263/B-Frame	H.263 frame of type B
h.263/EI-Frame	H.263 frame of type EI
h.263/EP-Frame	H.263 frame of type EP
mpeg-4/I-Frame	MPEG-4 frame of type I
mpeg-4/P-Frame	MPEG-4 frame of type P
mpeg-4/B-Frame	MPEG-4 frame of type B
h.264/I-Frame	H.264 frame of type I
h.264/P-Frame	H.264 frame of type P
h.264/B-Frame	H.264 frame of type B
h.265/I-Frame	H.265 frame of type I
h.265/P-Frame	H.265 frame of type P
h.265/B-Frame	H.265 frame of type B
mxpeg/jpeg	MxPEG frame of type JPEG
mxpeg	MxPEG frame
audio/L24	L-PCM audio chunk
audio/basic	G.711 audio chunk
audio/G726-16	G.726 audio chunk in 16kbps
audio/G726-24	G.726 audio chunk in 24kbps
audio/G726-32	G.726 audio chunk in 32kbps
audio/G726-40	G.726 audio chunk in 40kbps
audio/AAC	AAC audio chunk
application/octet-stream	Unrecognized format

3.8 Authentication

To access the programming interface commands user and password authentication is required.

This programming interface supports 3 different types of authentication:

Types of authentication:

Value	Description
Basic HTTP	Basic HTTP authentication
Basic with parameters	Basic authentication by parameters in the URL
Safe	Safe authentication by parameters in the URL

3.8.1 Basic HTTP authentication

The basic HTTP authentication method can be used for authentication in the interface.

This is an unsafe authentication method, which most libraries of HTTP communication must support. This is a method that sends the user and password in Base 64 Code in the headers of the HTTP message.

Warning: We do not recommend the use of this type of authentication due to the high risk of exposure of the access credentials. Instead of using this type of authentication, give preference to the use of [Safe Autentication](#). While safe authentication is the best and safest of the 3 supported authentication methods, it is also the most difficult to use, as it involves the use of hashing with MD5, which may not be supported by all of the libraries.

To learn more about the basic HTTP authentication, refer to the [RFC 2617](#).

3.8.2 Basic authentication with parameters

Basic authentication with parameters is an unsafe form of authentication that sends the access credentials (User and password) by way of parameters of the URL using pure text.

While this type of authentication is the easiest to use, it is also the most unsafe, as the access credentials are sent without cryptography or any coding.

Warning: We do not recommend the use of this type of authentication due to the high risk of exposure of the access credentials. Instead of using this type of authentication, give preference to the use of [Safe Authentication](#). While being the best and safest of the 3 supported methods, safe authentication is also the most difficult to use, as it involves the use of hashing with MD5, which may not be supported by all of the libraries. When the use of safe authentication is not possible, give preference to [Basic HTTP Authentication](#) instead of Basic Autentication with parameters

To use this type of authentication, add the `AuthUser` and `AuthPass` parameters in the URL of the command.

Parameters of authentication:

Parameter	Type	Description
<code>AuthUser</code>	String	User for authentication
<code>AuthPass</code>	String	Password for authentication

Example 1: Requests the version of the API with authentication with the user admin, without password
`http://192.168.0.1:8601/Interface/GetAPIVersion?AuthUser=admin`

Example 2: Requests the version of the API with auhtentication with the user admin, and the password pass

```
http://192.168.0.1:8601/Interface/GetAPIVersion?AuthUser=admin&
AuthPass=pass
```

3.8.3 Safe authentication

Safe authentication with parameters is the recommended method for authentication in the programming interface. This is the safest of the 3 available methods, however, it is also the most difficult to implementar, as it uses [Hashing MD5](#) techniques, which may not be available in all of the libraries.

The implementation of this method was conceived using as its base some concepts of HTTP Digest authentication ([RFC 2617](#)), though with simpler methods.

Due to the nature of HTTP protocol HTTP not maintaining a TCP connection for all requests, to authenticate using safe authentication, we must first [create an authentication session](#).

After creating the session, it's necessary to [calculate the autentication data](#).

After calculating the authentication data, simply add the `AuthSession` and `AuthData` parameters in the URL of the command to be executed.

Parameters of authentication:

Parameter	Type	Description
<code>AuthSession</code>	String	ID of the authentication session returned by the command for to create the authentication session
<code>AuthData</code>	String	Authentication data calculated based on the parameters returnad by the command to create the authentication session

Example 1: Requests the version of the API with safe authentication

```
http://192.168.0.1:8601/Interface/GetAPIVersion?AuthSession=1&
AuthData=AF63604073043A3C47FB5A506D8A8EFD
```

With the safe authentication session created, you must now [keep it open](#).

3.8.3.1 Creating an authentication session

Create an authentication session.

Compatibility: All editions

Method: HTTP GET

Syntax:

```
http://<server_address>/Interface/CreateAuthSession[?<general_argument>
[&<general_argument>...]]
```

Response:

An authentication session will be created, and the ID and the random value NOnce will be generated.

HTTP Return: 200 OK

Parameters of return:

Parameter	Type	Description
<code>ID</code>	Integer	ID of the authentication session
<code>NONCE</code>	String	Random value for hashing the access credentials

Example of return in text:

```
RESPONSE_CODE=0
RESPONSE_MESSAGE=OK
ID=2
NONCE=D11193880891BEB0931A52E3F3CA322F
```

Example of return in XML:

```
<Response>
  <Code>0</Code>
  <Message>OK</Message>
  <Data>
    <Session>
      <ID>3</ID>
      <NOnce>76569D03D9D479201EDB1FAB36DBEDEA</NOnce>
    </Session>
  </Data>
</Response>
```

Example of return in JSON:

```
{
  "Response": {
    "Code": 0,
    "Message": "OK",
    "Data": {
      "Session": {
        "ID": 32,
        "NOnce": "164EF22C931C647424949799404B9567"
      }
    }
  }
}
```

3.8.3.2 Calculating the authentication data

To calculate the authentication data, we use the following formula:

```
AuthData =UpperCase(MD5Hash(NOnce + ":" + UpperCase(Username) + ":" + UpperCase(MD5Ha
```

Where `MD5Hash` is the call for a function which accepts the string as input parameter and returns a Hash MD5 with values in hexadecimal.

Where `UpperCase` is the call for a function which accepts a string as input parameter and returns this string with the characters in upper case.

Where `Username` is the user of authentication.

Where `Password` is the password of the user of authentication.

Therefore, `AuthData` is the result of the Hash MD5 of the concatenation of the value `NOnce` sent by the server, with the colon character `:`, with the user of authentication in upper case characters, with the colon character `:`, with the Hash MD5 of the password of authentication.

Example of calculation of the authentication for the user "admin" with the password "pass":**NOnce value received:**

```
NONCE=68F1EE37050F456851DC90D62791839E
```

Calculation of AuthData:

```
AuthData =UpperCase(MD5Hash(68F1EE37050F456851DC90D62791839E + ":" +
"ADMIN" + ":" +UpperCase(MD5Hash(pass))));
```

Result of AuthData:

```
AuthData=AF63604073043A3C47FB5A506D8A8EFD
```

3.8.3.3 Maintaining an authentication session

Due to the nature of HTTP protocol not maintaining TCP connections (consequently making it difficult to maintain authentication sessions), after creating a safe authentication session, you must keep it open during the entire period of use of the API.

By default, an authentication session will expire in 60 seconds, if there is no activity in the API using its authentication ID. If its authentication session expires, it will be necessary to create a new session.

To keep an authentication session open, simply maintain activity in the API by call of some command, thus updating the hour of the last call of the API and maintaining the session for 60 more seconds.

However, a special command was created for maintaining an authentication session, and the use of this command is recommended in case the activity in the interface is not constant.

Compatibility: All editions

Method: HTTP GET

Syntax:

```
http://<server_address>/Interface/UpdateAuthSession[?<argument=value>
[&<argument=value>...][&<general_argument>...]]
```

Arguments:

Argument	Valid values	Description
AuthSession	Integer >= 1	ID of the authentication session to be kept open
AuthData	String	Data of the authentication of the session

Response:

Default response of the API.

HTTP Return: 200 OK

Parameters of return: [Default return of the API](#)

3.9 Responses

The responses to the commands of API can be formatted in two types: Text and XML.

3.9.1 Response in text

The responses in text format will be in the format of a list of Parameters and Values:

```
<parameter_1>=<value>
<parameter_2>=<value>
..
<parameter_n>=<value>
```

Example:

```
RESPONSE_CODE=0
RESPONSE_MESSAGE=OK
COUNT=10
```

3.9.1.1 List of parameters in text

When the response of a command returns a list of parameters (Ex: List of cameras, list of users, etc...), these parameters will obey the following syntax:

```
<object>_<num>_<field>=<value>
```

Where:

Value	Description
Object	Name of the type of object
Num	Number of the record
Field	Name of the field
Value	Value of the field

Example 1: List of cameras

```
CAMERA_1_NAME=Entrance
CAMERA_1_DESCRIPTION=Front Camera
CAMERA_1_MODEL=Generic
CAMERA_1_DEVICETYPE=2
CAMERA_2_NAME=Backdoor
CAMERA_2_DESCRIPTION=Door Camera
CAMERA_2_MODEL=Generic
CAMERA_2_DEVICETYPE=2
```

Example 2: List of users

```
USER_1_NAME=admin
USER_1_DESCRIPTION=System Administration Account
USER_2_NAME=Charlie
USER_2_DESCRIPTION=Charlie Brown
```

3.9.2 Response in XML

The responses in XML will be inserted into the tag root <Response></Response>.

The [default return parameters](#) will be inside the tag root, whereby the parameters of response of the called command will be returned inside the tag <Data></Data>.

Example:

```
<Response>
  <Code>0</Code>
  <Message>OK</Message>
  <Data>
    <ApiVersion>
      <Name>Digifort HTTP API Server</Name>
      <Version>1.1.0</Version>
      <Major>1</Major>
      <Minor>1</Minor>
      <BugFix>0</BugFix>
```

```

    </ApiVersion>
  </Data>
</Response>
```

3.9.2.1 List of parameters in XML

When the response of a command returns a list of parameters (Ex: List of cameras, list of users, etc...), these parameters will obey the following syntax:

```

<Objects>
  <Count>COUNT</Count>
  <Object>
    <Field>FIELD_VALUE</Field>
  </Object>
  ..
  <Object>
    <Field>FIELD_VALUE</Field>
  </Object>
</Objects>
```

Where:

Value	Description
Object	Name of the type of object
Count	Total number of registers
Field	Name of the field
FIELD_VALUE	Value of the field

Example 1: List of cameras

```

<Response>
  <Code>0</Code>
  <Message>OK</Message>
  <Data>
    <Cameras>
      <Count>2</Count>
      <Camera>
        <Name>Camera1</Name>
        <Description>My Camera</Description>
        <Model>Generic</Model>
        <DeviceType>1</DeviceType>
      </Camera>
      <Camera>
        <Name>Camera2</Name>
        <Description>My Video Server</Description>
        <Model>Generic</Model>
        <DeviceType>2</DeviceType>
      </Camera>
    </Cameras>
  </Data>
</Response>
```

Example 2: List of users

```

<Response>
  <Code>0</Code>
```

```

<Message>OK</Message>
<Data>
  <Users>
    <Count>2</Count>
    <User>
      <Name>admin</Name>
      <Description>System administration account</Description>
    </User>
    <User>
      <Name>Guest</Name>
      <Description>Guest user</Description>
    </User>
  </Users>
</Data>
</Response>

```

3.9.3 Response in JSON

The responses in JSON will be inserted into the root node `Response`.

The [default return parameters](#) will be inside the root node, whereby the parameters of response of the called command will be returned inside the node `Data`.

Example:

```
{
  "Response": {
    "Code": 0,
    "Message": "OK",
    "Data": {
      "ApiVersion": {
        "Name": "Digifort HTTP API Server",
        "Version": "1.9.0",
        "Edition": "ENTERPRISE",
        "Major": 1,
        "Minor": 9,
        "BugFix": 0
      }
    }
  }
}
```

3.9.4 Default parameters of return

All commands of the API will return a default response and the response data of the called command.

Default parameters of response:

Parameter	Type	Description
CODE	Integer	Code of the response
MESSAGE	String	Message of the response

List of parameters of response in text:

The parameters of response in text will obey the following syntax:

```
RESPONSE_<field>=<value>
```

Example of return in text:

```
RESPONSE_CODE=0
RESPONSE_MESSAGE=OK
```

Example of return in XML:

```
<Response>
  <Code>0</Code>
  <Message>OK</Message>
</Response>
```

Example of return in JSON:

```
{
  "Response": {
    "Code": 0,
    "Message": "OK"
  }
}
```

3.9.5 Return codes

The tables below summarize all of the error return codes and messages of API HTTP

General errors (Applicable to any command):

Value	Message
0	OK
1	Missing parameter
2	RESERVED (NOT CURRENTLY IN USE)
3	Object not found
4	Object deactivated
5	You don't have enough rights to access the desired object
6	Data not ready yet - DEPRECATED
7	Invalid parameter value
8	To access this command you need to use the Admin user account
9	Authentication required
10	You don't have enough rights to use the specified command
11	The provided data is invalid

Safe Authentication Errors (Applicable only to commands for safe authentication):

Value	Message
100	Invalid or expired authentication session
101	Authentication error

Authentication Errors:

Value	Message
200	Internal error on authentication
201	User was not found
202	The user cannot login at this time (Login time restriction)
203	The user cannot login from this IP address (IP address restriction)
204	The user account is blocked
205	The user account is expired
206	The authentication is invalid

Camera Errors (Applicable only to camera commands):

Value	Message
10000	Camera out of order
10001	Camera is under privacy mode
10100	You don't have enough rights to control PTZ cameras - DEPRECATED
10101	You don't have enough rights to control the specified camera
10102	The camera is locked by another user
10103	The PTZ controls for the specified camera are disabled
10104	Invalid PTZ operation
10105	Command not supported by the PTZ driver
10106	Stopping PTZ Patrol is disabled
10107	PTZ Operation Out Of Schedule
10200	Image not found for the specified date and time
10201	Decoding error
10202	Database error
10300	Camera manual event was not found
10400	No rights to listen to camera audio
10401	No rights to send audio to the camera
10500	Edge recording for the specified camera is not activated
10501	Camera does not support edge recording
10600	Recording lock directory is not configured
10601	Recording lock directory is not accessible
10602	Recording lock range is too big (Maximum 24 hours range)
10603	Recording lock expiration date must be higher than start date

I/O Errors (Applicable only to I/O commands):

Value	Message
20000	Alarm output action not found
20001	Virtual Port number is invalid or the operation was not successful

Virtual Matrix Errors (Applicable only to virtual matrix commands):

Value	Message
30000	Monitor not found

LPR Errors (Applicable only to LPR commands):

Value	Message
40000	Record not found
40100	LPR Configuration out of order

Analytics Errors (Applicable only to Analytics commands):

Value	Message
50000	Analytics metadata was not found

3.9.6 Extra HTTP Headers

The system allows for adding extra HTTP headers in the responses for specific application integration.

For example, many applications require the `Access-Control-Allow-Origin` header to allow for cross realm communication.

Extra headers can be added to every API response by creating a file called `HTTPApiCustomHeaders` (no extension) in the same folder as `Server.exe`, this should be a plain text file. Each line of the file must contain one HTTP header followed by colon and value.

Example of contents of a file:

```
Access-Control-Allow-Origin: *
CustomHttpHeader1: CustomValue1
CustomHttpHeader2: CustomValue2
```

Part



IV

4 API Groups

4.1 Version

4.1.1 Requesting the version of API

Requests the version of API HTTP

Compatibility: All editions

Security level: Requires user authentication

Method: HTTP GET

Syntax:

```
http://<server_address>/Interface/GetAPIVersion[?<general_argument>
[&<general_argument>...]]
```

Example 1: Requests the version of API with response in XML

```
http://192.168.0.1:8601/Interface/GetAPIVersion?ResponseFormat=XML
```

Example 2: Requests the version of API with response in Text and authentication of the user admin

```
http://192.168.0.1:8601/Interface/GetAPIVersion?ResponseFormat=Text&
AuthUser=admin
```

Response:

A list of parameter-value pairs is returned

HTTP Return: 200 OK

Parameters of return:

Parameter	Type	Description
NAME	String	Name of the API HTTP server
VERSION	String	Version of API HTTP (MAJOR.MINOR.BUGFIX)
EDITION	String	Edition of the server
MAJOR	Integer	Main Version of API
MINOR	Integer	Features Version of API
BUGFIX	Integer	Corrections Version of API

Example of return in text:

```
RESPONSE_CODE=0
RESPONSE_MESSAGE=OK
NAME=Digifort HTTP API Server
VERSION=1.9.0
EDITION=ENTERPRISE
MAJOR=1
MINOR=9
BUGFIX=0
```

Example of return in XML:

```
<Response>
  <Code>0</Code>
  <Message>OK</Message>
  <Data>
    <ApiVersion>
      <Name>Digifort HTTP API Server</Name>
      <Version>1.9.0</Version>
      <Edition>ENTERPRISE</Edition>
      <Major>1</Major>
      <Minor>9</Minor>
      <BugFix>0</BugFix>
    </ApiVersion>
  </Data>
</Response>
```

Example of return in JSON:

```
{
  "Response": {
    "Code": 0,
    "Message": "OK",
    "Data": {
      "ApiVersion": {
        "Name": "Digifort HTTP API Server",
        "Version": "1.9.0",
        "Edition": "ENTERPRISE",
        "Major": 1,
        "Minor": 9,
        "BugFix": 0
      }
    }
  }
}
```

4.2 Server

4.2.1 Requesting the machine code of the server

Requests the machine code of the server. The machine code of the server is the code that identifies the licenses of each computer, you can use it to identify the servers.

Compatibility: All editions

Security level: Requires user authentication

Method: HTTP GET

Syntax:

```
http://<server_address>/Interface/Server/GetMachineCode
[?<general_argument>[&<general_argument>...]]
```

Example 1: Requests the machine code of the server with response in XML

```
http://192.168.0.1:8601/Interface/Server/GetMachineCode?ResponseFormat=XML
```

Example 2: Requests the machine code of the server with response in Text and authentication of the user admin

```
http://192.168.0.1:8601/Interface/Server/GetMachineCode?
ResponseFormat=Text&AuthUser=admin
```

Response:

A list of parameter-value pairs is returned

HTTP Return HTTP: 200 OK

Parameters of return:

Parameter	Type	Description
MACHINECODE	String	Machine code of the server
SERVERID	String	ID of the server

Example of return in text:

```
RESPONSE_CODE=0
RESPONSE_MESSAGE=OK
MACHINECODE=AF20-DGF-9016277-C3681*1CFDC9/5AD0-MKEY-D921A7
SERVERID=8E18A60F508A3428346B23D504B74484
```

Example of return in XML:

```
<Response>
  <Code>0</Code>
  <Message>OK</Message>
  <Data>
    <MachineCode>
      <MachineCode>AF20-DGF-9016277-C3681*1CFDC9/5AD0-MKEY-D921A7</MachineCode>
      <ServerID>8E18A60F508A3428346B23D504B74484</ServerID>
    </MachineCode>
  </Data>
</Response>
```

Example of return in JSON:

```
{
  "Response": {
    "Code": 0,
    "Message": "OK",
    "Data": {
      "MachineCode": {
        "MachineCode": "AF20-DGF-9016277-C3681*1CFDC9/5AD0-MKEY-D921A7",
        "ServerID": "8E18A60F508A3428346B23D504B74484"
      }
    }
  }
}
```

4.2.2 Requesting data about the server

Requests data of the server.

Compatibility: All editions

Security level: Requires user authentication

Method: HTTP GET

Syntax:

```
http://<server_address>/Interface/Server/GetInfo  
[?<general_argument>[&<general_argument>...]]
```

Example 1: Requests the data of the server with response in XML

```
http://192.168.0.1:8601/Interface/Server/GetInfo?ResponseFormat=XML
```

Example 2: Requests the data of the server with response in text and authentication of the user admin

```
http://192.168.0.1:8601/Interface/Server/GetInfo?  
ResponseFormat=Text&AuthUser=admin
```

Response:

A list of parameter-value pairs is returned

HTTP Return HTTP: 200 OK

Parameters of return:

Parameter	Type	Description
EDITION	String	Edition of the server
VERSION	String	Version of the server
RELEASEDATE	APIDate	Release date of the version
RELEASETYPE	String	Type of release of the version
PLATFORM	String	Platform of the server
UPTIME	Integer	The number of seconds that the server is active
DATE	APIDate	Date of the server
TIME	APITime	Time of the server
DATETIME	APITimestamp	Date and Time of the server in Timestamp format
UTCDATETIME	APITimestamp	UTC Date and Time of the server in Timestamp format
SERVERTYPE	MASTER SLAVE	Type of Master / Slave server

Example of return in text:

```
RESPONSE_CODE=0
RESPONSE_MESSAGE=OK
EDITION=ENTERPRISE
VERSION=7.3.0.1
RELEASEDATE=2020.05.06
RELEASETYPE=Final
PLATFORM=Windows
UPTIME=754571
DATE=2020.05.15
TIME=16.05.40.829
DATETIME=2020-05-15 16:05:40.829
UTCDATETIME=2020-05-15 19:05:40.829
SERVERTYPE=MASTER
```

Example of return in XML:

```

<Response>
  <Code>0</Code>
  <Message>OK</Message>
  <Data>
    <Info>
      <Edition>ENTERPRISE</Edition>
      <Version>7.3.0.1</Version>
      <ReleaseDate>2020.05.06</ReleaseDate>
      <ReleaseType>Final</ReleaseType>
      <Platform>Windows</Platform>
      <UpTime>754527</UpTime>
      <Date>2020.05.15</Date>
      <Time>16.04.56.999</Time>
      <DateTime>2020-05-15 16:04:56.999</DateTime>
      <UTCDateTime>2020-05-15 19:04:56.999</UTCDateTime>
      <ServerType>MASTER</ServerType>
    </Info>
  </Data>
</Response>

```

Example of return in JSON:

```
{
  "Response": {
    "Code": 0,
    "Message": "OK",
    "Data": {
      "Info": {
        "Edition": "ENTERPRISE",
        "Version": "7.3.0.1",
        "ReleaseDate": "2020.05.06",
        "ReleaseType": "Final",
        "Platform": "Windows",
        "UpTime": 754495,
        "Date": "2020.05.15",
        "Time": "16.04.24.700",
        "DateTime": "2020-05-15T16:04:24.700Z",
        "UTCDateTime": "2020-05-15T19:04:24.700Z",
        "ServerType": "MASTER"
      }
    }
  }
}
```

4.2.3 Requesting licensing information

Requests data of the server.

Compatibility: All editions

Security level: Requires user authentication with rights to configure the server

Method: HTTP GET

Syntax:

```
http://<server_address>/Interface/Server/GetLicenses
[?<general_argument>[&<general_argument>...]]
```

Example 1: Requests the licenses of the server with response in XML

```
http://192.168.0.1:8601/Interface/Server/GetLicenses?ResponseFormat=XML
```

Example 2: Requests the licenses of the server with response in text and authentication of the user admin

```
http://192.168.0.1:8601/Interface/Server/GetLicenses?
ResponseFormat=Text&AuthUser=admin
```

Response:

A list with the summary of all installed licenses will be returned. If there are licenses with expiration, an expiration section will be added to the result

HTTP Return: 200 OK

Parameters of return:

Summary Fixed Parameter:

Parameter	Type	Description
COUNT	Integer	Number of summary records (Not included in JSON response)

Parameters in the list of licenses in summary:

Parameter	Type	Description
OBJECTTYPE	CAMERA IO_DEVICE EDGE_ANALYTICS EDGE_LPR MULTI_CHANNEL_DEVICE LPR_BRIDGE	Type of object being licensed
FAILOVER	Boolean	License is Failover or Regular
TOTALICENSES	Integer	Number of license files for this object type
TOTALOBJECTS	Integer	Number of objects supported by installed licenses
USEDOBJECTS	Integer	Number of objects currently licensed
INFO	String	Extended information such as expiration time (For 4-hour demo license)

Parameters in expiration summary:

Parameter	Type	Description
DATE	APITimestamp	Expiration date
DAYSLEFT	Integer	Number of days until expiration

Example of return in text:

```
RESPONSE_CODE=0
RESPONSE_MESSAGE=OK
SUMMARY_COUNT=4
SUMMARY_LICENSE_1_OBJECTTYPE=CAMERA
SUMMARY_LICENSE_1_FAILOVER=FALSE
```

```

SUMMARY_LICENSE_1_TOTALLICENSES=4
SUMMARY_LICENSE_1_TOTALOBJECTS=168
SUMMARY_LICENSE_1_USEDOBJECTS=70
SUMMARY_LICENSE_1_INFO=
SUMMARY_LICENSE_2_OBJECTTYPE=IO_DEVICE
SUMMARY_LICENSE_2_FAILOVER=FALSE
SUMMARY_LICENSE_2_TOTALLICENSES=2
SUMMARY_LICENSE_2_TOTALOBJECTS=65
SUMMARY_LICENSE_2_USEDOBJECTS=10
SUMMARY_LICENSE_2_INFO=
SUMMARY_LICENSE_3_OBJECTTYPE=EDGE_ANALYTICS
SUMMARY_LICENSE_3_FAILOVER=FALSE
SUMMARY_LICENSE_3_TOTALLICENSES=1
SUMMARY_LICENSE_3_TOTALOBJECTS=99
SUMMARY_LICENSE_3_USEDOBJECTS=1
SUMMARY_LICENSE_3_INFO=
SUMMARY_LICENSE_4_OBJECTTYPE=EDGE_LPR
SUMMARY_LICENSE_4_FAILOVER=FALSE
SUMMARY_LICENSE_4_TOTALLICENSES=1
SUMMARY_LICENSE_4_TOTALOBJECTS=8
SUMMARY_LICENSE_4_USEDOBJECTS=0
SUMMARY_LICENSE_4_INFO=
EXPIRATION_DATE=2021-05-31 00:00:00.000
EXPIRATION_DAYSLEFT=381

```

Example of return in XML:

```

<Response>
  <Code>0</Code>
  <Message>OK</Message>
  <Data>
    <Summary>
      <Count>4</Count>
      <License>
        <ObjectType>CAMERA</ObjectType>
        <Failover>FALSE</Failover>
        <TotalLicenses>4</TotalLicenses>
        <TotalObjects>168</TotalObjects>
        <UsedObjects>70</UsedObjects>
        <Info />
      </License>
      <License>
        <ObjectType>IO_DEVICE</ObjectType>
        <Failover>FALSE</Failover>
        <TotalLicenses>2</TotalLicenses>
        <TotalObjects>65</TotalObjects>
        <UsedObjects>10</UsedObjects>
        <Info />
      </License>
      <License>
        <ObjectType>EDGE_ANALYTICS</ObjectType>
        <Failover>FALSE</Failover>
        <TotalLicenses>1</TotalLicenses>
        <TotalObjects>99</TotalObjects>
      </License>
    </Summary>
  </Data>
</Response>

```

```

<UsedObjects>1</UsedObjects>
<Info />
</License>
<License>
    <ObjectType>EDGE_LPR</ObjectType>
    <Failover>FALSE</Failover>
    <TotalLicenses>1</TotalLicenses>
    <TotalObjects>8</TotalObjects>
    <UsedObjects>0</UsedObjects>
    <Info />
</License>
</Summary>
<Expiration>
    <Date>2021-05-31 00:00:00.000</Date>
    <DaysLeft>381</DaysLeft>
</Expiration>
</Data>
</Response>
```

Example of return in JSON:

```
{
    "Response": {
        "Code": 0,
        "Message": "OK",
        "Data": [
            {
                "Summary": [
                    {
                        "ObjectType": "CAMERA",
                        "Failover": false,
                        "TotalLicenses": 4,
                        "TotalObjects": 168,
                        "UsedObjects": 70,
                        "Info": ""
                    },
                    {
                        "ObjectType": "IO_DEVICE",
                        "Failover": false,
                        "TotalLicenses": 2,
                        "TotalObjects": 65,
                        "UsedObjects": 10,
                        "Info": ""
                    },
                    {
                        "ObjectType": "EDGE_ANALYTICS",
                        "Failover": false,
                        "TotalLicenses": 1,
                        "TotalObjects": 99,
                        "UsedObjects": 1,
                        "Info": ""
                    },
                    {
                        "ObjectType": "EDGE_LPR",
                        "Failover": false,

```

```
        "TotalLicenses": 1,  
        "TotalObjects": 8,  
        "UsedObjects": 0,  
        "Info": ""  
    }  
],  
"Expiration": {  
    "Date": "2021-05-31T00:00:00.000Z",  
    "DaysLeft": 381  
}  
}  
}
```

4.2.4 Requesting data of usage of the server

Requests data about the usage of the server.

Compatibility: All editions

Security level: Requires user authentication with rights to monitor the server status

Method: HTTP GET

Syntax:

```
http://<server_address>/Interface/Server/GetUsage  
[?<general_argument>[&<general_argument>...]]
```

Example 1: Requests data about the usage of the server with response in XML

<http://192.168.0.1:8601/Interface/Server/GetUsage?ResponseFormat=XML>

Example 2: Requests data about the usage of the server with response in text and authentication of the user admin

```
http://192.168.0.1:8601/Interface/Server/GetUsage?  
ResponseFormat=Text&AuthUser=admin
```

Response:

A list of parameter-value pairs is returned

HTTP Return HTTP: 200 OK

Parameters of return:

Parameter	Type	Description
PROCESSOR	Integer	Processor utilization in percentage (0 to 100)
GLOBALMEMORY	Integer64	General memory usage in bytes
SERVERMEMORY	Integer64	Server memory usage in bytes
CONNECTIONS	Integer	Total number of TCP connections of clients connected to the server
CLIENTS	Integer	Total number of distinct clients connected to the server
INPUTTRAFFIC	Integer64	Input traffic in Kbps (Kbits per second)
OUTPUTTRAFFIC	Integer64	Output traffic in Kbps (Kbits per second)

Example of return in text:

```
RESPONSE_CODE=0
RESPONSE_MESSAGE=OK
PROCESSOR=5
GLOBALMEMORY=2206560256
SERVERMEMORY=44388
CONNECTIONS=56
CLIENTS=4
INPUTTRAFFIC=5847
OUTPUTTRAFFIC=20384
```

Example of return in XML:

```
<Response>
  <Code>0</Code>
  <Message>OK</Message>
  <Data>
    <Stats>
      <Processor>2</Processor>
      <GlobalMemory>2211676160</GlobalMemory>
      <ServerMemory>44388</ServerMemory>
      <Connections>56</Connections>
      <Clients>4</Clients>
      <InputTraffic>5847</InputTraffic>
      <OutputTraffic>20384</OutputTraffic>
    </Stats>
  </Data>
</Response>
```

Example of return in JSON:

```
{
  "Response": {
    "Code": 0,
    "Message": "OK",
    "Data": {
      "Stats": {
        "Processor": 22,
        "GlobalMemory": 5874896896,
        "ServerMemory": 1053618176,
        "Connections": 118,
        "Clients": 13,
        "InputTraffic": 132288,
        "OutputTraffic": 87944
      }
    }
  }
}
```

4.2.5 Requesting status of Master / Slave

Requests the current status of Master / Slave.

Compatibility: Professional, Enterprise

Security level: Requires user authentication with rights to monitor the server status

Method: HTTP GET

Syntax:

```
http://<server_address>/Interface/Server/GetMasterSlaveStatus  
[?<general_argument>[&<general_argument>...]]
```

Example 1: Requests the status of Master/Slave with response in XML

```
http://192.168.0.1:8601/Interface/Server/GetMasterSlaveStatus?  
ResponseFormat=XML
```

Example 2: Requests the status of Master/Slave with response in text and authentication of the user admin

```
http://192.168.0.1:8601/Interface/Server/GetMasterSlaveStatus?  
ResponseFormat=Text&AuthUser=admin
```

Response:

The current status of Master / Slave is returned. If the server being queried is a Master Server, the return will include the list of all connected Slave Servers. If the server being queried is a Slave Server, the return will include the current status of the connection to the Master Server.

HTTP Return: 200 OK

Parameters of return:

Summary Fixed Parameter:

Parameter	Type	Description
SERVERTYPE	MASTER SLAVE	Type of the server

Parameters in the list of slave connections, in case the serer is a Master Server:

Parameter	Type	Description
COUNT	Integer	Number of slave connections (Not included in JSON response)

Parameter	Type	Description
ADDRESS	String	IP Address of the connected Slave Server
SERVERID	String	ID of the connected Slave Server

Parameters in case the server is a Slave Server:

Parameter	Type	Description										
MASTERSTATUS	DISCONNECTED CONNECTING AUTHENTICATED ERROR_AUTHENTICATION ERROR_NOT_MASTER_SERVER WORKING	Status of the connection to the master server <table border="1" data-bbox="864 1552 1444 1911"> <thead> <tr> <th>Name</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>DISCONNECTED</td> <td>Slave is disconnected from Master</td> </tr> <tr> <td>CONNECTING</td> <td>Slave is connected to Master</td> </tr> <tr> <td>AUTHENTICATED</td> <td>Slave is authenticating on Master</td> </tr> <tr> <td>ERROR_AUTHENTICATION</td> <td>Error authenticating on Master</td> </tr> </tbody> </table>	Name	Description	DISCONNECTED	Slave is disconnected from Master	CONNECTING	Slave is connected to Master	AUTHENTICATED	Slave is authenticating on Master	ERROR_AUTHENTICATION	Error authenticating on Master
Name	Description											
DISCONNECTED	Slave is disconnected from Master											
CONNECTING	Slave is connected to Master											
AUTHENTICATED	Slave is authenticating on Master											
ERROR_AUTHENTICATION	Error authenticating on Master											

Parameter	Type	Description	
		ERROR_NOT_MASTER_SERVER	Server specified for connection is not a Master Server
		WORKING	Connection established and working
MASTERSTATUSMSG	String	Number of days until expiration	

Example of return from a Master Server in text:

```
RESPONSE_CODE=0
RESPONSE_MESSAGE=OK
SERVERTYPE=MASTER
SLAVECONNECTIONS_COUNT=1
SLAVECONNECTIONS_CONNECTION_1_ADDRESS=192.168.0.1
SLAVECONNECTIONS_CONNECTION_1_SERVERID=9821935DB3A81292A0A57BCA52152B8E
```

Example of return from a Master Server in XML:

```
<Response>
  <Code>0</Code>
  <Message>OK</Message>
  <Data>
    <Status>
      <ServerType>MASTER</ServerType>
      <SlaveConnections>
        <Count>1</Count>
        <Connection>
          <Address>10.101.1.124</Address>
          <ServerID>9821935DB3A81292A0A57BCA52152B8E</ServerID>
        </Connection>
      </SlaveConnections>
    </Status>
  </Data>
</Response>
```

Example of return from a Master Server in JSON:

```
{
  "Response": {
    "Code": 0,
    "Message": "OK",
    "Data": {
      "Status": {
        "ServerType": "MASTER",
        "SlaveConnections": [
          {
            "Address": "192.168.0.1",
            "ServerID": "9821935DB3A81292A0A57BCA52152B8E"
          }
        ]
      }
    }
  }
}
```

```
}
```

Example of return from a Slave Server in text:

```
RESPONSE_CODE=0
RESPONSE_MESSAGE=OK
SERVERTYPE=SLAVE
MASTERSTATUS=WORKING
MASTERSTATUSMSG=Working...
```

Example of return from a Slave Server in XML:

```
<Response>
  <Code>0</Code>
  <Message>OK</Message>
  <Data>
    <Status>
      <ServerType>SLAVE</ServerType>
      <MasterStatus>WORKING</MasterStatus>
      <MasterStatusMsg>Working...</MasterStatusMsg>
    </Status>
  </Data>
</Response>
```

Example of return from a Slave Server in JSON:

```
{
  "Response": {
    "Code": 0,
    "Message": "OK",
    "Data": {
      "Status": {
        "ServerType": "SLAVE",
        "MasterStatus": "WORKING",
        "MasterStatusMsg": "Working..."
      }
    }
  }
}
```

4.3 Cameras

4.3.1 Requesting the list of cameras

Requests the list of cameras that the user has live view or playback rights

Compatibility: All editions

Security level: Requires user authentication

Method: HTTP GET

Syntax:

```
http://<server_address>/Interface/Cameras/GetCameras[?<argument=value>
[&<argument=value>...] [&<general argument>...]]
```

Arguments:

Argument	Valid values	Description																										
Cameras	APIMasks	Mask to filter the results. Specify which cameras must be returned based on the provided masks.																										
Fields	Name Description Active Model DeviceType ConnectionAddress ConnectionPort Latitude Longitude Memo MediaProfiles Group	Specifies the list of desired fields. In case this parameter is omitted, all of the fields will be sent. The fields must be separated by commas <table border="1"> <thead> <tr> <th>Name</th><th>Description</th></tr> </thead> <tbody> <tr> <td>Name</td><td>Name of the camera</td></tr> <tr> <td>Description</td><td>Description of the camera</td></tr> <tr> <td>Active</td><td>Camera is activated / deactivated</td></tr> <tr> <td>Model</td><td>Model of the camera</td></tr> <tr> <td>DeviceType</td><td>Type of devices</td></tr> <tr> <td>ConnectionAddress</td><td>Camera network address</td></tr> <tr> <td>ConnectionPort</td><td>Camera network port</td></tr> <tr> <td>Latitude</td><td>Camera latitude</td></tr> <tr> <td>Longitude</td><td>Camera longitude</td></tr> <tr> <td>Memo</td><td>General-use memo of the camera</td></tr> <tr> <td>MediaProfiles</td><td>List of camera media profiles</td></tr> <tr> <td>Group</td><td>Group that the camera belongs to</td></tr> </tbody> </table>	Name	Description	Name	Name of the camera	Description	Description of the camera	Active	Camera is activated / deactivated	Model	Model of the camera	DeviceType	Type of devices	ConnectionAddress	Camera network address	ConnectionPort	Camera network port	Latitude	Camera latitude	Longitude	Camera longitude	Memo	General-use memo of the camera	MediaProfiles	List of camera media profiles	Group	Group that the camera belongs to
Name	Description																											
Name	Name of the camera																											
Description	Description of the camera																											
Active	Camera is activated / deactivated																											
Model	Model of the camera																											
DeviceType	Type of devices																											
ConnectionAddress	Camera network address																											
ConnectionPort	Camera network port																											
Latitude	Camera latitude																											
Longitude	Camera longitude																											
Memo	General-use memo of the camera																											
MediaProfiles	List of camera media profiles																											
Group	Group that the camera belongs to																											

Example 1: Requests the list of cameras with all of the fields and response in XML`http://192.168.0.1:8601/Interface/Cameras/GetCameras?ResponseFormat=XML`**Example 2:** Requests the list of cameras with all of the fields and response in text`http://192.168.0.1:8601/Interface/Cameras/GetCameras?ResponseFormat=Text`**Example 3:** Requests the list of cameras with only name and description, response in XML and authentication of the user Admin`http://192.168.0.1:8601/Interface/Cameras/GetCameras?Fields=Name, Description&ResponseFormat=XML&AuthUser=admin`**Example 4:** Request the list of cameras starting with A, with just name and description, response in XML and authentication with Admin user`http://192.168.0.1:8601/Interface/Cameras/GetCameras?Cameras=A* & Fields=Name, Description&ResponseFormat=XML&AuthUser=admin`**Response:**

A list of all of the cameras that the user has live view or playback rights is returned. The fields returned in the will depend on the values informed in the argument **Fields**

HTTP Return: 200 OK**Parameters of return:****Fixed Parameter:**

Parameter	Type	Description
COUNT	Integer	Total number of cameras (Not included in JSON response)

Parameters in the list of cameras:

Parameter	Type	Description												
NAME	String	Name of the camera												
DESCRIPTION	String	Description of the camera												
ACTIVE	Boolean	Camera is activated / deactivated												
MODEL	String	Model of the camera												
DEVICETYPE	Integer	Type of device <table border="1" data-bbox="758 496 1395 696"> <thead> <tr> <th>Type</th><th>Description</th></tr> </thead> <tbody> <tr> <td>0</td><td>Media Device</td></tr> <tr> <td>1</td><td>IP Camera</td></tr> <tr> <td>2</td><td>Video Server</td></tr> <tr> <td>3</td><td>Network Video Recorder (NVR)</td></tr> <tr> <td>4</td><td>Digital Video Recorder (DVR)</td></tr> </tbody> </table>	Type	Description	0	Media Device	1	IP Camera	2	Video Server	3	Network Video Recorder (NVR)	4	Digital Video Recorder (DVR)
Type	Description													
0	Media Device													
1	IP Camera													
2	Video Server													
3	Network Video Recorder (NVR)													
4	Digital Video Recorder (DVR)													
CONNECTIONADDRESS	String	Camera network connection address. This information will only be available if the user has rights to camera register.												
CONNECTIONPORT	String	Camera network connection port. This information will only be available if the user has rights to camera register.												
LATITUDE	APILatLn	Camera latitude with 6 digit precision												
LONGITUDE	APILatLn	Camera longitude with 6 digit precision												
MEMO	String	Camera memo in Base64 format. Base64 format is used to properly encode special characters and line breaks that might be added to the memo by the user												
MEDIAPROFILES	String	Names of the camera media profiles, comma separated												
GROUP	DUID	ID of the group that the camera belongs to												

Example of return in text:

```

RESPONSE_CODE=0
RESPONSE_MESSAGE=OK
COUNT=2
CAMERA_1_NAME=Camera1
CAMERA_1_DESCRIPTION=My Camera
CAMERA_1_ACTIVE=TRUE
CAMERA_1_MODEL=Generic
CAMERA_1_DEVICETYPE=1
CAMERA_1_CONNECTIONADDRESS=192.168.1.1
CAMERA_1_CONNECTIONPORT=80
CAMERA_1_LATITUDE=0.000000
CAMERA_1_LONGITUDE=0.000000
CAMERA_1_MEMO=
CAMERA_1_MEDIAPROFILES=Recording,Visualization
CAMERA_1_GROUP=A6981C1B-DF35-427B-A4E6-9ED8AA31B626
CAMERA_2_NAME=Camera2
CAMERA_2_DESCRIPTION=My Video Server
CAMERA_2_ACTIVE=FALSE
CAMERA_2_MODEL=Generic
CAMERA_2_DEVICETYPE=2
CAMERA_2_CONNECTIONADDRESS=192.168.1.2
CAMERA_2_CONNECTIONPORT=80
CAMERA_2_LATITUDE=-23.630363
CAMERA_2_LONGITUDE=-46.554916

```

```
CAMERA_2_MEMO=
CAMERA_2_MEDIAPROFILES=Recording,Visualization
CAMERA_2_GROUP=00000000-0000-0000-0000-000000000000
```

Example of return in XML:

```
<Response>
  <Code>0</Code>
  <Message>OK</Message>
  <Data>
    <Cameras>
      <Count>2</Count>
      <Camera>
        <Name>Camera1</Name>
        <Description>My Camera</Description>
        <Active>TRUE</Active>
        <Model>Generic</Model>
        <DeviceType>1</DeviceType>
        <ConnectionAddress>192.168.1.1</ConnectionAddress>
        <ConnectionPort>80</ConnectionPort>
        <Latitude>0.000000</Latitude>
        <Longitude>0.000000</Longitude>
        <Memo/>
        <MediaProfiles>Recording,Visualization</MediaProfiles>
        <Group>A6981C1B-DF35-427B-A4E6-9ED8AA31B626</Group>
      </Camera>
      <Camera>
        <Name>Camera2</Name>
        <Description>My Video Server</Description>
        <Active>FALSE</Active>
        <Model>Generic</Model>
        <DeviceType>2</DeviceType>
        <ConnectionAddress>192.168.1.2</ConnectionAddress>
        <ConnectionPort>80</ConnectionPort>
        <Latitude>-23.630363</Latitude>
        <Longitude>-46.554916</Longitude>
        <Memo/>
        <MediaProfiles>Recording,Visualization</MediaProfiles>
        <Group>00000000-0000-0000-0000-000000000000</Group>
      </Camera>
    </Cameras>
  </Data>
</Response>
```

Example of return in JSON:

```
{
  "Response": {
    "Code": 0,
    "Message": "OK",
    "Data": {
      "Cameras": [
        {
          "Name": "Camera1",
          "Description": "My Camera",

```

```
        "Active": true,
        "Model": "Generic",
        "DeviceType": 1,
        "ConnectionAddress": "192.168.1.1",
        "ConnectionPort": 80,
        "Latitude": "0.000000",
        "Longitude": "0.000000",
        "Memo": "",
        "MediaProfiles": "Recording,Visualization",
        "Group": "A6981C1B-DF35-427B-A4E6-9ED8AA31B626"
    },
    {
        "Name": "Camera2",
        "Description": "My Video Server",
        "Active": false,
        "Model": "Generic",
        "DeviceType": 2,
        "ConnectionAddress": "192.168.1.2",
        "ConnectionPort": 80,
        "Latitude": "-23.630363",
        "Longitude": "-46.554916",
        "Memo": "",
        "MediaProfiles": "Recording,Visualization",
        "Group": "00000000-0000-0000-0000-000000000000"
    }
]
```

4.3.2 Requesting the list of camera groups

Requests the list of camera groups

Compatibility: All editions

Security level: Requires user authentication

Method: HTTP GET

Syntax:

```
http://<server_address>/Interface/Cameras/GetGroups[?<argument=value>
[&<argument=value>...][&<general\_argument>...]]
```

Example 1: Requests the list of cameras groups with response in XML

```
http://192.168.0.1:8601/Interface/Cameras/GetGroups?ResponseFormat=XML
```

Example 2: Requests the list of cameras groups with response in text

```
http://192.168.0.1:8601/Interface/Cameras/GetGroups?ResponseFormat=Text
```

Example 3: Requests the list of cameras groups with response in JSON

```
http://192.168.0.1:8601/Interface/Cameras/GetGroups?ResponseFormat=JSON
```

Response:

A list of all of the camera groups is returned.

HTTP Return: 200 OK

Parameters of return:**Fixed Parameter:**

Parameter	Type	Description
COUNT	Integer	Total number of camera groups (Not included in JSON response)

Parameters in the list of groups:

Parameter	Type	Description
ID	DUID	Group ID
NAME	String	Group Name
PARENT	DUID	ID of the Parent Group. ID will be zeroed if there is no Parent Group

Example of return in text:

```
RESPONSE_CODE=0
RESPONSE_MESSAGE=OK
COUNT=2
GROUP_1_ID=A6981C1B-DF35-427B-A4E6-9ED8AA31B626
GROUP_1_NAME=1
GROUP_1_PARENT=00000000-0000-0000-0000-000000000000
GROUP_2_ID=8493EC1D-B424-4187-8044-9CFE6AD3F49C
GROUP_2_NAME=2
GROUP_2_PARENT=A6981C1B-DF35-427B-A4E6-9ED8AA31B626
```

Example of return in XML:

```
<Response>
  <Code>0</Code>
  <Message>OK</Message>
  <Data>
    <Groups>
      <Count>2</Count>
      <Group>
        <ID>A6981C1B-DF35-427B-A4E6-9ED8AA31B626</ID>
        <Name>1</Name>
        <Parent>00000000-0000-0000-0000-000000000000</Parent>
      </Group>
      <Group>
        <ID>8493EC1D-B424-4187-8044-9CFE6AD3F49C</ID>
        <Name>2</Name>
        <Parent>A6981C1B-DF35-427B-A4E6-9ED8AA31B626</Parent>
      </Group>
    </Groups>
  </Data>
</Response>
```

Example of return in JSON:

```
{
  "Response": {
```

```

    "Code": 0,
    "Message": "OK",
    "Data": {
        "Groups": [
            {
                "ID": "A6981C1B-DF35-427B-A4E6-9ED8AA31B626",
                "Name": "1",
                "Parent": "00000000-0000-0000-0000-000000000000"
            },
            {
                "ID": "8493EC1D-B424-4187-8044-9CFE6AD3F49C",
                "Name": "2",
                "Parent": "A6981C1B-DF35-427B-A4E6-9ED8AA31B626"
            }
        ]
    }
}

```

4.3.3 Requesting the status of the cameras

Request the status of the cameras over which the user has live view or playback rights.

Compatibility: All editions

Security level: Requires user authentication

Method: HTTP GET

Syntax:

```
http://<server_address>/Interface/Cameras/GetStatus[?<argument=value>
[&<argument=value>...][&<general argument>...]]
```

Arguments:

Argument	Valid values	Description										
Cameras	APIMasks	Mask to filter the results. Specify which cameras must be returned based on the provided masks.										
Fields	Active Working RecordingHours RecordingHoursEstimative	Specifies the list of desired fields. In case this parameter is omitted, all of the fields will be sent. The fields must be separated by commas										
		<table border="1"> <thead> <tr> <th>Name</th><th>Description</th></tr> </thead> <tbody> <tr> <td>Active</td><td>Identify if the camera is active</td></tr> <tr> <td>Working</td><td>Identify if the camera is working</td></tr> <tr> <td>RecordingHours</td><td>Amount of recording hours</td></tr> <tr> <td>RecordingHoursEstimative</td><td>Estimative of recording hours</td></tr> </tbody> </table>	Name	Description	Active	Identify if the camera is active	Working	Identify if the camera is working	RecordingHours	Amount of recording hours	RecordingHoursEstimative	Estimative of recording hours
Name	Description											
Active	Identify if the camera is active											
Working	Identify if the camera is working											
RecordingHours	Amount of recording hours											
RecordingHoursEstimative	Estimative of recording hours											
Connections	TRUE FALSE	In case this parameter is specified, a list of all connections to the cameras will be returned. If this parameter is omitted, the default value of FALSE will be used										
Active	TRUE	In case this parameter is specified, a filter will be applied										

	FALSE	and only the cameras that matches this filter will be returned, thus, in case the value Active=TRUE is specified, the result will only include the active cameras, while if the value Active=FALSE is specified, the result will only include the deactivated cameras.
Working	TRUE FALSE	In case this parameter is specified, a filter will be applied and only the cameras that matches this filter will be returned, thus, in case the value Working=TRUE is specified, the result will only include the working cameras, while if the value Working=FALSE is specified, the result will only include the cameras that are out of order.

Example 1: Request the status of all cameras with all fields and response in XML

```
http://192.168.0.1:8601/Interface/Cameras/GetStatus?ResponseFormat=XML
```

Example 2: Request the status of all active cameras with response in text

```
http://192.168.0.1:8601/Interface/Cameras/GetStatus?Active=TRUE&ResponseFormat=Text
```

Example 3: Request the status of all active cameras starting with A, with response in text

```
http://192.168.0.1:8601/Interface/Cameras/GetStatus?Cameras=A*&Active=TRUE&ResponseFormat=Text
```

Example 4: Request the status of all active cameras that are not working, with response in XML and authentication with admin user (No password)

```
http://192.168.0.1:8601/Interface/Cameras/GetStatus?Active=TRUE&Working=FALSE&ResponseFormat=XML&AuthUser=admin
```

Response:

A list with the status of all of the cameras that the user has live view or playback rights is returned. The fields returned in the will depend on the values informed in the argument Fields

HTTP Return: 200 OK

Parameters of return:**Fixed Parameter:**

Parameter	Type	Description
COUNT	Integer	Total number of cameras (Not included in JSON response)

Parameters in the list of status of cameras:

Parameter	Type	Description
NAME	String	Name of the camera
ACTIVE	Boolean	Identify if the camera is active
WORKING	Boolean	Identify if the camera is working
ACTIVETIME	Integer	Amount of time (in seconds) since the last time the object was activated / updated
INACTIVETIME	Integer	In case the camera is not working, this parameter will indicate the amount of time (in seconds) since the camera stopped working (This time will reset to 0 once the camera start working again)
CONFIGUREDTORECORD	Boolean	Number of cameras that are configured to record.

Parameter	Type	Description
		Camera configured to record includes Motion Detection, Event Recording, Scheduled, anything other than "Do Not Record"
WRITTINGTODISK	Boolean	Number of cameras that are currently writing to the disk. Eg. If a camera is configured to motion recording but no motion is being detected it won't be writing to disk, when motion is detected then it will be included in the count
RECORDINGFPS	Integer	Current recording frame rate (Not configured FPS, but actual recording FPS)
RECORDINGHOURS	APIDouble	Number of recorded hours. This parameter will only be available if the user has rights to view the status of cameras
RECORDINGHOURSESTIMATIVE	APIDouble	Estimative of recording hours. This parameter will only be available if the user has rights to view the status of cameras
USEDDISKSPACE	Integer64	Amount of disk space that is currently being used by the camera in BYTES
CONNECTIONS	Connection List	List with all connections when parameter Connections=TRUE

Fixed parameters of connection list:

Parameter	Type	Description
COUNT	Integer	Total number of connections to the camera (Not included in JSON response)

Parameters in the list of connections:

Parameter	Type	Description
PROFILENAME	String	Name of the media profile
PROFILEDESCRIPTION	String	Description of the media profile
NODES	Integer	Number of internal nodes using this connection
VIDEO	Type	Description
STATUS	String	Status of the video connection
RECEIVEDBYTES	Integer	Total bytes received since connection was established
RECEIVEDFRAMES	Integer	Total frames received since connection was established
BYTESPERSECOND	Integer	Current BPS rate (Multiply by 8 to calculate Bits Per Second)
FRAMESPERSECOND	Integer	Current FPS rate
IFRAMEDISTANCE	Integer	Current distance of the I-Frame
AUDIO (If available)	Type	Description
STATUS	String	Status of the audio connection
RECEIVEDBYTES	Integer	Total bytes received since connection was established
RECEIVEDFRAMES	Integer	Total frames received since connection was established
BYTESPERSECOND	Integer	Current BPS rate (Multiply by 8 to calculate Bits Per Second)
FRAMESPERSECOND	Integer	Current FPS rate

Example of return in text:

```
RESPONSE_CODE=0
RESPONSE_MESSAGE=OK
COUNT=2
CAMERA_1_NAME=01
CAMERA_1_ACTIVE=TRUE
CAMERA_1_WORKING=TRUE
CAMERA_1_ACTIVETIME=1022413
CAMERA_1_INACTIVETIME=0
CAMERA_1_CONFIGUREDTORECORD=TRUE
CAMERA_1_WRITTINGTODISK=TRUE
CAMERA_1_RECORDINGFPS=8
CAMERA_1_RECORDINGHOURS=867.4
CAMERA_1_RECORDINGHOURSESTIMATIVE=867.7
CAMERA_1_USEDDISKSPACE=3376315458577
CAMERA_2_NAME=02
CAMERA_2_ACTIVE=TRUE
CAMERA_2_WORKING=TRUE
CAMERA_2_ACTIVETIME=1022413
CAMERA_2_INACTIVETIME=0
CAMERA_2_CONFIGUREDTORECORD=TRUE
CAMERA_2_WRITTINGTODISK=FALSE
CAMERA_2_RECORDINGFPS=29
CAMERA_2_RECORDINGHOURS=867.4
CAMERA_2_RECORDINGHOURSESTIMATIVE=867.7
CAMERA_2_USEDDISKSPACE=8840251239
```

Example of return in XML:

```
<Response>
  <Code>0</Code>
  <Message>OK</Message>
  <Data>
    <Cameras>
      <Count>2</Count>
      <Camera>
        <Name>01</Name>
        <Active>TRUE</Active>
        <Working>TRUE</Working>
        <ActiveTime>1022422</ActiveTime>
        <InactiveTime>0</InactiveTime>
        <ConfiguredToRecord>TRUE</ConfiguredToRecord>
        <WrittingToDisk>TRUE</WrittingToDisk>
        <RecordingFPS>8</RecordingFPS>
        <RecordingHours>867.4</RecordingHours>
        <RecordingHoursEstimative>867.7</RecordingHoursEstimative>
        <UsedDiskSpace>3376323817658</UsedDiskSpace>
      </Camera>
      <Camera>
        <Name>02</Name>
        <Active>TRUE</Active>
        <Working>TRUE</Working>
        <ActiveTime>1022422</ActiveTime>
        <InactiveTime>0</InactiveTime>
        <ConfiguredToRecord>TRUE</ConfiguredToRecord>
        <WrittingToDisk>FALSE</WrittingToDisk>
        <RecordingFPS>29</RecordingFPS>
        <RecordingHours>867.4</RecordingHours>
        <RecordingHoursEstimative>867.7</RecordingHoursEstimative>
        <UsedDiskSpace>8840251239</UsedDiskSpace>
      </Camera>
    </Cameras>
  </Data>
</Response>
```

Example of return in JSO:

```
{
  "Response": {
    "Code": 0,
    "Message": "OK",
    "Data": {
      "Cameras": [
        {
          "Name": "01",
          "Active": true,
          "Working": true,
          "ActiveTime": 1022445,
          "InactiveTime": 0,
          "ConfiguredToRecord": true,
          "WrittingToDisk": true,
          "RecordingFPS": 9,
          "RecordingHours": 867.4,
          "RecordingHoursEstimative": 867.7,
          "UsedDiskSpace": 3376348147873
        },
        {
          "Name": "02",
          "Active": true,
          "Working": true,
          "ActiveTime": 1022445,
          "InactiveTime": 0,
          "ConfiguredToRecord": true,
          "WrittingToDisk": false,
          "RecordingFPS": 30,
          "RecordingHours": 867.4,
          "RecordingHoursEstimative": 867.7,
          "UsedDiskSpace": 8840251239
        }
      ]
    }
  }
}
```

4.3.4 Activating / Deactivating cameras

Allows the activation or deactivation of multiple cameras simultaneously

Compatibility: All editions

Security level: Requires user authentication with rights to camera registration

Method: HTTP GET

Syntax:

`http://<server_address>/Interface/Cameras/Activation?<argument=value> [&<argument=value>...] [&<general argument>...]`

Arguments:

Argument	Valid values	Description
----------	--------------	-------------

Cameras*	String	List of cameras to activate or deactivate. The camera names must be separated by commas. This command is only mandatory when Action=Activate or Action=Deactivate										
Action*	Activate Deactivate ActivateAll DeactivateAll	Action to be executed <table border="1" data-bbox="701 481 1387 728"> <thead> <tr> <th>Operation</th><th>Description</th></tr> </thead> <tbody> <tr> <td>Activate</td><td>Activate the cameras on Cameras argument</td></tr> <tr> <td>Deactivate</td><td>Deactivate the cameras on Cameras argument</td></tr> <tr> <td>ActivateAll</td><td>Activate all cameras from the server</td></tr> <tr> <td>DeactivateAll</td><td>Deactivate all cameras from the server</td></tr> </tbody> </table>	Operation	Description	Activate	Activate the cameras on Cameras argument	Deactivate	Deactivate the cameras on Cameras argument	ActivateAll	Activate all cameras from the server	DeactivateAll	Deactivate all cameras from the server
Operation	Description											
Activate	Activate the cameras on Cameras argument											
Deactivate	Deactivate the cameras on Cameras argument											
ActivateAll	Activate all cameras from the server											
DeactivateAll	Deactivate all cameras from the server											

* Mandatory parameters

Example 1: Activate all cameras from the server

```
http://192.168.0.1:8601/Interface/Cameras/Activation?Action=ActivateAll
```

Example 2: Activate cameras Camera1 and Camera2

```
http://192.168.0.1:8601/Interface/Cameras/Activation?Action=Activate&Cameras=Camera1,Camera2
```

Example 3: Deactivate all cameras from the server

```
http://192.168.0.1:8601/Interface/Cameras/Activation?Action=DeactivateAll
```

Response:

Default response of API.

HTTP Return HTTP: 200 OK

Parameters of return: [Default return of API](#)

4.3.5 Requesting a live image (Snapshot)

Requests a live image (JPEG Snapshot) of the specified camera.

Compatibility: All editions

Security level: Requires user authentication

Method: HTTP GET

Syntax:

```
http://<server_address>/Interface/Cameras/GetSnapshot?<argument=value> [&<argument=value>...] [&<general\_argument>...]
```

Arguments:

Argument	Valid values	Description
Camera*	String	Name of the camera
Profile	Recording	Type of media profile:

	Visualization Mobile Custom	Recording - Request the snapshot using the default recording profile Visualization - Request the snapshot using the default visualization profile Mobile - Request the snapshot using the default mobile profile Custom - Request the snapshot using the profile specified on <code>CustomProfile</code> argument If this parameter is omitted, the default value Recording will be used										
<code>CustomProfile</code>	String	Media profile name in case the value of argument <code>Profile</code> is "Custom"										
<code>Width</code>	Integer. $X \geq 1$	Width of the snapshot image If this parameter is omitted, the default value of 352 will be used										
<code>Height</code>	Integer. $X \geq 1$	Height of the snapshot image If this parameter is omitted, the default value of 240 will be used										
<code>KeepAspectRatio</code>	TRUE FALSE	Keep the aspect ratio of the original image. When <code>Keep Aspect Ratio</code> is used, the original image will be resized within the provided <code>Width/Height</code> values, but will also maintain the original proportional which means that either <code>Width</code> or <code>Height</code> could be less than the value provided If this parameter is omitted, the default value FALSE will be used										
<code>Quality</code>	Integer. $0 \geq X \leq 100$	JPEG Compression quality of the snapshot image If this parameter is omitted, the default value of 50 will be used										
<code>Metadata</code>	TRUE FALSE	Add analytics metadata to the image (If applicable) If this parameter is omitted, the default value FALSE will be used										
<code>RenderInfo</code>	All Name Description Timestamp	Specifies the list of camera information to render on the image. In case this parameter is omitted, no camera information will be rendered. The fields must be separated by commas <table border="1" data-bbox="783 1626 1428 1805"> <thead> <tr> <th>Name</th><th>Description</th></tr> </thead> <tbody> <tr> <td>All</td><td>Render all available information</td></tr> <tr> <td>Name</td><td>Render camera name</td></tr> <tr> <td>Description</td><td>Render camera description</td></tr> <tr> <td>Timestamp</td><td>Render camera timestamp</td></tr> </tbody> </table>	Name	Description	All	Render all available information	Name	Render camera name	Description	Render camera description	Timestamp	Render camera timestamp
Name	Description											
All	Render all available information											
Name	Render camera name											
Description	Render camera description											
Timestamp	Render camera timestamp											
<code>WatermarkText</code>	String	Specifies a text that will be displayed on top of the image. If this parameter is omitted, no watermark text will be										

		rendered on the image
WatermarkColor	APIColor	Watermark text color. If this parameter is omitted, the default value White will be used
WatermarkSize	Integer. 8 >= X <= 64	Watermark text font size. If this parameter is omitted, the default value 26 will be used
WatermarkPosition	TopLeft TopRight Center BottomLeft BottomRight	Watermark text position. If this parameter is omitted, the default value BottomRight will be used

* Mandatory parameters

Example 1: Requests the image of a camera with the default values and error response in XML

```
http://192.168.0.1:8601/Interface/Cameras/GetSnapshot?Camera=Camera1&
ResponseFormat=XML
```

Example 2: Requests the image of a camera with size of 640x480, compression 70 and error response in text

```
http://192.168.0.1:8601/Interface/Cameras/GetSnapshot?Camera=Camera1&
Width=640&Height=480&Quality=70&ResponseFormat=Text
```

Example 3: Requests the image of a camera using mobile profile with size of 640x480, compression 70 and error response in text

```
http://192.168.0.1:8601/Interface/Cameras/GetSnapshot?Camera=Camera1&
Profile=Mobile&Width=640&Height=480&Quality=70&ResponseFormat=Text
```

Example 4: Requests the image of a camera using mobile profile with size of 640x480, compression 70, proportional resize, analytics metadata, name and description rendered on the image and error response in text

```
http://192.168.0.1:8601/Interface/Cameras/GetSnapshot?Camera=Camera1&
Profile=Mobile&Width=640&Height=480&Quality=70&KeepAspectRatio=TRUE&
Metadata=TRUE&RenderInfo=Name, Description&ResponseFormat=Text
```

Response:

In case of success, the JPEG image will be returned. In the case of error, a code and an error message will be returned. A error can be returned if the user does not have live viewing rights of the camera, if the camera is deactivated, if the camera was not found, if the camera is out of order or if the camera is under privacy mode.

HTTP Return: 200 OK

Parameters of return: [Default return of API](#)

4.3.6 Requesting a live media stream

Request a live media stream of the specified camera. The media stream will be sent using the coding from the chosen media profile.

Compatibility: All editions

Security level: Requires user authentication

Method: HTTP GET

Syntax:

```
http://<server_address>/Interface/Cameras/GetMediaStream?<argument=value>
[ &<argument=value>... ] [ &<general argument>... ]
```

Arguments:

Argument	Valid values	Description
Camera*	String	Camera name
Profile	Recording Visualization Mobile Custom	Type of media profile: Recording - Request the media stream using the default recording profile Visualization - Request the media stream using the default visualization profile Mobile - Request the media stream using the default mobile profile Custom - Request the media stream using the profile specified on <code>CustomProfile</code> argument If this parameter is omitted, the default value Recording will be used
CustomProfile	String	Media profile name in case the value of argument <code>Profile</code> is "Custom"
Audio	TRUE FALSE	This parameter specifies if audio should be sent. If this parameter is omitted, the default value FALSE will be used
Video	TRUE FALSE	This parameter specifies if video should be sent. If this parameter is omitted, the default value TRUE will be used If both <code>Audio</code> and <code>Video</code> parameters are FALSE, the command will fail
TimestampHeader	TRUE FALSE	Add custom HTTP Headers for frame timestamp. If this parameter is omitted, the default value FALSE will be used

* Mandatory parameters

Example 1: Request a media stream from a camera using the default values and error response in XML

```
http://192.168.0.1:8601/Interface/Cameras/GetMediaStream?Camera=Camera1&
ResponseFormat=XML
```

Example 2: Request a media stream from a camera using the default visualization profile and error response in text

```
http://192.168.0.1:8601/Interface/Cameras/GetMediaStream?Camera=Camera1&
Profile=Visualization&ResponseFormat=Text
```

Example 3: Request a media stream from a camera using the custom profile "HighResolution" and error response in text

```
http://192.168.0.1:8601/Interface/Cameras/GetMediaStream?Camera=Camera1&
Profile=Custom&CustomProfile=HighResolution&ResponseFormat=Text
```

Example 4: Request a media stream from a camera using the custom profile "HighResolution", audio,

timestamp header and error response in text

```
http://192.168.0.1:8601/Interface/Cameras/GetMediaStream?Camera=Camera1&
Profile=Custom&CustomProfile=HighResolution&Audio=TRUE&
TimestampHeader=TRUE&ResponseFormat=Text
```

Response:

In case of success, a media stream will be sent by using HTTP Multipart x-mixed-replace transmission. Each media frame is separated by the multipart boundary --DigifortBoundary.

After the multipart boundary, a little HTTP header will be sent containing the media frame type and the frame size

Header	Valid values	Description
Content-Type	Media MIME	Identify the media type
Content-Length	Integer	Specify the size (In bytes) of the media frame
DGF-FrameDate	APIDate	Frame date in APIDate format This header will only be included if <code>TimestampHeader</code> parameter is TRUE
DGF-FrameTime	APITime	Frame time in APITime format This header will only be included if <code>TimestampHeader</code> parameter is TRUE
DGF-FrameType	Media Frame MIME	Identify the media frame type

In the case of error, a code and an error message will be returned. A error can be returned if the user does not have live viewing rights of the camera, if the camera is deactivated, if the camera was not found, if the camera is out of order or if the camera is under privacy mode.

Example of JPEG media stream:

```
--DigifortBoundary
Content-Type: image/jpeg
Content-Length: 35463
DGF-FrameDate: 2014.12.01
DGF-FrameTime: 10.00.00.755
DGF-FrameType: jpeg

JPEG_DATA
JPEG_DATA

..
..
..

JPEG_DATA
--DigifortBoundary
Content-Type: image/jpeg
Content-Length: 34236
DGF-FrameDate: 2014.12.01
DGF-FrameTime: 10.00.01.120
DGF-FrameType: jpeg

JPEG_DATA
JPEG_DATA

..
..
..

JPEG_DATA
```

Example of H.264 media stream:

```
--DigifortBoundary
Content-Type: video/h264
Content-Length: 10436
DGF-FrameDate: 2014.12.01
DGF-FrameTime: 10.00.00.755
DGF-FrameType: h.264/I-Frame

H264_DATA
H264_DATA
..
..
..
H264_DATA
--DigifortBoundary
Content-Type: video/h264
Content-Length: 2548
DGF-FrameDate: 2014.12.01
DGF-FrameTime: 10.00.01.120
DGF-FrameType: h.264/P-Frame

H264_DATA
H264_DATA
..
..
..
H264_DATA
```

Example of H.264 media stream with audio:

```
--DigifortBoundary
Content-Type: video/h264
Content-Length: 10436
DGF-FrameDate: 2014.12.01
DGF-FrameTime: 10.00.00.755
DGF-FrameType: h.264/I-Frame

H264_DATA
H264_DATA
..
..
..
H264_DATA
--DigifortBoundary
Content-Type: audio/basic
Content-Length: 240
DGF-FrameDate: 2014.12.01
DGF-FrameTime: 10.00.01.000
DGF-FrameType: audio/basic

AUDIO_DATA
AUDIO_DATA
..
..
..
AUDIO_DATA
--DigifortBoundary
Content-Type: video/h264
Content-Length: 584
DGF-FrameDate: 2014.12.01
DGF-FrameTime: 10.00.01.120
DGF-FrameType: h.264/P-Frame

H264_DATA
H264_DATA
..
..
..
H264_DATA
```

HTTP Return: 200 OK

Parameters of return: [Default return of API](#)

4.3.7 Requesting a live video stream (MJPEG)

Requests a live video stream (MJPEG) of the specified camera.

Warning: This command can overload the server processing. This will occur because in order to deliver the live MJPEG video stream the server will transcode the video that it is receiving from the camera.

Compatibility: All editions

Security level: Requires user authentication

Method: HTTP GET

Syntax:

```
http://<server_address>/Interface/Cameras/GetJPEGStream?<argument=value>
[&<argument=value>...] [&<general argument>...]
```

Arguments:

Argument	Valid values	Description
Camera*	String	Camera name
Profile	Recording Visualization Custom	Type of media profile: Recording - Request the media stream using the default recording profile Visualization - Request the media stream using the default visualization profile Custom - Request the media stream using the profile specified on <code>CustomProfile</code> argument If this parameter is omitted, the default value Recording will be used
CustomProfile	String	Media profile name in case the value of argument <code>Profile</code> is "Custom"
Width	Integer. X >= 1	Image width If this parameter is omitted, the default value of 352 will be used
Height	Integer. X >= 1	Image height If this parameter is omitted, the default value of 240 will be used
KeepAspectRatio	TRUE FALSE	Keep the aspect ratio of the original image. When Keep Aspect Ratio is used, the original image will be resized within the provided Width/Height values, but will also maintain the original proportional which means that either Width or Height could be less than the value provided If this parameter is omitted, the default value FALSE will be used
Quality	Integer. 0 >= X <= 100	JPEG image compression quality value If this parameter is omitted, the default value of 50 will be used
Metadata	TRUE FALSE	Add analytics metadata to the image (If applicable) If this parameter is omitted, the default value FALSE will be used
RenderInfo	All Name Description Timestamp	Specifies the list of camera information to render on the image. In case this parameter is omitted, no camera information will be rendered.

		The fields must be separated by commas <table border="1"> <thead> <tr> <th>Name</th><th>Description</th></tr> </thead> <tbody> <tr> <td>All</td><td>Render all available information</td></tr> <tr> <td>Name</td><td>Render camera name</td></tr> <tr> <td>Description</td><td>Render camera description</td></tr> <tr> <td>Timestamp</td><td>Render camera timestamp</td></tr> </tbody> </table>	Name	Description	All	Render all available information	Name	Render camera name	Description	Render camera description	Timestamp	Render camera timestamp
Name	Description											
All	Render all available information											
Name	Render camera name											
Description	Render camera description											
Timestamp	Render camera timestamp											
WatermarkText	String	Specifies a text that will be displayed on top of the image. If this parameter is omitted, no watermark text will be rendered on the image										
WatermarkColor	APIColor	Watermark text color. If this parameter is omitted, the default value White will be used										
WatermarkSize	Integer. 8 >= X <= 64	Watermark text font size. If this parameter is omitted, the default value 26 will be used										
WatermarkPosition	TopLeft TopRight Center BottomLeft BottomRight	Watermark text position. If this parameter is omitted, the default value BottomRight will be used										
FPS	Integer. 1 >= X <= 30	Amount of frames per second If this parameter is omitted, the default value of 30 will be used										
TimestampHeader	TRUE FALSE	Add custom HTTP Headers for frame timestamp. If this parameter is omitted, the default value FALSE will be used										
Audio	TRUE FALSE	Request synchronized audio with video. If this parameter is omitted, the default value FALSE will be used.										
AudioFormat	PCM WAV AAC AAC-MP4	Audio output format. If this parameter is omitted, the default value PCM will be used. <table border="1"> <thead> <tr> <th>Name</th><th>Description</th></tr> </thead> <tbody> <tr> <td>PCM</td><td>Pure PCM audio</td></tr> <tr> <td>WAV</td><td>PCM audio with WAV header</td></tr> <tr> <td>AAC</td><td>AAC with ADTS header</td></tr> <tr> <td>AAC-MP4</td><td>AAC with MP4 config header</td></tr> </tbody> </table>	Name	Description	PCM	Pure PCM audio	WAV	PCM audio with WAV header	AAC	AAC with ADTS header	AAC-MP4	AAC with MP4 config header
Name	Description											
PCM	Pure PCM audio											
WAV	PCM audio with WAV header											
AAC	AAC with ADTS header											
AAC-MP4	AAC with MP4 config header											

* Mandatory parameters

Note: The server will not be able to deliver more frames than it is receiving from the camera by using the recording profile

Example 1: Request an MJPEG video stream from a camera with default values and error response in XML

```
http://192.168.0.1:8601/Interface/Cameras/GetJPEGStream?Camera=Camera1&
ResponseFormat=XML
```

Example 2: Request an MJPEG video stream from a camera with 640x480 resolution, compression 70, 10 frames per second and error response in text

```
http://192.168.0.1:8601/Interface/Cameras/GetJPEGStream?Camera=Camera1&Width=640&Height=480&Quality=70&FPS=10&ResponseFormat=Text
```

Example 3: Request an MJPEG video stream from a camera using mobile profile with 640x480 resolution, compression 70, 10 frames per second, timestamp header and error response in text

```
http://192.168.0.1:8601/Interface/Cameras/GetJPEGStream?Camera=Camera1&Profile=Mobile&Width=640&Height=480&Quality=70&FPS=10&TimestampHeader=TRUE&ResponseFormat=Text
```

Example 4: Request an MJPEG video stream from a camera using mobile profile with 640x480 resolution, compression 70, 10 frames per second, proportional resizing, timestamp header, analytics metadata, name and description rendered on the image and error response in text

```
http://192.168.0.1:8601/Interface/Cameras/GetJPEGStream?Camera=Camera1&Profile=Mobile&Width=640&Height=480&KeepAspectRatio=TRUE&Quality=70&FPS=10&TimestampHeader=TRUE&Metadata=TRUE&RenderInfo=Name,Description&ResponseFormat=Text
```

Response:

In case of success, a stream of JPEG images will be sent by using HTTP Multipart x-mixed-replace transmission. Each image is separated by the multipart boundary --DigifortBoundary.

In the case of error, a code and an error message will be returned. A error can be returned if the user does not have live viewing rights of the camera, if the camera is deactivated, if the camera was not found, if the camera is out of order or if the camera is under privacy mode.

After the multipart boundary, a little HTTP header will be sent containing extra information

Header	Valid values	Description
Content-Type	Media MIME	Identify the media type
Content-Length	Integer	Specify the size (In bytes) of the media frame
DGF-FrameDate	APIDate	Frame date in APIDate format This header will only be included if TimestampHeader parameter is TRUE
DGF-FrameTime	APITime	Frame time in APITime format This header will only be included if TimestampHeader parameter is TRUE
DGF-AACMP4Config	String	MP4 Setup Header in Base64 format This header will only be included if AudioFormat parameter is AAC-MP4
DGF-PCMChannels	Integer	Number of PCM channels This header will only be included if AudioFormat parameter is PCM
DGF-PCMSamplesPerSecond	Integer	Number of Samples per Second This header will only be included if AudioFormat parameter is PCM

DGF-PCMBitsPerSample	Integer	Number of Bits per Sample
		This header will only be included if <code>AudioFormat</code> parameter is PCM

Example of JPEG image stream:

```
--DigifortBoundary
Content-Type: image/jpeg
Content-Length: 35463
DGF-FrameDate: 2014.12.01
DGF-FrameTime: 10.00.00.755

JPEG_DATA
JPEG_DATA
..
..
..
JPEG_DATA
--DigifortBoundary
Content-Type: image/jpeg
Content-Length: 34236
DGF-FrameDate: 2014.12.01
DGF-FrameTime: 10.00.01.120

JPEG_DATA
JPEG_DATA
..
..
..
JPEG_DATA
```

HTTP Return: 200 OK**Parameters of return:** [Default return of API](#)**4.3.8 Requesting a live video stream (MP4)**

Requests a live video stream (MP4) of the specified camera. The system will return a Fragmented MP4 stream that can be directly displayed by any compatible web browser.

Warning: This command can overload the server CPU. This will occur because in order to deliver the live MP4 video stream the server may have to transcode the video that it is receiving from the camera.

Compatibility: All editions**Security level:** Requires user authentication**Method:** HTTP GET**Syntax:**

```
http://<server_address>/Interface/Cameras/GetMP4Stream?<argument=value>
[&<argument=value>...] [&<general argument>...]
```

Arguments:

Argument	Valid values	Description
Camera*	String	Camera name
Profile	Recording Visualization Custom	Type of media profile: Recording - Request the media stream using the default recording profile Visualization - Request the media stream using the default visualization profile Custom - Request the media stream using the profile specified on CustomProfile argument If this parameter is omitted, the default value Recording will be used
CustomProfile	String	Media profile name in case the value of argument Profile is "Custom"
Video	TRUE FALSE	Specifies if the MP4 stream will contain video. If this parameter is omitted, the default value TRUE will be used
Audio	TRUE FALSE	Specifies if the MP4 stream will contain audio. If this parameter is omitted, the default value FALSE will be used
Transcode	TRUE FALSE	Force transcoding of the source video to H.264 format. If this parameter is omitted, the default value FALSE will be used. Transcoding will be forced to TRUE If the source video is not in H.264 format or if parameters such as Metadata, RenderInfo, Watermark are activated or if the user watermark is forced by the user account. Using transcoding will put heavy load to the server
Width	Integer. X >= -1 && X != 0	Custom image width to be used when Transcode=TRUE Specify value -1 to use the original image width If this parameter is omitted, the default value -1 will be used
Height	Integer. X >= -1 && X != 0	Custom image height to be used when Transcode=TRUE Specify value -1 to use the original image height If this parameter is omitted, the default value -1 will be used
KeepAspectRatio	TRUE FALSE	Keep the aspect ratio of the original image. When Keep Aspect Ratio is used, the original image will be resized within the provided Width/Height values, but will also maintain the original proportional which means that either Width or Height could be less than the value provided If this parameter is omitted, the default value TRUE will be used

Quality	Integer. 1 >= X <= 100	H.264 compression quality value If this parameter is omitted, the default value of 70 will be used										
Metadata	TRUE FALSE	Add analytics metadata to the image (If applicable) If this parameter is omitted, the default value FALSE will be used										
RenderInfo	All Name Description Timestamp	Specifies the list of camera information to render on the image. In case this parameter is omitted, no camera information will be rendered. The fields must be separated by commas <table border="1" data-bbox="816 665 1428 844"> <thead> <tr> <th>Name</th><th>Description</th></tr> </thead> <tbody> <tr> <td>All</td><td>Render all available information</td></tr> <tr> <td>Name</td><td>Render camera name</td></tr> <tr> <td>Description</td><td>Render camera description</td></tr> <tr> <td>Timestamp</td><td>Render camera timestamp</td></tr> </tbody> </table>	Name	Description	All	Render all available information	Name	Render camera name	Description	Render camera description	Timestamp	Render camera timestamp
Name	Description											
All	Render all available information											
Name	Render camera name											
Description	Render camera description											
Timestamp	Render camera timestamp											
WatermarkText	String	Specifies a text that will be displayed on top of the image. If this parameter is omitted, no watermark text will be rendered on the image										
WatermarkColor	APIColor	Watermark text color. If this parameter is omitted, the default value White will be used										
WatermarkSize	Integer. 8 >= X <= 64	Watermark text font size. If this parameter is omitted, the default value 26 will be used										
WatermarkPosition	TopLeft TopRight Center BottomLeft BottomRight	Watermark text position. If this parameter is omitted, the default value BottomRight will be used										

* Mandatory parameters

Example 1: Request an MP4 video stream from a camera with default values and error response in XML
`http://192.168.0.1:8601/Interface/Cameras/GetMP4Stream?Camera=Camera1&ResponseFormat=XML`

Example 2: Request an MP4 video stream from a camera with transcoding, 640x480 resolution, compression 70 and error response in text
`http://192.168.0.1:8601/Interface/Cameras/GetMP4Stream?Camera=Camera1&Transcode=TRUE&Width=640&Height=480&Quality=70&ResponseFormat=Text`

Example 3: Request an MP4 video stream from a camera using mobile profile with transcoding, 640x480 resolution with proportional resizing, analytics metadata, compression 70, render all info and error response in text
`http://192.168.0.1:8601/Interface/Cameras/GetMP4Stream?Camera=Camera1&Profile=Mobile&Transcode=TRUE&Width=640&Height=480&KeepAspectRatio=TRUE&Quality=70&Metadata=TRUE&RenderInfo=All&ResponseFormat=Text`

Response:

In case of success, the MP4 stream will be sent

In the case of error, a code and an error message will be returned. A error can be returned if the user

does not have live viewing rights of the camera, if the camera is deactivated, if the camera was not found, if the camera is out of order or if the camera is under privacy mode.

HTTP Return: 200 OK

Parameters of return: [Default return of API](#)

4.3.9 Requesting a live audio stream

Requests a live audio stream of the specified camera. The audio stream provided will be in PCM, WAV or AAC format. PCM and WAV are pure digital audio and can be reproduced without the need of any decoders while AAC will require an AAC decoder. Although PCM and WAV will be heavier in the network, it is also easy to play and should not have any significant impact in LAN networks.

Compatibility: All editions

Security level: Requires user authentication

Method: HTTP GET

Syntax:

```
http://<server_address>/Interface/Cameras/GetAudioStream?<argument=value>
[&<argument=value>...] [&<general argument>...]
```

Arguments:

Argument	Valid values	Description										
Camera*	String	Camera name										
Profile	Recording Visualization Custom	Type of media profile: Recording - Request the media stream using the default recording profile Visualization - Request the media stream using the default visualization profile Custom - Request the media stream using the profile specified on CustomProfile argument If this parameter is omitted, the default value Recording will be used										
CustomProfile	String	Media profile name in case the value of argument Profile is "Custom"										
Format	PCM WAV AAC AAC-MP4	Audio output format If this parameter is omitted, the default value PCM will be used <table border="1" data-bbox="747 1628 1421 1797"> <thead> <tr> <th>Name</th><th>Description</th></tr> </thead> <tbody> <tr> <td>PCM</td><td>Pure PCM audio</td></tr> <tr> <td>WAV</td><td>PCM audio with WAV header</td></tr> <tr> <td>AAC</td><td>AAC with ADTS header</td></tr> <tr> <td>AAC-MP4</td><td>AAC with MP4 config header</td></tr> </tbody> </table>	Name	Description	PCM	Pure PCM audio	WAV	PCM audio with WAV header	AAC	AAC with ADTS header	AAC-MP4	AAC with MP4 config header
Name	Description											
PCM	Pure PCM audio											
WAV	PCM audio with WAV header											
AAC	AAC with ADTS header											
AAC-MP4	AAC with MP4 config header											
TimestampHeader	TRUE FALSE	Add custom HTTP Headers for frame timestamp. If this parameter is omitted, the default value FALSE will be										

	used
--	------

* Mandatory parameters

Example 1: Request decoded PCM audio stream from camera "Camera1" with default values and error response in XML

```
http://192.168.0.1:8601/Interface/Cameras/GetAudioStream?Camera=Camera1&ResponseFormat=XML
```

Example 2: Request decoded PCM audio stream from camera "Camera1" using visualization profile, timestamp header and error response in text

```
http://192.168.0.1:8601/Interface/Cameras/GetAudioStream?Camera=Camera1&Profile=Visualization&TimestampHeader=TRUE&ResponseFormat=Text
```

Example 3: Request decoded PCM audio stream with WAV headers from camera "Camera1" using visualization profile, timestamp header and error response in text

```
http://192.168.0.1:8601/Interface/Cameras/GetAudioStream?Camera=Camera1&Profile=Visualization&Format=WAV&TimestampHeader=TRUE&ResponseFormat=Text
```

Response:

In case of success, a stream of audio data in chunks will be sent by using HTTP Multipart x-mixed-replace transmission. Each chunk of data is separated by the multipart boundary --DigifortBoundary.

Each multipart header will contain the following custom HTTP headers:

Header	Valid values	Description
DGF-PCMChannels	Integer	Amount of channels of the PCM stream
DGF-PCMSamplesPerSecond	Integer	Amount of samples per second of the PCM stream
DGF-PCMBitsPerSample	Integer	Amount of bits per sample of the PCM stream
DGF-FrameDate	APIDate	Frame date in APIDate format This header will only be included if <code>TimestampHeader</code> parameter is TRUE
DGF-FrameTime	APITime	Frame time in APITime format This header will only be included if <code>TimestampHeader</code> parameter is TRUE
DGF-AACMP4Config	String	MP4 Setup Header in Base64 format This header will only be included if <code>Format</code> parameter is AAC-MP4

In the case of error, a code and an error message will be returned. A error can be returned if the user does not have live viewing rights of the camera, if the camera is deactivated, if the camera was not found, if the camera is out of order or if the camera is under privacy mode.

Example of JPEG image stream:

```
--DigifortBoundary
Content-Type: audio/pcm
Content-Length: 258
DGF-PCMChannels: 1
DGF-PCMSamplesPerSecond: 16000
DGF-PCMBitsPerSample: 16
DGF-FrameDate: 2014.12.01
DGF-FrameTime: 10.00.01.000

PCM_DATA
PCM_DATA
..
..
..
PCM_DATA
--DigifortBoundary
Content-Type: audio/pcm
Content-Length: 258
DGF-PCMChannels: 1
DGF-PCMSamplesPerSecond: 16000
DGF-PCMBitsPerSample: 16
DGF-FrameDate: 2014.12.01
DGF-FrameTime: 10.00.01.033

PCM_DATA
PCM_DATA
..
..
..
PCM_DATA
```

HTTP Return: 200 OK

Parameters of return: [Default return of API](#)

4.3.10 Sending a live audio stream

This command should be used to send audio stream to a camera for two-way audio communication.

Important: At this time, this command will only support PCM format with 1 channel, 8khz and 16bits per sample

Compatibility: All editions

Security level: Requires user authentication

Method: HTTP POST

Syntax:

```
http://<server_address>/Interface/Cameras/SendAudioStream?<argument=value>
[&<argument=value>...] [&<general\_argument>...]
```

Arguments:

Argument	Valid values	Description
Camera*	String	Camera name

* Mandatory parameters

The PCM data should be sent over the POST request.

This command will support HTTP Keep-Alive connection.

Important: For sending a continuous flow of PCM data, you should send each chunk of data within a separated request, by using HTTP Keep-Alive you can re-use existing connection for sending all chunks of data. We recommend that you send about 300 milliseconds of data within each POST request. Do not send over 1 second of audio inside a single POST command since the server will only send the received data after it is completely received, which could introduce delays to the communication.

Content-Type must be audio/basic

Format must be PCM with 1 channel, 8khz and 16bits per sample

Example 1: Send PCM audio stream to camera "Camera1"

```
POST /Interface/Cameras/SendAudioStream?Camera=Camera1 HTTP/1.1
```

Host: 192.168.0.1

Content-Length: 1600

Authorization: Basic YWRtaW46

Content-Type: audio/basic

Connection: Keep-Alive

PCM_DATA

PCM_DATA

..

..

..

PCM_DATA

```
POST /Interface/Cameras/SendAudioStream?Camera=Camera1 HTTP/1.1
```

Host: 192.168.0.1

Content-Length: 1600

Authorization: Basic YWRtaW46

Content-Type: audio/basic

Connection: Keep-Alive

PCM_DATA

PCM_DATA

..

..

..

PCM_DATA

```
POST /Interface/Cameras/SendAudioStream?Camera=Camera1 HTTP/1.1
```

Host: 192.168.0.1

Content-Length: 1600

Authorization: Basic YWRtaW46

Content-Type: audio/basic

Connection: Keep-Alive

```

PCM_DATA
PCM_DATA
..
..
..
PCM DATA

```

Response:

In case the PCM stream was sent successfully, the default API result will be returned for each POST command.

HTTP Return: 200 OK

Parameters of return: [Default return of API](#)

4.3.11 Playback

4.3.11.1 Requesting a recorded image (Snapshot)

Requests a recorded image (JPEG Snapshot) of a specified camera.

Compatibility: All editions

Security level: Requires user authentication

Method: HTTP GET

Syntax:

```
http://<server_address>/Interface/Cameras/Playback/GetSnapshot?
<argument=value>[&<argument=value>...][&<general\_argument>...]
```

Arguments:

Argument	Valid values	Description
Camera*	String	Name of the camera
Date*	APIDate	Date of the image in APIDate format
Time*	APITime	Time of the image in APITime format
Width	Integer. X >= -1 && X != 0	Width of the snapshot image Use the value -1 for the original width of the recorded image If this parameter is omitted, the default value of 352 will be used
Height	Integer. X >= -1 && X != 0	Height of the snapshot image Use the value -1 for the original height of the recorded image If this parameter is omitted, the default value of 240 will be used
KeepAspectRatio	TRUE FALSE	Keep the aspect ratio of the original image. When Keep Aspect Ratio is used, the original image will be resized within the provided Width/Height values, but will also

		<p>maintain the original proportional which means that either Width or Height could be less than the value provided</p> <p>If this parameter is omitted, the default value FALSE will be used</p>										
Quality	Integer. 0 >= X <= 100	<p>Quality of the JPEG compression of the snapshot image</p> <p>If this parameter os omitted, the default value of 50 will be used</p>										
Metadata	TRUE FALSE	<p>TRUE - Send recorded footage along with rendered recorded metadata if applicable</p> <p>FALSE - Send normal recorded footage</p> <p>If this parameter is omitted, the default value FALSE will be used</p>										
RenderInfo	All Name Description Timestamp	<p>Specifies the list of camera information to render on the image. In case this parameter is omitted, no camera information will be rendered.</p> <p>The fields must be separated by commas</p> <table border="1"> <thead> <tr> <th>Name</th><th>Description</th></tr> </thead> <tbody> <tr> <td>All</td><td>Render all available information</td></tr> <tr> <td>Name</td><td>Render camera name</td></tr> <tr> <td>Description</td><td>Render camera description</td></tr> <tr> <td>Timestamp</td><td>Render camera timestamp</td></tr> </tbody> </table>	Name	Description	All	Render all available information	Name	Render camera name	Description	Render camera description	Timestamp	Render camera timestamp
Name	Description											
All	Render all available information											
Name	Render camera name											
Description	Render camera description											
Timestamp	Render camera timestamp											
WatermarkText	String	Specifies a text that will be displayed on top of the image. If this parameter is omitted, no watermark text will be rendered on the image										
WatermarkColor	APIColor	Watermark text color. If this parameter is omitted, the default value White will be used										
WatermarkSize	Integer. 8 >= X <= 64	Watermark text font size. If this parameter is omitted, the default value 26 will be used										
WatermarkPosition	TopLeft TopRight Center BottomLeft BottomRight	Watermark text position. If this parameter is omitted, the default value BottomRight will be used										
SourceStorage	HOT COLD	<p>This parameter specifies the source storage used to retrieve the image.</p> <table border="1"> <thead> <tr> <th>Name</th><th>Description</th></tr> </thead> <tbody> <tr> <td>HOT</td><td>Hot storage is the main storage of the system, where all recordings are stored</td></tr> <tr> <td>COLD</td><td>Cold storage refers to archived recordings by the Archiving system</td></tr> </tbody> </table> <p>If this parameter is omitted, the default value HOT will be used.</p>	Name	Description	HOT	Hot storage is the main storage of the system, where all recordings are stored	COLD	Cold storage refers to archived recordings by the Archiving system				
Name	Description											
HOT	Hot storage is the main storage of the system, where all recordings are stored											
COLD	Cold storage refers to archived recordings by the Archiving system											

* Mandatory parameters

Example 1: Requests the image of a camera of the 15th of November, 2009 at 10:00:00 with error response in XML

```
http://192.168.0.1:8601/Interface/Cameras/Playback/GetSnapshot?
Camera=Cameral&Date=2009.11.15&Time=10.00.00&ResponseFormat=XML
```

Example 2: Requests the image of a camera of the 20th of November, 2009 at 15:55:20 with size of 640x480, compression 70 and error response in text

```
http://192.168.0.1:8601/Interface/Cameras/Playback/GetSnapshot?
Camera=Cameral&Date=2009.11.20&Time=15.55.20&Width=640&Height=480&
Quality=70&ResponseFormat=Text
```

Example 3: Requests the image of a camera of the 20th of November, 2009 at 15:55:20 with size of 640x480, compression 70, from COLD storage and error response in text

```
http://192.168.0.1:8601/Interface/Cameras/Playback/GetSnapshot?
Camera=Cameral&Date=2009.11.20&Time=15.55.20&Width=640&Height=480&
Quality=70&SourceStorage=COLD&ResponseFormat=Text
```

Example 4: Requests the image of a camera of the 20th of November, 2009 at 15:55:20 with size of 640x480, compression 70, original image proportions, analytics metadata, name and description rendered on the image, from COLD storage and error response in text

```
http://192.168.0.1:8601/Interface/Cameras/Playback/GetSnapshot?
Camera=Cameral&Date=2009.11.20&Time=15.55.20&Width=640&Height=480&
KeepAspectRatio=TRUE&Quality=70&Metadata=TRUE&RenderInfo=Name,Description&
SourceStorage=COLD&ResponseFormat=Text
```

Response:

In the case of success, a JPEG image will be returned. In the case of error, a code and an error message will be returned. An error can be returned if the user does not have viewing rights of the recording of the camera, if the camera is not found or if the image was not found.

HTTP Return: 200 OK

Parameters of return: [Default return of API](#)

4.3.11.2 Requesting a recorded media stream

Requests a recorded media stream from the specified camera. The media stream will be sent on its original format (No media transcoding will be performed).

Compatibility: All editions

Security level: Requires user authentication

Method: HTTP GET

Syntax:

```
http://<server_address>/Interface/Cameras/Playback/GetMediaStream?
<argument=value>[&<argument=value>...][&<general\_argument>...]
```

Arguments:

Argument	Valid values	Description
Camera*	String	Camera name
StartDate*	APIDate	Start date of recordings on APIDate format

StartTime*	APITime	Start time of recordings on APITime format								
EndDate	APIDate	End date of recordings on APIDate format If this parameter is omitted, the current date will be used								
EndTime	APITime	End time of recordings on APITime format If this parameter is omitted, the current time will be used								
Push	TRUE FALSE	TRUE - Send recorded footage as fast as possible, with no playback speed control (Use when downloading recordings). FALSE - Send recorded footage with playback speed control of 1x If this parameter is omitted, the default value FALSE will be used								
Audio	TRUE FALSE	TRUE - Send audio multiplexed with video (If audio is recorded) FALSE - Send video only If this parameter is omitted, the default value FALSE will be used								
SourceStorage	HOT COLD EDGE	This parameter specifies the source storage used to retrieve the media. <table border="1"> <thead> <tr> <th>Name</th><th>Description</th></tr> </thead> <tbody> <tr> <td>HOT</td><td>Hot storage is the main storage of the system, where all recordings are stored</td></tr> <tr> <td>COLD</td><td>Cold storage refers to archived recordings by the Archiving system</td></tr> <tr> <td>EDGE</td><td>Edge storage refers to the storage of recordings inside the device</td></tr> </tbody> </table> If this parameter is omitted, the default value HOT will be used.	Name	Description	HOT	Hot storage is the main storage of the system, where all recordings are stored	COLD	Cold storage refers to archived recordings by the Archiving system	EDGE	Edge storage refers to the storage of recordings inside the device
Name	Description									
HOT	Hot storage is the main storage of the system, where all recordings are stored									
COLD	Cold storage refers to archived recordings by the Archiving system									
EDGE	Edge storage refers to the storage of recordings inside the device									

* Mandatory parameters

Example 1: Request recordings of camera "Camera1" from January 01, 2014 10:00:00AM to January 02, 2014 09:00:00AM with error response in XML

```
http://192.168.0.1:8601/Interface/Cameras/Playback/GetMediaStream?  
Camera=Camera1&StartDate=2014.01.01&StartTime=10.00.00.000&  
EndDate=2014.01.02&EndTime=09.00.00.000&ResponseFormat=XML
```

Example 2: Request recordings of camera "Camera1" from January 01, 2014 to date, with audio, without playback speed control and error response in text

```
http://192.168.0.1:8601/Interface/Cameras/Playback/GetMediaStream?  
Camera=Camera1&StartDate=2014.01.01&StartTime=00.00.00.000&Audio=TRUE&  
Push=TRUE&ResponseFormat=Text
```

Example 3: Request recordings of camera "Camera1" from January 25, 2014 to date, with audio and playback speed control

```
http://192.168.0.1:8601/Interface/Cameras/Playback/GetMediaStream?  
Camera=Camera1&StartDate=2014.01.20&StartTime=00.00.00.000&Audio=TRUE
```

Example 4: Request recordings of camera "Camera1" from January 25, 2014 to date, with audio and playback speed control from cold storage

```
http://192.168.0.1:8601/Interface/Cameras/Playback/GetMediaStream?  
Camera=Camera1&StartDate=2014.01.20&StartTime=00.00.00.000&Audio=TRUE&
```

```
SourceStorage=COLD
```

Response:

In case of success, a media stream will be sent by using HTTP Multipart x-mixed-replace transmission. Each media frame is separated by the multipart boundary --DigifortBoundary.

The main HTTP response header will contain the following custom headers:

Header	Valid values	Description
DGF-FrameCount	Integer	Specifies the amount of recorded frames that will be sent within the transmission

After the multipart boundary, a little HTTP header will be sent containing information about the frame:

Header	Valid values	Description
Content-Type	Media MIME	Identify the media type
Content-Length	Integer	Specify the size (In bytes) of the media frame
DGF-FrameNumber	Integer	Frame number
DGF-FrameDate	APIDate	Frame date in APIDate format
DGF-FrameTime	APITime	Frame time in APITime format
DGF-FrameType	Media Frame MIME	Identify the media frame type

In the case of error, a code and an error message will be returned. A error can be returned if the user does not have live viewing rights of the camera, if the camera is deactivated or if the camera was not found.

In case no recordings are found the connection will be gracefully closed and the HTTP custom header DGF-FrameCount will contain value 0

Example of JPEG media stream:

```
HTTP/1.1 200 OK
Connection: close
Content-Type: multipart/x-mixed-replace; boundary=--DigifortBoundary
DGF-FrameCount: 103531

--DigifortBoundary
Content-Type: image/jpeg
Content-Length: 35463
DGF-FrameNumber: 1
DGF-FrameDate: 2014.12.01
DGF-FrameTime: 10.00.00.755
DGF-FrameType: jpeg

JPEG_DATA
JPEG_DATA
..
..
..
JPEG_DATA
--DigifortBoundary
Content-Type: image/jpeg
Content-Length: 34236
```

```
DGF-FrameNumber: 2
DGF-FrameDate: 2014.12.01
DGF-FrameTime: 10.00.01.126
DGF-FrameType: jpeg

JPEG_DATA
JPEG_DATA
..
..
..
JPEG DATA
```

Example of H.264 media stream

```
HTTP/1.1 200 OK
Connection: close
Content-Type: multipart/x-mixed-replace; boundary=--DigifortBoundary
DGF-FrameCount: 47263

--DigifortBoundary
Content-Type: video/h264
Content-Length: 10436
DGF-FrameNumber: 1
DGF-FrameDate: 2014.12.01
DGF-FrameTime: 10.00.00.231
DGF-FrameType: h.264/I-Frame

H264_DATA
H264_DATA
..
..
..
H264_DATA
--DigifortBoundary
Content-Type: video/h264
Content-Length: 2548
DGF-FrameNumber: 2
DGF-FrameDate: 2014.12.01
DGF-FrameTime: 10.00.00.438
DGF-FrameType: h.264/P-Frame

H264_DATA
H264_DATA
..
..
..
H264_DATA
```

HTTP Return: 200 OK

Parameters of return: [Default return of API](#)

4.3.11.3 Requesting a recorded media stream (MJPEG)

Requests a recorded media stream from the specified camera. The media stream will be sent in MJPEG format.

Warning: This command can overload the server processing. This will occur because in order to deliver the live MJPEG video stream the server will transcode the video that it is receiving from the camera.

Compatibility: All editions

Security level: Requires user authentication

Method: HTTP GET

Syntax

```
http://<server_address>/Interface/Cameras/Playback/GetJPEGStream?  
<argument=value>[&<argument=value>... ] [&<general_argument>... ]
```

Arguments:

Argument	Valid values	Description
Camera*	String	Camera name
StartDate*	APIDate	Start date of recordings on APIDate format
StartTime*	APITime	Start time of recordings on APITime format
EndDate	APIDate	End date of recordings on APIDate format If this parameter is omitted, the current date will be used
EndTime	APITime	End time of recordings on APITime format If this parameter is omitted, the current time will be used
Width	Integer. X >= -1 && X != 0	Custom image width Specify value -1 to use the original recorded image width If this parameter is omitted, the default value -1 will be used
Height	Integer. X >= -1 && X != 0	Custom image height Specify value -1 to use the original recorded image height If this parameter is omitted, the default value -1 will be used
KeepAspectRatio	TRUE FALSE	Keep the aspect ratio of the original image. When Keep Aspect Ratio is used, the original image will be resized within the provided Width/Height values, but will also maintain the original proportional which means that either Width or Height could be less than the value provided If this parameter is omitted, the default value FALSE will be used
Quality	Integer. 0 >= X <= 100	JPEG compression quality If this parameter is omitted, the default value 50 will be used

Metadata	TRUE FALSE	TRUE - Send recorded footage along with rendered recorded metadata if applicable FALSE - Send normal recorded footage If this parameter is omitted, the default value FALSE will be used										
RenderInfo	All Name Description Timestamp	Specifies the list of camera information to render on the image. In case this parameter is omitted, no camera information will be rendered. The fields must be separated by commas <div style="margin-top: 10px; border: 1px solid black; padding: 5px;"> <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th>Name</th><th>Description</th></tr> </thead> <tbody> <tr> <td>All</td><td>Render all available information</td></tr> <tr> <td>Name</td><td>Render camera name</td></tr> <tr> <td>Description</td><td>Render camera description</td></tr> <tr> <td>Timestamp</td><td>Render camera timestamp</td></tr> </tbody> </table> </div>	Name	Description	All	Render all available information	Name	Render camera name	Description	Render camera description	Timestamp	Render camera timestamp
Name	Description											
All	Render all available information											
Name	Render camera name											
Description	Render camera description											
Timestamp	Render camera timestamp											
WatermarkText	String	Specifies a text that will be displayed on top of the image. If this parameter is omitted, no watermark text will be rendered on the image										
WatermarkColor	APIColor	Watermark text color. If this parameter is omitted, the default value White will be used										
WatermarkSize	Integer. 8 >= X <= 64	Watermark text font size. If this parameter is omitted, the default value 26 will be used										
WatermarkPosition	TopLeft TopRight Center BottomLeft BottomRight	Watermark text position. If this parameter is omitted, the default value BottomRight will be used										
Push	TRUE FALSE	TRUE - Send recorded footage as fast as possible, with no playback speed control (Use when downloading recordings). FALSE - Send recorded footage with playback speed control of 1x If this parameter is omitted, the default value FALSE will be used										
SourceStorage	HOT COLD EDGE	This parameter specifies the source storage used to retrieve the images. <div style="margin-top: 10px; border: 1px solid black; padding: 5px;"> <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th>Name</th><th>Description</th></tr> </thead> <tbody> <tr> <td>HOT</td><td>Hot storage is the main storage of the system, where all recordings are stored</td></tr> <tr> <td>COLD</td><td>Cold storage refers to archived recordings by the Archiving system</td></tr> <tr> <td>EDGE</td><td>Edge storage refers to the storage of recordings inside the device</td></tr> </tbody> </table> </div> If this parameter is omitted, the default value HOT will be used.	Name	Description	HOT	Hot storage is the main storage of the system, where all recordings are stored	COLD	Cold storage refers to archived recordings by the Archiving system	EDGE	Edge storage refers to the storage of recordings inside the device		
Name	Description											
HOT	Hot storage is the main storage of the system, where all recordings are stored											
COLD	Cold storage refers to archived recordings by the Archiving system											
EDGE	Edge storage refers to the storage of recordings inside the device											

Audio	TRUE FALSE	Request synchronized audio with video. If this parameter is omitted, the default value FALSE will be used.										
AudioFormat	PCM WAV AAC AAC-MP4	Audio output format. If this parameter is omitted, the default value PCM will be used. <table border="1"> <thead> <tr> <th>Name</th><th>Description</th></tr> </thead> <tbody> <tr> <td>PCM</td><td>Pure PCM audio</td></tr> <tr> <td>WAV</td><td>PCM audio with WAV header</td></tr> <tr> <td>AAC</td><td>AAC with ADTS header</td></tr> <tr> <td>AAC-MP4</td><td>AAC with MP4 config header</td></tr> </tbody> </table>	Name	Description	PCM	Pure PCM audio	WAV	PCM audio with WAV header	AAC	AAC with ADTS header	AAC-MP4	AAC with MP4 config header
Name	Description											
PCM	Pure PCM audio											
WAV	PCM audio with WAV header											
AAC	AAC with ADTS header											
AAC-MP4	AAC with MP4 config header											

* Mandatory parameters

Example 1: Request recordings of camera "Camera1" from January 01, 2014 10:00:00AM to January 02, 2014 09:00:00AM with error response in XML

```
http://192.168.0.1:8601/Interface/Cameras/Playback/GetJPEGStream?  
Camera=Camera1&StartDate=2014.01.01&StartTime=10.00.00.000&  
EndDate=2014.01.02&EndTime=09.00.00.000&ResponseFormat=XML
```

Example 2: Request recordings of camera "Camera1" from January 01, 2014 to date, without playback speed control and error response in text

```
http://192.168.0.1:8601/Interface/Cameras/Playback/GetJPEGStream?  
Camera=Camera1&StartDate=2014.01.01&StartTime=00.00.00.000&  
Push=TRUE&ResponseFormat=Text
```

Example 3: Request recordings of camera "Camera1" from January 25, 2014 to date, with playback speed control, custom size of 352x240 and compression quality of 80

```
http://192.168.0.1:8601/Interface/Cameras/Playback/GetJPEGStream?  
Camera=Camera1&StartDate=2014.01.20&StartTime=00.00.00.000&  
Width=352&Height=240&Quality=80
```

Example 4: Request recordings of camera "Camera1" from January 25, 2014 to date, with playback speed control, custom size of 352x240 and compression quality of 80 from cold storage

```
http://192.168.0.1:8601/Interface/Cameras/Playback/GetJPEGStream?  
Camera=Camera1&StartDate=2014.01.20&StartTime=00.00.00.000&  
Width=352&Height=240&Quality=80&SourceStorage=COLD
```

Example 5: Request recordings of camera "Camera1" from January 25, 2014 to date, with playback speed control, custom size of 352x240 with original proportions and compression quality of 80, analytics metadata, name and description rendered on the image, from cold storage

```
http://192.168.0.1:8601/Interface/Cameras/Playback/GetJPEGStream?  
Camera=Camera1&StartDate=2014.01.20&StartTime=00.00.00.000&  
Width=352&Height=240&KeepAspectRatio=TRUE&Quality=80&  
Metadata=TRUE&RenderInfo=Name, Description&SourceStorage=COLD
```

Response:

In case of success, a JPEG stream will be sent by using HTTP Multipart x-mixed-replace transmission. Each JPEG frame is separated by the multipart boundary --DigifortBoundary.

The main HTTP response header will contain the following custom headers:

Header	Valid values	Description
DGF-FrameCount	Integer	Specifies the amount of recorded frames that will be sent within the transmission

After the multipart boundary, a little HTTP header will be sent containing information about the frame:

Header	Valid values	Description
Content-Type	Media MIME	Identify the media type
Content-Length	Integer	Specify the size (In bytes) of the media frame
DGF-FrameNumber	Integer	Frame number
DGF-FrameDate	APIDate	Frame date in APIDate format This header will only be included if TimestampHeader parameter is TRUE
DGF-FrameTime	APITime	Frame time in APITime format This header will only be included if TimestampHeader parameter is TRUE
DGF-AACMP4Config	String	MP4 Setup Header in Base64 format This header will only be included if AudioFormat parameter is AAC-MP4
DGF-PCMChannels	Integer	Number of PCM channels This header will only be included if AudioFormat parameter is PCM
DGF-PCMSamplesPerSecond	Integer	Number of Samples per Second This header will only be included if AudioFormat parameter is PCM
DGF-PCMBitsPerSample	Integer	Number of Bits per Sample This header will only be included if AudioFormat parameter is PCM

In the case of error, a code and an error message will be returned. A error can be returned if the user does not have live viewing rights of the camera, if the camera is deactivated or if the camera was not found.

In case no recordings are found the connection will be gracefully closed and the HTTP custom header DGF-FrameCount will contain value 0

Example of JPEG media stream:

```

HTTP/1.1 200 OK
Connection: close
Content-Type: multipart/x-mixed-replace; boundary=--DigifortBoundary
DGF-FrameCount: 103531

--DigifortBoundary
Content-Type: image/jpeg
Content-Length: 35463
DGF-FrameNumber: 1
DGF-FrameDate: 2014.12.01
DGF-FrameTime: 10.00.00.755

JPEG_DATA
JPEG_DATA
..
..
..
JPEG_DATA
--DigifortBoundary
Content-Type: image/jpeg
Content-Length: 34236
DGF-FrameNumber: 2
DGF-FrameDate: 2014.12.01
DGF-FrameTime: 10.00.01.126

JPEG_DATA
JPEG_DATA
..
..
..
JPEG_DATA

```

HTTP Return: 200 OK

Parameters of return: [Default return of API](#)

4.3.11.4 Requesting a recorded media stream (MP4)

Requests a recorded media stream from the specified camera. The media stream will be sent in MP4 format. The system will return a Fragmented MP4 stream that can be directly displayed by any compatible web browser.

Warning: This command can overload the server CPU. This will occur because in order to deliver the live MP4 video stream the server may have to transcode the video that it is receiving from the camera.

Compatibility: All editions

Security level: Requires user authentication

Method: HTTP GET

Syntax

```
http://<server_address>/Interface/Cameras/Playback/GetMP4Stream?
```

<argument=value> [&<argument=value>...] [&<general argument>...]

Arguments:

Argument	Valid values	Description
Camera*	String	Camera name
StartDate*	APIDate	Start date of recordings on APIDate format
StartTime*	APITime	Start time of recordings on APITime format
EndDate	APIDate	End date of recordings on APIDate format If this parameter is omitted, the current date will be used
EndTime	APITime	End time of recordings on APITime format If this parameter is omitted, the current time will be used
Transcode	TRUE FALSE	Force transcoding of the source video to H.264 format. If this parameter is omitted, the default value FALSE will be used. Transcoding will be forced to TRUE if the source video is not in H.264 format or if parameters such as Metadata, RenderInfo, Watermark are activated or if the user watermark is forced by the user account. Using transcoding will put heavy load to the server
Width	Integer. X >= -1 && X != 0	Custom image width Specify value -1 to use the original recorded image width If this parameter is omitted, the default value -1 will be used
Height	Integer. X >= -1 && X != 0	Custom image height Specify value -1 to use the original recorded image height If this parameter is omitted, the default value -1 will be used
KeepAspectRatio	TRUE FALSE	Keep the aspect ratio of the original image. When Keep Aspect Ratio is used, the original image will be resized within the provided Width/Height values, but will also maintain the original proportional which means that either Width or Height could be less than the value provided If this parameter is omitted, the default value TRUE will be used
Quality	Integer. 1 >= X <= 100	H.264 compression quality value If this parameter is omitted, the default value of 70 will be used
Metadata	TRUE FALSE	TRUE - Send recorded footage along with rendered recorded metadata if applicable FALSE - Send normal recorded footage If this parameter is omitted, the default value FALSE will be used

RenderInfo	All Name Description Timestamp	<p>Specifies the list of camera information to render on the image. In case this parameter is omitted, no camera information will be rendered.</p> <p>The fields must be separated by commas</p> <table border="1"> <thead> <tr> <th>Name</th><th>Description</th></tr> </thead> <tbody> <tr> <td>All</td><td>Render all available information</td></tr> <tr> <td>Name</td><td>Render camera name</td></tr> <tr> <td>Description</td><td>Render camera description</td></tr> <tr> <td>Timestamp</td><td>Render camera timestamp</td></tr> </tbody> </table>	Name	Description	All	Render all available information	Name	Render camera name	Description	Render camera description	Timestamp	Render camera timestamp
Name	Description											
All	Render all available information											
Name	Render camera name											
Description	Render camera description											
Timestamp	Render camera timestamp											
WatermarkText	String	Specifies a text that will be displayed on top of the image. If this parameter is omitted, no watermark text will be rendered on the image										
WatermarkColor	APIColor	Watermark text color. If this parameter is omitted, the default value White will be used										
WatermarkSize	Integer. 8 >= X <= 64	Watermark text font size. If this parameter is omitted, the default value 26 will be used										
WatermarkPosition	TopLeft TopRight Center BottomLeft BottomRight	Watermark text position. If this parameter is omitted, the default value BottomRight will be used										
Push	TRUE FALSE	<p>TRUE - Send recorded footage as fast as possible, with no playback speed control (Use when downloading recordings).</p> <p>FALSE - Send recorded footage with playback speed control of 1x</p> <p>If this parameter is omitted, the default value FALSE will be used</p>										
SourceStorage	HOT COLD EDGE	<p>This parameter specifies the source storage used to retrieve the images.</p> <table border="1"> <thead> <tr> <th>Name</th><th>Description</th></tr> </thead> <tbody> <tr> <td>HOT</td><td>Hot storage is the main storage of the system, where all recordings are stored</td></tr> <tr> <td>COLD</td><td>Cold storage refers to archived recordings by the Archiving system</td></tr> <tr> <td>EDGE</td><td>Edge storage refers to the storage of recordings inside the device</td></tr> </tbody> </table> <p>If this parameter is omitted, the default value HOT will be used.</p>	Name	Description	HOT	Hot storage is the main storage of the system, where all recordings are stored	COLD	Cold storage refers to archived recordings by the Archiving system	EDGE	Edge storage refers to the storage of recordings inside the device		
Name	Description											
HOT	Hot storage is the main storage of the system, where all recordings are stored											
COLD	Cold storage refers to archived recordings by the Archiving system											
EDGE	Edge storage refers to the storage of recordings inside the device											
Audio	TRUE FALSE	<p>Request synchronized audio with video.</p> <p>If this parameter is omitted, the default value FALSE will be used.</p>										

* Mandatory parameters

Example 1: Request recordings of camera "Camera1" from January 01, 2014 10:00:00AM to January 02, 2014 09:00:00AM with error response in XML

```
http://192.168.0.1:8601/Interface/Cameras/Playback/GetMP4Stream?
Camera=Camera1&StartDate=2014.01.01&StartTime=10.00.00.000&
EndDate=2014.01.02&EndTime=09.00.00.000&ResponseFormat=XML
```

Example 2: Request recordings of camera "Camera1" from January 01, 2014 to date, without playback speed control and error response in text

```
http://192.168.0.1:8601/Interface/Cameras/Playback/GetMP4Stream?
Camera=Camera1&StartDate=2014.01.01&StartTime=00.00.00.000&
Push=TRUE&ResponseFormat=Text
```

Example 3: Request recordings of camera "Camera1" from January 25, 2014 to date, with playback speed control, transcoding with custom size of 352x240 and compression quality of 80

```
http://192.168.0.1:8601/Interface/Cameras/Playback/GetMP4Stream?
Camera=Camera1&StartDate=2014.01.20&StartTime=00.00.00.000&
Transcoding=TRUE&Width=352&Height=240&Quality=80
```

Example 4: Request recordings of camera "Camera1" from January 25, 2014 to date, with playback speed control, transcoding with custom size of 352x240 and compression quality of 80 from cold storage

```
http://192.168.0.1:8601/Interface/Cameras/Playback/GetMP4Stream?
Camera=Camera1&StartDate=2014.01.20&StartTime=00.00.00.000&
Transcoding=TRUE&Width=352&Height=240&Quality=80&SourceStorage=COLD
```

Example 5: Request recordings of camera "Camera1" from January 25, 2014 to date, with playback speed control, transcoding with custom size of 352x240 with original proportions and compression quality of 80, analytics metadata, name and description rendered on the image, from cold storage

```
http://192.168.0.1:8601/Interface/Cameras/Playback/GetMP4Stream?
Camera=Camera1&StartDate=2014.01.20&StartTime=00.00.00.000&
Transcoding=TRUE&Width=352&Height=240&KeepAspectRatio=TRUE&Quality=80&
Metadata=TRUE&RenderInfo=Name,Description&SourceStorage=COLD
```

Response:

In case of success, the MP4 stream will be sent

In the case of error, a code and an error message will be returned. A error can be returned if the user does not have live viewing rights of the camera, if the camera is deactivated or if the camera was not found.

In case no recordings are found the connection will be gracefully closed.

HTTP Return: 200 OK

Parameters of return: [Default return of API](#)

4.3.11.5 Requesting the timeline data

Requests the timeline data information on the recorded footage of the specified camera.

Compatibility: All editions

Security level: Requires user authentication

Method: HTTP GET

Syntax:

```
http://<server_address>/Interface/Cameras/Playback/GetTimelineData?
<argument=value>[&<argument=value>...] [&<general_argument>...]
```

Arguments:

Argument	Valid values	Description						
Camera*	String	Camera name						
StartDate*	APIDate	Start date of recordings on APIDate format						
StartTime*	APITime	Start time of recordings on APITime format						
EndDate	APIDate	End date of recordings on APIDate format If this parameter is omitted, the current date will be used						
EndTime	APITime	End time of recordings on APITime format If this parameter is omitted, the current time will be used						
Video	TRUE FALSE	Include timeline data for video track If this parameter is omitted, the default value TRUE will be used						
Audio	TRUE FALSE	Include timeline data for audio track If this parameter is omitted, the default value TRUE will be used						
Metadata	TRUE FALSE	Include timeline data for metadata track If this parameter is omitted, the default value TRUE will be used						
SourceStorage	HOT COLD	This parameter specifies the source storage used to retrieve the data. <table border="1" style="margin-left: 20px;"> <thead> <tr> <th>Name</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>HOT</td> <td>Hot storage is the main storage of the system, where all recordings are stored</td> </tr> <tr> <td>COLD</td> <td>Cold storage refers to archived recordings by the Archiving system</td> </tr> </tbody> </table> If this parameter is omitted, the default value HOT will be used.	Name	Description	HOT	Hot storage is the main storage of the system, where all recordings are stored	COLD	Cold storage refers to archived recordings by the Archiving system
Name	Description							
HOT	Hot storage is the main storage of the system, where all recordings are stored							
COLD	Cold storage refers to archived recordings by the Archiving system							

* Mandatory parameters

Example 1: Request timeline data of recordings from camera "Camera1" from January 01, 2014 10:00:00AM to January 02, 2014 09:00:00AM with error response in XML

```
http://192.168.0.1:8601/Interface/Cameras/Playback/GetTimelineData?
Camera=Camera1&StartDate=2014.01.01&StartTime=10.00.00.000&
EndDate=2014.01.02&EndTime=09.00.00.000&ResponseFormat=XML
```

Example 2: Request timeline data of recordings from "Camera1" from January 01, 2014 to date, without metadata track and response in text

```
http://192.168.0.1:8601/Interface/Cameras/Playback/GetTimelineData?
Camera=Camera1&StartDate=2014.01.01&StartTime=00.00.00.000&Metadata=FALSE&
ResponseFormat=Text
```

Response:

In case of success, the timeline data will be sent by using HTTP Multipart x-mixed-replace transmission. Each block of timeline data is separated by the multipart boundary -- DigiFortBoundary.

Multipart transmission is used because the process of reading the timeline could be lengthy. Small chunks of data will be sent on every multipart boundary.

In the case of error, a code and an error message will be returned. An error can be returned if the user does not have live viewing rights of the camera, if the camera is deactivated or if the camera was not found.

In case no recordings are found the connection will be gracefully closed.

Each block of timeline data delimited by a multipart boundary will have the following fields:

Parameter	Type	Description																		
MEDIAKIND	VIDEO AUDIO METADATA	Identifies the media kind <table border="1" style="margin-left: 20px;"> <thead> <tr> <th>Name</th><th>Description</th></tr> </thead> <tbody> <tr> <td>VIDEO</td><td>Timeline data from video track</td></tr> <tr> <td>AUDIO</td><td>Timeline data from audio track</td></tr> <tr> <td>METADATA</td><td>Timeline data from metadata track</td></tr> </tbody> </table>	Name	Description	VIDEO	Timeline data from video track	AUDIO	Timeline data from audio track	METADATA	Timeline data from metadata track										
Name	Description																			
VIDEO	Timeline data from video track																			
AUDIO	Timeline data from audio track																			
METADATA	Timeline data from metadata track																			
FRAMEKIND	VIDEO AUDIO ANALYTICS EVENTS MOTION_25 MOTION_50 MOTION_75 MOTION_100	Identifies the type of timeline frame <table border="1" style="margin-left: 20px;"> <thead> <tr> <th>Name</th><th>Description</th></tr> </thead> <tbody> <tr> <td>VIDEO</td><td>Every frame from video track will be VIDEO type</td></tr> <tr> <td>AUDIO</td><td>Every frame from audio track will be AUDIO type</td></tr> <tr> <td>ANALYTICS</td><td>Metadata track contains Analytics metadata</td></tr> <tr> <td>EVENTS</td><td>Metadata track is identifying that the recorded footage is from event</td></tr> <tr> <td>MOTION_25</td><td>25% motion was found on the recordings for this chunk of time</td></tr> <tr> <td>MOTION_50</td><td>50% motion was found on the recordings for this chunk of time</td></tr> <tr> <td>MOTION_75</td><td>75% motion was found on the recordings for this chunk of time</td></tr> <tr> <td>MOTION_100</td><td>100% motion was found on the recordings for this chunk of time</td></tr> </tbody> </table>	Name	Description	VIDEO	Every frame from video track will be VIDEO type	AUDIO	Every frame from audio track will be AUDIO type	ANALYTICS	Metadata track contains Analytics metadata	EVENTS	Metadata track is identifying that the recorded footage is from event	MOTION_25	25% motion was found on the recordings for this chunk of time	MOTION_50	50% motion was found on the recordings for this chunk of time	MOTION_75	75% motion was found on the recordings for this chunk of time	MOTION_100	100% motion was found on the recordings for this chunk of time
Name	Description																			
VIDEO	Every frame from video track will be VIDEO type																			
AUDIO	Every frame from audio track will be AUDIO type																			
ANALYTICS	Metadata track contains Analytics metadata																			
EVENTS	Metadata track is identifying that the recorded footage is from event																			
MOTION_25	25% motion was found on the recordings for this chunk of time																			
MOTION_50	50% motion was found on the recordings for this chunk of time																			
MOTION_75	75% motion was found on the recordings for this chunk of time																			
MOTION_100	100% motion was found on the recordings for this chunk of time																			
STARTTIME	APITimestamp	Start timestamp of the chunk of timeline data																		
ENDTIME	APITimestamp	End timestamp of the chunk of timeline data																		

Example of timeline data with response in text:

```

HTTP/1.1 200 OK
Connection: keep-alive
Content-Type: multipart/x-mixed-replace; boundary=---DigifortBoundary

--DigifortBoundary
Content-Type: text/plain; charset=UTF-8
Content-Length: 192

RESPONSE_CODE=0

```

```
RESPONSE_MESSAGE=OK
TIMELINEDATA_MEDIAKIND=VIDEO
TIMELINEDATA_FRAMETYPE=VIDEO
TIMELINEDATA_STARTTIME=2018-03-13 13:00:00.294
TIMELINEDATA_ENDTIME=2018-03-13 16:00:20.330
--DigifortBoundary
Content-Type: text/plain; charset=UTF-8
Content-Length: 199

RESPONSE_CODE=0
RESPONSE_MESSAGE=OK
TIMELINEDATA_MEDIAKIND=METADATA
TIMELINEDATA_FRAMETYPE=MOTION_50
TIMELINEDATA_STARTTIME=2018-03-13 13:00:00.025
TIMELINEDATA_ENDTIME=2018-03-13 13:00:03.024
```

Example of timeline data with response in XML (XML indentation was added to this example, the real data does not have indentation)

```
HTTP/1.1 200 OK
Connection: close
Content-Type: multipart/x-mixed-replace; boundary=--DigifortBoundary

--DigifortBoundary
Content-Type: text/xml; charset=UTF-8
Content-Length: 281

<?xml version="1.0" encoding="UTF-8" ?>
<Response>
  <Code>0</Code>
  <Message>OK</Message>
  <Data>
    <TimelineData>
      <MediaKind>VIDEO</MediaKind>
      <FrameType>VIDEO</FrameType>
      <StartTime>2018-03-13 13:00:00.294</StartTime>
      <EndTime>2018-03-13 15:58:31.062</EndTime>
    </TimelineData>
  </Data>
</Response>
--DigifortBoundary
Content-Type: text/xml; charset=UTF-8
Content-Length: 288

<?xml version="1.0" encoding="UTF-8" ?>
<Response>
  <Code>0</Code>
  <Message>OK</Message>
  <Data>
    <TimelineData>
      <MediaKind>METADATA</MediaKind>
      <FrameType>MOTION_50</FrameType>
      <StartTime>2018-03-13 13:00:00.025</StartTime>
```

```

<EndTime>2018-03-13 13:00:03.024</EndTime>
</TimelineData>
</Data>
</Response>
```

Example of timeline data with response in JSON (JSON indentation was added to this example, the real data does not have indentation)

```

HTTP/1.1 200 OK
Connection: close
Content-Type: multipart/x-mixed-replace; boundary=---DigifortBoundary

--DigifortBoundary
Content-Type: application/json; charset=UTF-8
Content-Length: 281

{
    "Code": 0,
    "Message": "OK",
    "Data": {
        "TimelineData": {
            "MediaKind": "VIDEO",
            "FrameType": "VIDEO",
            "StartTime": "2018-03-13T13:00:00.294Z",
            "EndTime": "2018-03-13T15:58:31.062Z"
        }
    }
}
--DigifortBoundary
Content-Type: application/json; charset=UTF-8
Content-Length: 288

{
    "Code": 0,
    "Message": "OK",
    "Data": {
        "TimelineData": {
            "MediaKind": "METADATA",
            "FrameType": "MOTION_50",
            "StartTime": "2018-03-13T13:00:00.025Z",
            "EndTime": "2018-03-13T13:00:03.024Z"
        }
    }
}
```

HTTP Return: 200 OK

Parameters of return: [Default return of API](#)

4.3.11.6 Exporting

This section describes the procedure and commands to export media in .mp4 file format

Since the exporting process can take several minutes to finish depending on the input settings, the process was split in 3 steps:

1. To start exporting, you must first call the command [StartExport](#), providing all the information for exporting. This command will let you keep track of the progress of exporting.
2. Once exporting has finished, the exported files will be held in a temporary folder in the server and you must download the files using [GetExportedFile](#) command
3. After all files were downloaded you must call [CloseExport](#) routine in order to notify the server to delete the temporary files. If this command is not called after an exporting, the server might keep a huge amount of temporary exporting files and may run out of disk space. Any undeleted temporary files are going to be deleted during next server startup.

4.3.11.6.1 Starting an export session

Start the exporting of media in MP4 file format

The exporting process could take several minutes to finish depending on the options selected. During the export process you must keep the HTTP connection open. If the connection is closed during exporting, the process will be canceled and all temporary exported data deleted.

Compatibility: All editions

Security level: Requires user authentication

Method: HTTP GET

Syntax:

```
http://<server_address>/Interface/Cameras/Playback/StartExport?
<argument=value>[&<argument=value>...][&<general_argument>...]
```

Arguments:

Argument	Valid values	Description
Camera*	String	Camera name
StartDate*	APIDate	Start date of recordings on APIDate format
StartTime*	APITime	Start time of recordings on APITime format
EndDate	APIDate	End date of recordings on APIDate format If this parameter is omitted, the current date will be used
EndTime	APITime	End time of recordings on APITime format If this parameter is omitted, the current time will be used
Transcode	TRUE FALSE	If Transcoding is activated the system will convert the source video to H264 format using Width, Height and Quality. Audio will not be transcoded. If Transcoding is deactivated (Default) the system will export the original media to .mp4 container. Depending on the original media format, some players may not be able to reproduce the exported file. Using transcoding will put heavy load to the server and will make the process of exporting much slower.
Width	Integer. X >= -1 && X != 0	Custom image width to be used when Transcode=TRUE Specify value -1 to use the original recorded image width

		If this parameter is omitted, the default value -1 will be used										
Height	Integer. X >= -1 && X != 0	Custom image height to be used when Transcode=TRUE Specify value -1 to use the original recorded image height If this parameter is omitted, the default value -1 will be used										
KeepAspectRatio	TRUE FALSE	Keep the aspect ratio of the original image when Transcode=TRUE. When Keep Aspect Ratio is used, the original image will be resized within the provided Width/Height values, but will also maintain the original proportional which means that either Width or Height could be less than the value provided If this parameter is omitted, the default value FALSE will be used										
Quality	Integer. 0 >= X <= 100	Transcoding quality to be used when Transcode=TRUE If this parameter is omitted, the default value 70 will be used										
Metadata	TRUE FALSE	TRUE - Send recorded footage along with rendered recorded metadata if applicable FALSE - Send normal recorded footage If this parameter is omitted, the default value FALSE will be used										
RenderInfo	All Name Description Timestamp	Specifies the list of camera information to render on the image. In case this parameter is omitted, no camera information will be rendered. The fields must be separated by commas <div style="margin-top: 10px; border: 1px solid black; padding: 5px;"> <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="background-color: #0070C0; color: white;">Name</th> <th style="background-color: #0070C0; color: white;">Description</th> </tr> </thead> <tbody> <tr> <td>All</td> <td>Render all available information</td> </tr> <tr> <td>Name</td> <td>Render camera name</td> </tr> <tr> <td>Description</td> <td>Render camera description</td> </tr> <tr> <td>Timestamp</td> <td>Render camera timestamp</td> </tr> </tbody> </table> </div>	Name	Description	All	Render all available information	Name	Render camera name	Description	Render camera description	Timestamp	Render camera timestamp
Name	Description											
All	Render all available information											
Name	Render camera name											
Description	Render camera description											
Timestamp	Render camera timestamp											
WatermarkText	String	Specifies a text that will be displayed on top of the image. If this parameter is omitted, no watermark text will be rendered on the image										
WatermarkColor	APIColor	Watermark text color. If this parameter is omitted, the default value White will be used										
WatermarkSize	Integer. 8 >= X <= 64	Watermark text font size. If this parameter is omitted, the default value 26 will be used										
WatermarkPosition	TopLeft TopRight Center BottomLeft BottomRight	Watermark text position. If this parameter is omitted, the default value BottomRight will be used										
Audio	TRUE FALSE	Activate audio exporting If this parameter is omitted, the default value FALSE will be										

		used						
SourceStorage	HOT COLD	<p>This parameter specifies the source storage used to retrieve the images.</p> <table border="1" style="margin-left: auto; margin-right: auto;"> <thead> <tr> <th>Name</th><th>Description</th></tr> </thead> <tbody> <tr> <td>HOT</td><td>Hot storage is the main storage of the system, where all recordings are stored</td></tr> <tr> <td>COLD</td><td>Cold storage refers to archived recordings by the Archiving system</td></tr> </tbody> </table> <p>If this parameter is omitted, the default value HOT will be used.</p>	Name	Description	HOT	Hot storage is the main storage of the system, where all recordings are stored	COLD	Cold storage refers to archived recordings by the Archiving system
Name	Description							
HOT	Hot storage is the main storage of the system, where all recordings are stored							
COLD	Cold storage refers to archived recordings by the Archiving system							
Zip	TRUE FALSE	The exported media might be split into multiple .mp4 files depending on several factors. By using the Zip parameter the system can pack all exported files into a single uncompressed Zip file for convenience.						
Progress	TRUE FALSE	<p>The process of exporting the media file could take several minutes to complete, with this option the system will send the progress in multi-part response format. You can use the progress data to provide some information to the user.</p> <p>If Progress is not selected, the server will only send a reply to the HTTP request once the exporting process has finished.</p>						
FilenameFormat	ID TIMESTAMP	<p>This parameter specifies the format of the exported file name.</p> <table border="1" style="margin-left: auto; margin-right: auto;"> <thead> <tr> <th>Name</th><th>Description</th></tr> </thead> <tbody> <tr> <td>ID</td><td>Filename will be composed of SessionID + File number</td></tr> <tr> <td>TIMESTAMP</td><td>Filename will contain the timestamps of the first and last exported frame inside the file</td></tr> </tbody> </table> <p>If this parameter is omitted, the default value ID will be used.</p>	Name	Description	ID	Filename will be composed of SessionID + File number	TIMESTAMP	Filename will contain the timestamps of the first and last exported frame inside the file
Name	Description							
ID	Filename will be composed of SessionID + File number							
TIMESTAMP	Filename will contain the timestamps of the first and last exported frame inside the file							

* Mandatory parameters

Example 1: Export media from "Camera1" from January 01, 2014 10:00:00AM to January 01, 2014 11:00:00AM with response in XML

```
http://192.168.0.1:8601/Interface/Cameras/Playback/StartExport?
Camera=Camera1&StartDate=2014.01.01&StartTime=10.00.00.000&
EndDate=2014.01.01&EndTime=11.00.00.000&ResponseFormat=XML
```

Example 2: Export media from "Camera1" from January 01, 2014 10:00:00AM to January 01, 2014 11:00:00AM with transcoding activated and quality 70, analytics metadata, name and description rendered on the image with response in Text

```
http://192.168.0.1:8601/Interface/Cameras/Playback/StartExport?
Camera=Camera1&StartDate=2014.01.01&StartTime=10.00.00.000&
EndDate=2014.01.01&EndTime=11.00.00.000&Transcode=TRUE&
Quality=70&Metadata=TRUE&RenderInfo=Name,Description&ResponseFormat=Text
```

Response:

In case of success, a new export session will be created.

If Progress is activated, a stream of data will be sent by using HTTP Multipart `x-mixed-replace` transmission. Each boundary is split by multipart boundary `--DigifortBoundary`. When export finishes, the last boundary will contain a list of all exported files

If Progress is deactivated, a list of all exported files will be sent after exporting has finished.

In case of an error, an error section will be sent in the multipart transmission if Progress is activated and if Progress is deactivated the system will just return the error code

Parameters in case of an error:

Parameter	Type	Description
CODE	String	Error code

Parameters for progress data:

Parameter	Type	Description
TOTALFRAMES	Integer64	Total number of frames in the exporting session
EXPORTEDFRAMES	Integer64	Number of frames already exported
PROGRESS	Integer	Progress percentage (0 to 100)

Parameters with result of exporting:

Parameter	Type	Description
ID	String	Session ID
FILES	List	List of exported files

Parameters for exported file list:

Fixed Parameters	Type	Description
COUNT	Integer	Total number of exported files

Parameter	Type	Description
FILENAME	String	Name of exported file
SIZE	Integer64	File size

Example of successful exported media in text with progress:

```
--DigifortBoundary
Content-Type: text/plain; charset=UTF-8
Content-Length: 85

RESPONSE_CODE=0
RESPONSE_MESSAGE=OK
TOTALFRAMES=911
EXPORTEDFRAMES=0
PROGRESS=0
--DigifortBoundary
Content-Type: text/plain; charset=UTF-8
Content-Length: 89

RESPONSE_CODE=0
RESPONSE_MESSAGE=OK
TOTALFRAMES=911
EXPORTEDFRAMES=911
PROGRESS=100
--DigifortBoundary
```

```
Content-Type: text/plain; charset=UTF-8
Content-Length: 166

RESPONSE_CODE=0
RESPONSE_MESSAGE=OK
ID=F09878EE6DA68CCE5C6F77EEC1CCD553
FILES_COUNT=1
FILE_1_FILENAME=F09878EE6DA68CCE5C6F77EEC1CCD553.zip
FILE_1_SIZE=14339668
```

Example of successful exported media in XML with progress (XML Indenting added to the example):

```
--DigifortBoundary
Content-Type: text/xml; charset=UTF-8
Content-Length: 215

<?xml version="1.0" encoding="UTF-8" ?>
<Response>
  <Code>0</Code>
  <Message>OK</Message>
  <Data>
    <Progress>
      <TotalFrames>911</TotalFrames>
      <ExportedFrames>0</ExportedFrames>
      <Progress>0</Progress>
    </Progress>
  </Data>
</Response>
--DigifortBoundary
Content-Type: text/xml; charset=UTF-8
Content-Length: 218

<?xml version="1.0" encoding="UTF-8" ?>
<Response>
  <Code>0</Code>
  <Message>OK</Message>
  <Data>
    <Progress>
      <TotalFrames>911</TotalFrames>
      <ExportedFrames>876</ExportedFrames>
      <Progress>96</Progress>
    </Progress>
  </Data>
</Response>--DigifortBoundary
Content-Type: text/xml; charset=UTF-8
Content-Length: 219

<?xml version="1.0" encoding="UTF-8" ?>
<Response>
  <Code>0</Code>
  <Message>OK</Message>
  <Data>
    <Progress>
```

```

<TotalFrames>911</TotalFrames>
<ExportedFrames>911</ExportedFrames>
<Progress>100</Progress>
</Progress>
</Data>
</Response>
--DigifortBoundary
Content-Type: text/xml; charset=UTF-8
Content-Length: 288

<?xml version="1.0" encoding="UTF-8" ?>
<Response>
<Code>0</Code>
<Message>OK</Message>
<Data>
<Export>
<ID>D0CC9293C1565D6F21162A5A05CB606F</ID>
<Files>
<Count>1</Count>
<File>
<Filename>D0CC9293C1565D6F21162A5A05CB606F.zip</Filename>
<Size>14339668</Size>
</File>
</Files>
</Export>
</Data>
</Response>

```

Example of successful exported media in JSON with progress (JSON Indenting added to the example):

```

--DigifortBoundary
Content-Type: application/json; charset=UTF-8
Content-Length: 215

{
    "Code": 0,
    "Message": "OK",
    "Data": {
        "Progress": {
            "TotalFrames": 911,
            "ExportedFrames": 0,
            "Progress": 0
        }
    }
}
--DigifortBoundary
Content-Type: application/json; charset=UTF-8
Content-Length: 218

{
    "Code": 0,
    "Message": "OK",
    "Data": {

```

```
"Progress": {
    "TotalFrames": 911,
    "ExportedFrames": 876,
    "Progress": 96
}
}
--DigifortBoundary
Content-Type: application/json; charset=UTF-8
Content-Length: 219

{
    "Code": 0,
    "Message": OK,
    "Data": {
        "Progress": {
            "TotalFrames": 911,
            "ExportedFrames": 911,
            "Progress": 100
        }
    }
}
--DigifortBoundary
Content-Type: application/json; charset=UTF-8
Content-Length: 288

{
    "Code": 0,
    "Message": "OK",
    "Data": {
        "Export": {
            "ID": "D0CC9293C1565D6F21162A5A05CB606F",
            "Files": [
                {
                    "Filename": "D0CC9293C1565D6F21162A5A05CB606F.zip",
                    "Size": 14339668
                }
            ]
        }
    }
}
```

4.3.11.6.2 Dow nloading exported files

Download the temporary exported files

Compatibility: All editions

Security level: Requires user authentication

Method: HTTP GET

Syntax:

```
http://<server_address>/Interface/Cameras/Playback/GetExportedFile?
<argument=value>[&<argument=value>...][&<general argument>...]
```

Arguments:

Argument	Valid values	Description
SessionID*	String	ID of the exporting session
Filename*	String	Name of the exported file

* Mandatory parameters

Example 1: Download exported file with error response in XML

```
http://192.168.0.1:8601/Interface/Cameras/Playback/GetExportedFile?
SessionID=C0273FC6E78D8D5B710EA6DE65CF70D6&
Filename=C0273FC6E78D8D5B710EA6DE65CF70D6_1.mp4&ResponseFormat=XML
```

Example 2: Download exported file with error response in Text

```
http://192.168.0.1:8601/Interface/Cameras/Playback/GetExportedFile?
SessionID=C0273FC6E78D8D5B710EA6DE65CF70D6&
Filename=C0273FC6E78D8D5B710EA6DE65CF70D6_1.mp4&ResponseFormat=Text
```

Response:

In case of success, the exported file will be returned. In case of error, a code and an error message will be returned. An error can be returned if the file or session is not found.

HTTP Return: 200 OK

Parameters of return: [Default return of API](#)

4.3.11.6.3 Closing an export session

Close an export session

This command must be called for all successful exporting sessions.

Compatibility: All editions

Security level: Requires user authentication

Method: HTTP GET

Syntax:

```
http://<server_address>/Interface/Cameras/Playback/CloseExport?
<argument=value>[&<argument=value>...][&<general argument>...]
```

Arguments:

Argument	Valid values	Description
SessionID*	String	ID of the exporting session

* Mandatory parameters

Example 1: Close an exporting session with response in Text

```
http://192.168.0.1:8601/Interface/Cameras/Playback/CloseExport?
SessionID=F09878EE6DA68CCE5C6F77EEC1CCD553&ResponseFormat=Text
```

Example 2: Close an exporting session with response in XML

```
http://192.168.0.1:8601/Interface/Cameras/Playback/CloseExport?
SessionID=F09878EE6DA68CCE5C6F77EEC1CCD553&ResponseFormat=XML
```

Response:

Default response of API.

HTTP Return: 200 OK

Parameters of return: [Default return of API](#)

4.3.12 PTZ

Commands for PTZ control and auxiliary commands

Not all cameras will support all of the commands supported by the system. To know which groups of commands a given camera supports, use the [Supported Commands](#) command

4.3.12.1 Simple

Simple PTZ allows the control of movable cameras in a simple way, with only a command call.

Compatibility: All editions

Security level: Requires user authentication

Method: HTTP GET

Syntax:

```
http://<server_address>/Interface/Cameras/PTZ/Simple?<argument=value>
[&<argument=value>...] [&<general\_argument>...]
```

Arguments:

Argument	Valid values	Description																										
Camera*	String	Name of the camera																										
Operation*	MoveLeft MoveRight MoveUp MoveDown MoveUpLeft MoveUpRight MoveDownLeft MoveDownRight Home ZoomTele ZoomWide FocusNear FocusFar IrisOpen IrisClose	Control operation to be executed <table border="1" data-bbox="734 1372 1419 1921"> <thead> <tr> <th>Operation</th><th>Description</th></tr> </thead> <tbody> <tr> <td>MoveLeft</td><td>Moves the camera to the left</td></tr> <tr> <td>MoveRight</td><td>Moves the camera to the right</td></tr> <tr> <td>MoveUp</td><td>Moves the camera upwards</td></tr> <tr> <td>MoveDown</td><td>Moves the camera downwards</td></tr> <tr> <td>MoveUpLeft</td><td>Moves the camera upwards and to the left simultaneously</td></tr> <tr> <td>MoveUpRight</td><td>Moves the camera upwards and to the right simultaneously</td></tr> <tr> <td>MoveDownLeft</td><td>Moves the camera downwards and to the left simultaneously</td></tr> <tr> <td>MoveDownRight</td><td>Moves the camera downwards and to the right simultaneously</td></tr> <tr> <td>Home</td><td>Calls the Home position</td></tr> <tr> <td>ZoomTele</td><td>Zoom in</td></tr> <tr> <td>ZoomWide</td><td>Zoom out</td></tr> <tr> <td>FocusNear</td><td>Focuses on objects near the camera</td></tr> </tbody> </table>	Operation	Description	MoveLeft	Moves the camera to the left	MoveRight	Moves the camera to the right	MoveUp	Moves the camera upwards	MoveDown	Moves the camera downwards	MoveUpLeft	Moves the camera upwards and to the left simultaneously	MoveUpRight	Moves the camera upwards and to the right simultaneously	MoveDownLeft	Moves the camera downwards and to the left simultaneously	MoveDownRight	Moves the camera downwards and to the right simultaneously	Home	Calls the Home position	ZoomTele	Zoom in	ZoomWide	Zoom out	FocusNear	Focuses on objects near the camera
Operation	Description																											
MoveLeft	Moves the camera to the left																											
MoveRight	Moves the camera to the right																											
MoveUp	Moves the camera upwards																											
MoveDown	Moves the camera downwards																											
MoveUpLeft	Moves the camera upwards and to the left simultaneously																											
MoveUpRight	Moves the camera upwards and to the right simultaneously																											
MoveDownLeft	Moves the camera downwards and to the left simultaneously																											
MoveDownRight	Moves the camera downwards and to the right simultaneously																											
Home	Calls the Home position																											
ZoomTele	Zoom in																											
ZoomWide	Zoom out																											
FocusNear	Focuses on objects near the camera																											

		FocusFar	Focuses on objects far from the camera
		IrisOpen	Opens Iris
		IrisClose	Closes Iris
Speed	Integer. 0 >= X <= 100	Speed of operation. If this parameter is omitted, the default value of 100 will be used	
VCam	Integer. X >= -1	Virtual Camera ID (For fisheye cameras with Multi-View directly from camera)	

* Mandatory parameters

Note: Some commands/parameters may not work with some cameras due to limitations of the communication interface itself developed by the camera's manufacturer. The basic parameters of Up, Down, Left, Right, Zoom In and Zoom Out should work for all cameras, whereas the parameters of Simultaneous Movement, Home Position, Focus and Iris may vary from one manufacturer to another.

Example 1: Moves a camera to the left at a speed of 50

```
http://192.168.0.1:8601/Interface/Cameras/PTZ/Simple?Camera=Camera1&
Operation=MoveRight&Speed=50
```

Example 2: Moves a camera downwards at a speed of 100 and response in text

```
http://192.168.0.1:8601/Interface/Cameras/PTZ/Simple?Camera=Camera1&
Operation=MoveDown&Speed=100&ResponseFormat=Text
```

Example 3: Execute zoom in a camera with response in XML and authentication of the user User1

```
http://192.168.0.1:8601/Interface/Cameras/PTZ/Simple?Camera=Camera1&
Operation=ZoomTele&ResponseFormat=XML&AuthUser=User1&AuthPass=User1Pass
```

Response:

Default response of API. In case the "Simple" command group is not supported by the camera's driver (due to restrictions in the communication interface developed by the camera's manufacturer), the error code 10105 (Command not supported by the PTZ driver) will be returned.

HTTP Return HTTP: 200 OK

Parameters of return: [Default return of API](#)

4.3.12.2 Relative

By way of relative movement control, you will be able to control the movement of the camera in degrees starting from the present position.

Compatibility: All editions

Security level: Requires authentication of the user

Method: HTTP GET

Syntax:

```
http://<server_address>/Interface/Cameras/PTZ/Relative?<argument=value>
[&<argument=value>...] [&<general\_argument>...]
```

Arguments:

Argument	Valid values	Description
----------	--------------	-------------

Camera*	String	Name of the camera												
Operation*	Pan Tilt Zoom Focus Iris	<p>Control operation to be executed</p> <table border="1"> <thead> <tr> <th>Operation</th><th>Description</th></tr> </thead> <tbody> <tr> <td>Pan</td><td>Relative operation of Pan (leftwards and rightwards movement)</td></tr> <tr> <td>Tilt</td><td>Relative operation of Tilt (Upwards and Downwards movement)</td></tr> <tr> <td>Zoom</td><td>Relative operation of Zoom</td></tr> <tr> <td>Focus</td><td>Relative operation of Focus</td></tr> <tr> <td>Iris</td><td>Relative operation of Iris</td></tr> </tbody> </table>	Operation	Description	Pan	Relative operation of Pan (leftwards and rightwards movement)	Tilt	Relative operation of Tilt (Upwards and Downwards movement)	Zoom	Relative operation of Zoom	Focus	Relative operation of Focus	Iris	Relative operation of Iris
Operation	Description													
Pan	Relative operation of Pan (leftwards and rightwards movement)													
Tilt	Relative operation of Tilt (Upwards and Downwards movement)													
Zoom	Relative operation of Zoom													
Focus	Relative operation of Focus													
Iris	Relative operation of Iris													
Value*	Integer. Consult the table of valid values	<p>Value of the operation in degrees for Pan/Tilt and steps for Zoom, Focus and Iris</p> <table border="1"> <thead> <tr> <th>Operation</th><th>Values of the operation</th></tr> </thead> <tbody> <tr> <td>Pan</td><td>-360 to 360 Negative values = Left Positive values = Right</td></tr> <tr> <td>Tilt</td><td>-360 to 360 Negative value = Up Positive values = Down</td></tr> <tr> <td>Zoom</td><td>-100 to 100 Negative values = Zoom Out Positive values = Zoom In</td></tr> <tr> <td>Focus</td><td>-100 to 100 Negative values = Focus Near Positive values = Focus Far</td></tr> <tr> <td>Iris</td><td>-100 a 100 Negative values = Close Iris Positive values = Open Iris</td></tr> </tbody> </table>	Operation	Values of the operation	Pan	-360 to 360 Negative values = Left Positive values = Right	Tilt	-360 to 360 Negative value = Up Positive values = Down	Zoom	-100 to 100 Negative values = Zoom Out Positive values = Zoom In	Focus	-100 to 100 Negative values = Focus Near Positive values = Focus Far	Iris	-100 a 100 Negative values = Close Iris Positive values = Open Iris
Operation	Values of the operation													
Pan	-360 to 360 Negative values = Left Positive values = Right													
Tilt	-360 to 360 Negative value = Up Positive values = Down													
Zoom	-100 to 100 Negative values = Zoom Out Positive values = Zoom In													
Focus	-100 to 100 Negative values = Focus Near Positive values = Focus Far													
Iris	-100 a 100 Negative values = Close Iris Positive values = Open Iris													
Speed	Integer. 0 >= X <= 100	Speed of operation. If this parameter is omitted, the default value of 100 will be used												
VCam	Integer. X >= -1	Virtual Camera ID (For fisheye cameras with Multi-View directly from camera)												

* Mandatory parameters

Note: Some commands/parameters may not work with some cameras due to limitations of the communication interface itself developed by the camera's manufacturer.

Example 1: Moves a camera 40 degrees to the right at a speed of 50

```
http://192.168.0.1:8601/Interface/Cameras/PTZ/Relative?Camera=Camera1&
Operation=Pan&Value=40&Speed=50
```

Example 2: Moves a camera downwards 10 degrees at a speed of 100 and response in text

```
http://192.168.0.1:8601/Interface/Cameras/PTZ/Relative?Camera=Camera1&Operation=Tilt&Value=10&Speed=100&ResponseFormat=Text
```

Example 3: Executes 20 steps of zoom in a camera with response in XML and authentication of the user User1

```
http://192.168.0.1:8601/Interface/Cameras/PTZ/Relative?Camera=Camera1&Operation=Zoom&Value=20&ResponseFormat=XML&AuthUser=User1&AuthPass=User1Pass
```

Response:

Default response of API. In case the "Relative" command group is not supported by the camera's driver (due to restrictions in the communication interface developed by the camera's manufacturer), the error code 10105 (Command not supported by the PTZ driver) will be returned.

HTTP Return HTTP: 200 OK

Parameters of return: [Default return of API](#)

4.3.12.3 Absolute

By way of absolute movement control, you will be able to position the camera in given coordinates in degrees.

Compatibility: All editions

Security level: Requires user authentication

Method: HTTP GET

Syntax:

```
http://<server_address>/Interface/Cameras/PTZ/Absolute?<argument=value>
[&<argument=value>...] [&<general argument>...]
```

Arguments:

Argument	Valid values	Description												
Camera*	String	Name of the camera												
Operation*	Pan Tilt Zoom Focus Iris	Control operation to be executed <table border="1" data-bbox="747 1425 1428 1700"> <thead> <tr> <th>Operation</th><th>Description</th></tr> </thead> <tbody> <tr> <td>Pan</td><td>Absolute operation of Pan (Leftwards and rightwards movement)</td></tr> <tr> <td>Tilt</td><td>Absolute operation of Tilt (Upwards and downwards movement)</td></tr> <tr> <td>Zoom</td><td>Absolute operation of Zoom</td></tr> <tr> <td>Focus</td><td>Absolute operation of Focus</td></tr> <tr> <td>Iris</td><td>Absolute operation of Iris</td></tr> </tbody> </table>	Operation	Description	Pan	Absolute operation of Pan (Leftwards and rightwards movement)	Tilt	Absolute operation of Tilt (Upwards and downwards movement)	Zoom	Absolute operation of Zoom	Focus	Absolute operation of Focus	Iris	Absolute operation of Iris
Operation	Description													
Pan	Absolute operation of Pan (Leftwards and rightwards movement)													
Tilt	Absolute operation of Tilt (Upwards and downwards movement)													
Zoom	Absolute operation of Zoom													
Focus	Absolute operation of Focus													
Iris	Absolute operation of Iris													
Value*	Integer. Consult the table of valid values	Value of the operation in degrees for Pan/Tilt and steps for Zoom, Focus and Iris <table border="1" data-bbox="747 1826 1428 1928"> <thead> <tr> <th>Operation</th><th>Values of the operation</th></tr> </thead> <tbody> <tr> <td>Pan</td><td>-180 to 180</td></tr> </tbody> </table>	Operation	Values of the operation	Pan	-180 to 180								
Operation	Values of the operation													
Pan	-180 to 180													

			Negative values = Left Positive values = Right
	Tilt		-180 to 180 Negative values = Up Positive values = Down
	Zoom		1 to 100
	Focus		1 to 100
	Iris		1 to 100
Speed	Integer. 0 >= X <= 100		Speed of operation. If this parameter is omitted, the default value of 100 will be used
VCam	Integer. X >= -1		Virtual Camera ID (For fisheye cameras with Multi-View directly from camera)

* Mandatory parameters

Note: Some commands/parameters may not work with some cameras due to limitations of the communication interface itself developed by the camera's manufacturer.

Example 1: Moves a camera to the Pan position of 40 degrees at a speed of 50

```
http://192.168.0.1:8601/Interface/Cameras/PTZ/Absolute?Camera=Camera1&
Operation=Pan&Value=40&Speed=50
```

Example 2: Moves a camera to the Tilt position of 10 degrees at a speed of 100 and response in text

```
http://192.168.0.1:8601/Interface/Cameras/PTZ/Absolute?Camera=Camera1&
Operation=Tilt&Value=10&Speed=100&ResponseFormat=Text
```

Example 3: Moves the lens of the camera to a position of 20 absolute steps of zoom with response in XML and authentication of the user User1

```
http://192.168.0.1:8601/Interface/Cameras/PTZ/Absolute?Camera=Camera1&
Operation=Zoom&Value=20&ResponseFormat=XML&AuthUser=User1&
AuthPass=User1Pass
```

Response:

Default response of API. In case the "Absolute" command group is not supported by the camera's driver (due to restrictions in the communication interface itself developed by the camera's manufacturer), the error code 10105 (Command not supported by the PTZ driver) will be returned.

HTTP Return HTTP: 200 OK

Parameters of return: [Default return of API](#)

4.3.12.4 Area Zoom

By way of area zoom coommand, you will be able to position the camera on a given image rectangle.

Compatibility: All editions

Security level: Requires user authentication

Method: HTTP GET

Syntax:

```
http://<server_address>/Interface/Cameras/PTZ/AreaZoom?<argument=value>
[&<argument=value>...] [&<general argument>...]
```

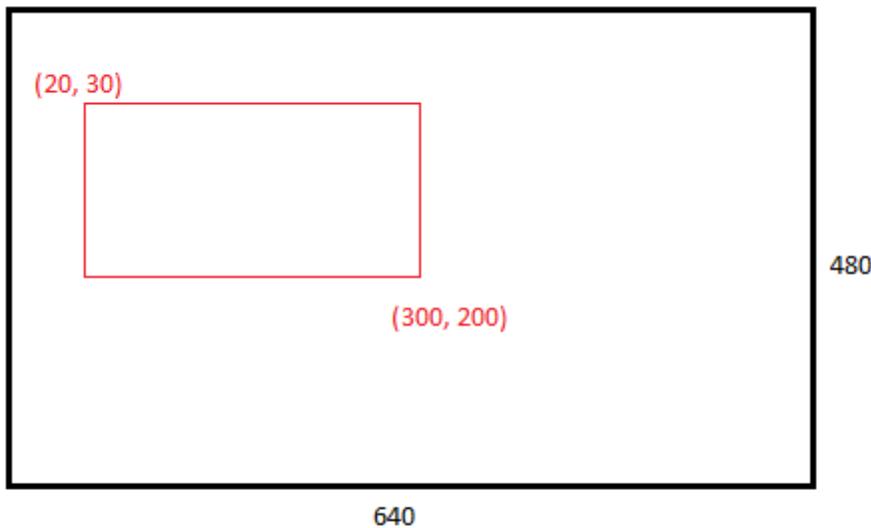
Arguments:

Argument	Valid values	Description
Camera*	String	Camera name
Width*	Integer. X >= 1	Image width
Height*	Integer. X >= 1	Image height
Left*	Integer. X >= 0 <= Width	Left position of rectangle
Right*	Integer. X >= Left <= Width	Right position of rectangle
Top*	Integer. X >= 0 <= Height	Top position of rectangle
Bottom*	Integer. X >= Top <= Height	Bottom position of rectangle
Speed	Integer. 0 >= X <= 100	Speed of operation
VCam	Integer. X >= -1	Virtual Camera ID (For fisheye cameras with Multi-View directly from camera)

* Mandatory parameters

Note: Some commands/parameters may not work with some cameras due to limitations of the communication interface itself developed by the camera's manufacturer.

Example: Perform zoom on an area with speed of 50 using the following image as reference:



```
http://192.168.0.1:8601/Interface/Cameras/PTZ/AreaZoom?Camera=Camera1&
Width=640&Height=480&Left=20&Top=30&Right=300&Bottom=200&Speed=50
```

Response:

Default response of API. In case the "Area Zoom" command group is not supported by the camera's driver (due to restrictions in the communication interface itself developed by the camera's manufacturer), the error code 10105 (Command not supported by the PTZ driver) will be returned.

HTTP Return HTTP: 200 OK

Parameters of return: [Default return of API](#)

4.3.12.5 Simultaneous

By way of simultaneous PT, also known as "Click and Center", you will supply X and Y coordinates of the camera, moving the camera to the desired location.

Compatibility: All editions

Security level: Requires user authentication

Method: HTTP GET

Syntax:

```
http://<server_address>/Interface/Cameras/PTZ/Simultaneous?<argument=value>
[&<argument=value>...] [&<general_argument>...]
```

Arguments:

Argument	Valid values	Description
Camera*	String	Name of the camera
X*	Integer. 0 >= X <= 640	Position X over the proportion of an image with width of 640 pixels
Y*	Integer. 0 >= Y <= 480	Position Y over the proportion of an image with height of 480 pixels
Speed	Integer. 0 >= X <= 100	Speed of operation. If this parameter is omitted, the default value of 100 will be used
VCam	Integer. X >= -1	Virtual Camera ID (For fisheye cameras with Multi-View directly from camera)

* Mandatory parameters

Note: The values of X and Y are calculated based on an image of 640x480. To calculate X and Y using different proportions, simply apply the Rule of 3 on the image.

Ex:

Image of 352x240

Desired position: X = 50, Y = 20

Calculation of X:

$(50 * 640) / 352 = 90$

Calculation of Y:

$(20 * 480) / 240 = 40$

X = 90, Y = 40

Note: Some commands/parameters may not work with some cameras due to limitations of the communication interface itself developed by the camera's manufacturer.

Example 1: Center a camera on the position X = 20, Y = 30 at a speed of 50

```
http://192.168.0.1:8601/Interface/Cameras/PTZ/Simultaneous?Camera=Camera1&
X=20&Y=30&Speed=50
```

Example 2: Moves a camera to the position X = 500, Y = 440 at a speed of 100 and response in text

```
http://192.168.0.1:8601/Interface/Cameras/PTZ/Simultaneous?Camera=Camera1&
X=500&Y=440&Speed=100&ResponseFormat=Text
```

Example 3: Moves a camera to the position X = 10, Y = 350 with response in XML and authentication of

the user User1

```
http://192.168.0.1:8601/Interface/Cameras/PTZ/Simultaneous?Camera=Camera1&X=10&Y=350&ResponseFormat=XML&AuthUser=User1&AuthPass=User1Pass
```

Response:

Default response of API. In case the "Simultaneous" command group is not supported by the camera's driver (due to restrictions in the communication interface itself developed by the camera's manufacturer), the error code 10105 (Command not supported by the PTZ driver) will be returned.

HTTP Return: 200 OK

Parameters of return: [Default return of API](#)

4.3.12.6 Continuous

By way of the continuous movement command, you will be able to control the cameras emulating a joystick

Compatibility: All editions

Security level: Requires user authentication

Method: HTTP GET

Syntax:

```
http://<server_address>/Interface/Cameras/PTZ/Continuous?<argument=value>
[&<argument=value>...] [&<general\_argument>...]
```

Arguments:

Argument	Valid values	Description								
Camera*	String	Name of the camera								
Pan	Integer. -100 >= X <= 100	<p>Movement speed for Pan</p> <table border="1"> <thead> <tr> <th>Type</th><th>Description</th></tr> </thead> <tbody> <tr> <td>0</td><td>Stop</td></tr> <tr> <td>Negative values</td><td>Left</td></tr> <tr> <td>Positive values</td><td>Right</td></tr> </tbody> </table> <p>If this parameter is omitted, the default value of 0 (Stop) will be used</p>	Type	Description	0	Stop	Negative values	Left	Positive values	Right
Type	Description									
0	Stop									
Negative values	Left									
Positive values	Right									
Tilt	Integer. -100 >= X <= 100	<p>Movement speed for Tilt</p> <table border="1"> <thead> <tr> <th>Type</th><th>Description</th></tr> </thead> <tbody> <tr> <td>0</td><td>Stop</td></tr> <tr> <td>Negative values</td><td>Up</td></tr> <tr> <td>Positive values</td><td>Down</td></tr> </tbody> </table> <p>If this parameter is omitted, the default value of 0 (Stop) will be used</p>	Type	Description	0	Stop	Negative values	Up	Positive values	Down
Type	Description									
0	Stop									
Negative values	Up									
Positive values	Down									
Zoom	Integer. -100 >= X <= 100	<p>Movement speed for Zoom</p> <table border="1"> <thead> <tr> <th>Type</th><th>Description</th></tr> </thead> <tbody> <tr> <td>0</td><td>Stop</td></tr> </tbody> </table>	Type	Description	0	Stop				
Type	Description									
0	Stop									

		<table border="1"> <tr><td>Negative values</td><td>Zoom Out</td></tr> <tr><td>Positive values</td><td>Zoom In</td></tr> </table> <p>If this parameter is omitted, the default value of 0 (Stop) will be used</p>	Negative values	Zoom Out	Positive values	Zoom In				
Negative values	Zoom Out									
Positive values	Zoom In									
Focus	Integer. -100 >= X <= 100	<p>Movement speed for Focus</p> <table border="1"> <thead> <tr><th>Type</th><th>Description</th></tr> </thead> <tbody> <tr><td>0</td><td>Stop</td></tr> <tr><td>Negative values</td><td>Focus Near</td></tr> <tr><td>Positive values</td><td>Focus Far</td></tr> </tbody> </table> <p>If this parameter is omitted, the default value of 0 (Stop) will be used</p>	Type	Description	0	Stop	Negative values	Focus Near	Positive values	Focus Far
Type	Description									
0	Stop									
Negative values	Focus Near									
Positive values	Focus Far									
Iris	Integer. -100 >= X <= 100	<p>Movement Speed for Iris</p> <table border="1"> <thead> <tr><th>Type</th><th>Description</th></tr> </thead> <tbody> <tr><td>0</td><td>Stop</td></tr> <tr><td>Negative values</td><td>Close Iris</td></tr> <tr><td>Positive values</td><td>Open Iris</td></tr> </tbody> </table> <p>If this parameter is omitted, the default value of 0 (Stop) will be used</p>	Type	Description	0	Stop	Negative values	Close Iris	Positive values	Open Iris
Type	Description									
0	Stop									
Negative values	Close Iris									
Positive values	Open Iris									
VCam	Integer. X >= -1	Virtual Camera ID (For fisheye cameras with Multi-View directly from camera)								

* Mandatory parameters

Note: When a movement command is sent, the camera will apply the movement until the command with the parameters for Stop is sent.

Note: Some commands/parameters may not work with some cameras due to limitations of the communication interface itself developed by the camera's manufacturer. The basic parameters of Up, Down, Left, Right, Zoom In and Zoom Out should work for all cameras, whereas the parameters of Simultaneous Movement, Home Position, Focus and Iris may vary from one manufacturer to another.

Example 1: Moves a camera to the right and downwards at Pan speed 50 and Tilt speed 60
`http://192.168.0.1:8601/Interface/Cameras/PTZ/Continuous?Camera=Camera1&Pan=50&Tilt=60`

Example 2: Stops the Pan and Tilt movement of a camera with response in text
`http://192.168.0.1:8601/Interface/Cameras/PTZ/Continuous?Camera=Camera1&Pan=0&Tilt=0&ResponseFormat=Text`

Example 3: Executes Zoom In in a camera at a speed of 100, response in XML and authentication of the user User1
`http://192.168.0.1:8601/Interface/Cameras/PTZ/Continuous?Camera=Camera1&Zoom=100&ResponseFormat=XML&AuthUser=User1&AuthPass=User1Pass`

Response:

Default response of API. In case the "Continuous" command group is not supported by the camera's driver (due to restrictions in the communication interface itself developed by the camera's manufacturer), the error code 10105 (Command not supported by the PTZ driver) will be returned.

HTTP Return: 200 OK

Parameters of return: [Default return of API](#)

4.3.12.7 Auto Focus

By way of this command, you will be able to activate or deactivate the Auto Focus.

Compatibility: All editions

Security level: Requires user authentication

Method: HTTP GET

Syntax:

```
http://<server_address>/Interface/Cameras/PTZ/AutoFocus?<argument=value>
[&<argument=value>...] [&<general argument>...]
```

Arguments:

Argument	Valid values	Description
Camera*	String	Name of the camera
Operation*	ON OFF	Turn on or off the Auto Focus
VCam	Integer. X >= -1	Virtual Camera ID (For fisheye cameras with Multi-View directly from camera)

* Mandatory parameters

Note: Some commands/parameters may not work with some cameras due to limitations of the communication interface itself developed by the camera's manufacturer.

Example 1: Activates the auto focus of a camera with response in text

```
http://192.168.0.1:8601/Interface/Cameras/PTZ/AutoFocus?Camera=Camera1&
Operation=ON&ResponseFormat=Text
```

Example 2: Deactivates the auto focus of a camera with response in XML

```
http://192.168.0.1:8601/Interface/Cameras/PTZ/AutoFocus?Camera=Camera1&
Operation=OFF&ResponseFormat=XML
```

Response:

Default response of API. In case the "Auto Focus" command group is not supported by the camera's driver (due to restrictions in the communication interface itself developed by the camera's manufacturer), the error code 10105 (Command not supported by the PTZ driver) will be returned.

HTTP Return: 200 OK

Parameters of return: [Default return of API](#)

4.3.12.8 Auto Iris

By way of this command, you will be able to activate or deactivate the Auto Iris.

Compatibility: All editions

Security level: Requires user authentication

Method: HTTP GET

Syntax:

```
http://<server_address>/Interface/Cameras/PTZ/AutoIris?<argument=value>
[&<argument=value>...] [&<general argument>...]
```

Arguments:

Argument	Valid values	Description
Camera*	String	Name of the camera
Operation*	ON OFF	Turn on or off the Auto Iris
VCam	Integer. X >= -1	Virtual Camera ID (For fisheye cameras with Multi-View directly from camera)

* Mandatory parameters

Note: Some commands/parameters may not work with some cameras due to limitations of the communication interface itself developed by the camera's manufacturer.

Example 1: Activates the auto iris of a camera with response in text

```
http://192.168.0.1:8601/Interface/Cameras/PTZ/AutoIris?Camera=Camera1&
Operation=ON&ResponseFormat=Text
```

Example 2: Deactivates the auto iris of a camera with response in XML

```
http://192.168.0.1:8601/Interface/Cameras/PTZ/AutoIris?Camera=Camera1&
Operation=OFF&ResponseFormat=XML
```

Response:

Default response of API. In case the "Auto Iris" command group is not supported by the camera's driver (due to restrictions in the communication interface itself developed by the camera's manufacturer), the error code 10105 (Command not supported by the PTZ driver) will be returned.

HTTP Return: 200 OK

Parameters of return: [Default return of API](#)

4.3.12.9 Menu control

Some cameras (especially analog ones) allow the remote control of the menu. By way of this command, you will have total control over the menu of cameras.

Compatibility: All editions

Security level: Requires user authentication

Method: HTTP GET

Syntax:

```
http://<server_address>/Interface/Cameras/PTZ/MenuControl?<argument=value>
[&<argument=value>...] [&<general argument>...]
```

Arguments:

Argument	Valid values	Description																		
Camera*	String	Name of the camera																		
Operation*	Open Close Left Right Up Down Enter Cancel	Control operation to be executed																		
		<table border="1"> <thead> <tr> <th>Operation</th><th>Description</th></tr> </thead> <tbody> <tr> <td>Open</td><td>Open the menu</td></tr> <tr> <td>Close</td><td>Close the menu</td></tr> <tr> <td>Left</td><td>Move the cursor to the left</td></tr> <tr> <td>Right</td><td>Move the cursor to the right</td></tr> <tr> <td>Up</td><td>Move the cursor upwards</td></tr> <tr> <td>Down</td><td>Move the cursor downwards</td></tr> <tr> <td>Enter</td><td>"Enter" key</td></tr> <tr> <td>Cancel</td><td>Cancel the operation</td></tr> </tbody> </table>	Operation	Description	Open	Open the menu	Close	Close the menu	Left	Move the cursor to the left	Right	Move the cursor to the right	Up	Move the cursor upwards	Down	Move the cursor downwards	Enter	"Enter" key	Cancel	Cancel the operation
Operation	Description																			
Open	Open the menu																			
Close	Close the menu																			
Left	Move the cursor to the left																			
Right	Move the cursor to the right																			
Up	Move the cursor upwards																			
Down	Move the cursor downwards																			
Enter	"Enter" key																			
Cancel	Cancel the operation																			
VCam	Integer. X >= -1	Virtual Camera ID (For fisheye cameras with Multi-View directly from camera)																		

* Mandatory parameters

Note: Some commands/parameters may not work with some cameras due to limitations of the communication interface itself developed by the camera's manufacturer.

Exemple 1: Open the menu of a camera

```
http://192.168.0.1:8601/Interface/Cameras/PTZ/MenuControl?Camera=Camera1&Operation=Open
```

Example 2: Close the menu of a camera with response in text

```
http://192.168.0.1:8601/Interface/Cameras/PTZ/MenuControl?Camera=Camera1&Operation=Close&ResponseFormat=Text
```

Example 3: Move the menu's cursor downwards with response in XML and authentication of the user User1

```
http://192.168.0.1:8601/Interface/Cameras/PTZ/MenuControl?Camera=Camera1&Operation=Down&ResponseFormat=XML&AuthUser=User1&AuthPass=User1Pass
```

Response:

Default response of API. In case the "Menu Control" command group is not supported by the camera's driver (due to restrictions in the communication interface itself developed by the camera's manufacturer), the error code 10105 (Command not supported by the PTZ driver) will be returned.

HTTP Return: 200 OK

Parameters of return: [Default return of API](#)

4.3.12.10 Presets

4.3.12.10.1 List of presets

Requests the list of presets of a camera.

Compatibility: All editions

Security level: Requires user authentication

Method: HTTP GET**Syntax:**

```
http://<server_address>/Interface/Cameras/PTZ/GetPresets?<argument=value>
[&<argument=value>...] [&<general argument>...]
```

Arguments:

Argument	Valid values	Description
Camera*	String	Name of the camera

* Mandatory parameters

Example 1: Requests the list of presets of a camera with response in text

```
http://192.168.0.1:8601/Interface/Cameras/PTZ/GetPresets?Camera=Camera1&
ResponseFormat=Text
```

Example 2: Requests the list of presets of a camera with response in XML

```
http://192.168.0.1:8601/Interface/Cameras/PTZ/GetPresets?Camera=Camera1&
ResponseFormat=XML
```

Response:

A list with the presets of a specified camera is returned.

HTTP Return: 200 OK

Parameters of return:**Fixed Parameters:**

Parameter	Type	Description
COUNT	Integer	Total number of presets (Not included in JSON response)

Parameters of the list of presets:

Parameter	Type	Description
ID	Integer	Number of the preset
DESCRIPTION	String	Description of the preset

Example of return in text:

```
RESPONSE_CODE=0
RESPONSE_MESSAGE=OK
COUNT=3
PRESET_1_ID=0
PRESET_1_DESCRIPTION=Position 1
PRESET_2_ID=1
PRESET_2_DESCRIPTION=Position 2
PRESET_3_ID=2
PRESET_3_DESCRIPTION=Position 3
```

Example of return in XML:

```

<Response>
  <Code>0</Code>
  <Message>OK</Message>
  <Data>
    <Presets>
      <Count>3</Count>
      <Preset>
        <ID>0</ID>
        <Description>Position 1</Description>
      </Preset>
      <Preset>
        <ID>1</ID>
        <Description>Position 2</Description>
      </Preset>
      <Preset>
        <ID>2</ID>
        <Description>Position 3</Description>
      </Preset>
    </Presets>
  </Data>
</Response>

```

Example of return in JSON:

```
{
  "Response": {
    "Code": 0,
    "Message": "OK",
    "Data": {
      "Presets": [
        {
          "ID": 0,
          "Description": "Position 1"
        },
        {
          "ID": 1,
          "Description": "Position 2"
        },
        {
          "ID": 2,
          "Description": "Position 3"
        }
      ]
    }
  }
}
```

4.3.12.10.2 Call a preset

Command for calling a preset of a camera.

Compatibility: All editions

Security level: Requires user authentication

Method: HTTP GET

Syntax:

```
http://<server_address>/Interface/Cameras/PTZ/CallPreset?<argument=value>
[&<argument=value>...] [&<general argument>...]
```

Arguments:

Argument	Valid values	Description
Camera*	String	Name of the camera
Value*	Integer. X >= 0	Number of the preset
Speed	Integer. 0 >= X <= 100	Speed of operation. If this parameter is omitted, the default value of 100 will be used
VCam	Integer. X >= -1	Virtual Camera ID (For fisheye cameras with Multi-View directly from camera)

* Mandatory parameters

Note: Some commands/parameters may not work with some cameras due to limitations of the communication interface itself developed by the camera's manufacturer.

Example 1: Call the preset 10 with speed of 90

```
http://192.168.0.1:8601/Interface/Cameras/PTZ/CallPreset?Camera=Camera1&
Value=10&Speed=90
```

Example 2: Call the preset 15 with response in text

```
http://192.168.0.1:8601/Interface/Cameras/PTZ/CallPreset?Camera=Camera1&
Value=15&ResponseFormat=Text
```

Example 3: Call the preset 20 with response in XML and authentication of the user User1

```
http://192.168.0.1:8601/Interface/Cameras/PTZ/CallPreset?Camera=Camera1&
Value=20&ResponseFormat=XML&AuthUser=User1&AuthPass=User1Pass
```

Response:

Default response of API. In case the "Call Preset" command group is not supported by the camera's driver (due to restrictions in the communication interface itself developed by the camera's manufacturer), the error code 10105 (Command not supported by the PTZ driver) will be returned.

HTTP Return: 200 OK

Parameters of return: [Default return of API](#)

4.3.12.10.3 Set a preset

Command for setting a preset of a camera. If the specified preset does not exist, a new one will be created, otherwise the current preset will be overwritten. The preset will be set using the current position of the camera.

Compatibility: All editions

Security level: Requires user authentication

Method: HTTP GET

Syntax:

```
http://<server_address>/Interface/Cameras/PTZ/SetPreset?<argument=value>
[&<argument=value>...] [&<general\_argument>...]
```

Arguments:

Argument	Valid values	Description
Camera*	String	Name of the camera
Value*	Integer. X >= 0	Number of the preset
Description*	String	Description of the preset
VCam	Integer. X >= -1	Virtual Camera ID (For fisheye cameras with Multi-View directly from camera)

* Mandatory parameters

Note: Some commands/parameters may not work with some cameras due to limitations of the communication interface itself developed by the camera's manufacturer.

Example 1: Set the preset 10 from Camera1 with description "Position 1"

```
http://192.168.0.1:8601/Interface/Cameras/PTZ/SetPreset?Camera=Camera1&
Value=10&Description=Position 1
```

Example 2: Set the preset 4 from Camera2 with description "Hallway entrance"

```
http://192.168.0.1:8601/Interface/Cameras/PTZ/CallPreset?Camera=Camera2&
Value=4&Description=Hallway Entrance&ResponseFormat=Text
```

Example 3: Set the preset 20 from Camera3 with description "Backyard" with response in XML and authentication of the user User1

```
http://192.168.0.1:8601/Interface/Cameras/PTZ/SetPreset?Camera=Camera3&
Value=20&Description=Backyard&ResponseFormat=XML&AuthUser=User1&
AuthPass=User1Pass
```

Response:

Default response of API. In case the "Set Preset" command group is not supported by the camera's driver (due to restrictions in the communication interface itself developed by the camera's manufacturer), the error code 10105 (Command not supported by the PTZ driver) will be returned.

HTTP Return: 200 OK

Parameters of return: [Default return of API](#)

4.3.12.11 Call a pattern

Command for calling the pattern of a camera.

Compatibility: All editions

Security level: Requires user authentication

Method: HTTP GET

Syntax:

```
http://<server_address>/Interface/Cameras/PTZ/CallPattern?<argument=value>
[&<argument=value>...] [&<general\_argument>...]
```

Arguments:

Argument	Valid values	Description
Camera*	String	Name of the camera
Value*	Integer. X >= 0	Number of the pattern
Speed	Integer. 0 >= X <= 100	Speed of operation. If this parameter is omitted, the default value of 100 will be used
VCam	Integer. X >= -1	Virtual Camera ID (For fisheye cameras with Multi-View directly from camera)

* Mandatory parameters

Note: Some commands/parameters may not work with some cameras due to limitations of the communication interface itself developed by the camera's manufacturer.

Example 1: Call the pattern 10 with speed of 90

```
http://192.168.0.1:8601/Interface/Cameras/PTZ/CallPattern?Camera=Camera1&
Value=10&Speed=90
```

Example 2: Call the pattern 15 with response in text

```
http://192.168.0.1:8601/Interface/Cameras/PTZ/CallPattern?Camera=Camera1&
Value=15&ResponseFormat=Text
```

Example 3: Call the pattern 20 with response in XML and authentication of the user User1

```
http://192.168.0.1:8601/Interface/Cameras/PTZ/CallPattern?Camera=Camera1&
Value=20&ResponseFormat=XML&AuthUser=User1&AuthPass=User1Pass
```

Response:

Default response of API. In case the "Pattern" command group is not supported by the camera's driver (due to restrictions in the communication interface itself developed by the camera's manufacturer), the error code 10105 (Command not supported by the PTZ driver) will be returned.

HTTP Return: 200 OK

Parameters of return: [Default return of API](#)

4.3.12.12 Windshield wiper

By way of this command, you will be able to activate or deactivate the windshield wiper (if the camera has one).

Compatibility: All editions

Security level: Requires user authentication

Method: HTTP GET

Syntax:

```
http://<server_address>/Interface/Cameras/PTZ/Wiper?<argument=value>
[&<argument=value>...] [&<general\_argument>...]
```

Arguments:

Argument	Valid values	Description
Camera*	String	Name of the camera
Operation*	ON OFF	Turns the windshield wiper on or off

VCam	Integer. X >= -1	Virtual Camera ID (For fisheye cameras with Multi-View directly from camera)
------	------------------	--

* Mandatory parameters

Note: Some commands/parameters may not work with some cameras due to limitations of the communication interface itself developed by the camera's manufacturer.

Example 1: Activates the windshield wiper of a camera with response in text

```
http://192.168.0.1:8601/Interface/Cameras/PTZ/Wiper?Camera=Camera1&
Operation=ON&ResponseFormat=Text
```

Example 2: Deactivates the windshield wiper of a camera with response in XML

```
http://192.168.0.1:8601/Interface/Cameras/PTZ/Wiper?Camera=Camera1&
Operation=OFF&ResponseFormat=XML
```

Response:

Default response of API. In case the "Wiper" command group is not supported by the camera's driver (due to restrictions in the communication interface itself developed by the camera's manufacturer), the error code 10105 (Command not supported by the PTZ driver) will be returned.

HTTP Return: 200 OK

Parameters of return: [Default return of API](#)

4.3.12.13 Auxiliary

4.3.12.13.1 List of Auxiliaries

Requests the list of auxiliaries of a camera.

Compatibility: All editions

Security level: Requires user authentication

Method: HTTP GET

Syntax:

```
http://<server_address>/Interface/Cameras/PTZ/GetAuxiliaries?
<argument=value>[&<argument=value>...][&<general_argument>...]
```

Arguments:

Argument	Valid values	Description
Camera*	String	Name of the camera

* Mandatory parameters

Example 1: Requests the list of auxiliaries of a camera with response in text

```
http://192.168.0.1:8601/Interface/Cameras/PTZ/GetAuxiliaries?
Camera=Camera1&ResponseFormat=Text
```

Example 2: Requests the list of auxiliaries of a camera with response in XML

```
http://192.168.0.1:8601/Interface/Cameras/PTZ/GetAuxiliaries?
Camera=Camera1&ResponseFormat=XML
```

Response:

A list with the auxiliaries of a specified camera is returned.

HTTP Return: 200 OK

Parameters of return:**Fixed Parameters:**

Parameter	Type	Description
COUNT	Integer	Total number of auxiliaries (Not included in JSON response)

Parameters of the list of auxiliaries:

Parameter	Type	Description
ID	Integer	Number of the auxiliary
DESCRIPTION	String	Description of the auxiliary

Example of return in text:

```
RESPONSE_CODE=0
RESPONSE_MESSAGE=OK
COUNT=2
AUXILIARY_1_ID=0
AUXILIARY_1_DESCRIPTION=Wiper
AUXILIARY_2_ID=1
AUXILIARY_2_DESCRIPTION=Alarm
```

Example of return in XML:

```
<Response>
  <Code>0</Code>
  <Message>OK</Message>
  <Data>
    <Auxiliaries>
      <Count>2</Count>
      <Auxiliary>
        <ID>0</ID>
        <Description>Wiper</Description>
      </Auxiliary>
      <Auxiliary>
        <ID>1</ID>
        <Description>Alarm</Description>
      </Auxiliary>
    </Auxiliaries>
  </Data>
</Response>
```

Example of return in JSON:

```
{
  "Response": {
    "Code": 0,
    "Message": "OK",
    "Data": {
      "Auxiliaries": [
        {
          "ID": 0,
          "Description": "Wiper"
        },
        {
          "ID": 1,
          "Description": "Alarm"
        }
      ]
    }
  }
}
```

4.3.12.13.2 Call an Auxiliary

With this command you can activate or deactivate the auxiliary controls the camera (If it has).

Compatibility: All editions

Security level: Requires user authentication

Method: HTTP GET

Syntax:

```
http://<server_address>/Interface/Cameras/PTZ/Auxiliary?<argument=value>
[&<argument=value>...] [&<general argument>...]
```

Arguments:

Arguments	Valid values	Description
Camera*	String	Camera name
Operation*	ON OFF	Turn on or off the auxiliary control
Value*	Integer	Auxiliary control number
VCam	Integer. X >= -1	Virtual Camera ID (For fisheye cameras with Multi-View directly from camera)

* Mandatory parameters

Note: Some commands/parameters may not work with some cameras due to limitations of the communication interface itself developed by the camera's manufacturer.

Example 1: Activates the auxiliary command 1 of a camera with response in text

```
http://192.168.0.1:8601/Interface/Cameras/PTZ/Auxiliary?Camera=Camera1&
Operation=ON&Value=1&ResponseFormat=Text
```

Example 2: Deactivates the auxiliary command 1 of a camera with response in XML

```
http://192.168.0.1:8601/Interface/Cameras/PTZ/Auxiliary?Camera=Camera1&
Operation=OFF&Value=1&ResponseFormat=XML
```

Example 3: Activates the auxiliary command 3 of a camera with response in XML

```
http://192.168.0.1:8601/Interface/Cameras/PTZ/Auxiliary?Camera=Camera1&Operation=ON&Value=3&ResponseFormat=XML
```

Response:

Default response of API. In case the "Auxiliary" command group is not supported by the camera's driver (due to restrictions in the communication interface itself developed by the camera's manufacturer), the error code 10105 (Command not supported by the PTZ driver) will be returned.

HTTP Return: 200 OK

Parameters of return: [Default return of API](#)

4.3.12.14 Patrol

4.3.12.14.1 Requesting the list of patrols

Request the list of PTZ Patrols of a given camera

Compatibility: Standard, Professional, Enterprise

Security level: Requires user authentication

Method: HTTP GET

Syntax:

```
http://<server_address>/Interface/Cameras/PTZ/GetPatrols?<argument=value>
[ &<argument=value>... ] [ &<general\_argument>... ]
```

Arguments:

Argument	Valid values	Description
Camera*	String	Name of the camera

* Mandatory parameters

Example 1: Request the list of PTZ Patrols of a camera with response in text

```
http://192.168.0.1:8601/Interface/Cameras/PTZ/GetPatrols?Camera=Camera1&ResponseFormat=Text
```

Example 2: Request the list of PTZ Patrols of a camera with response in XML

```
http://192.168.0.1:8601/Interface/Cameras/PTZ/GetPatrols?Camera=Camera1&ResponseFormat=XML
```

Response:

A list with the PTZ Patrols of a specified camera is returned.

HTTP Return: 200 OK

Parameters of return:

Fixed Parameters:

Parameter	Type	Description
COUNT	Integer	Total number of patrols (Not included in JSON response)

Parameters of the list of patrols:

Parameter	Type	Description
ID	Integer	PTZ Patrol ID
NAME	String	PTZ Patrol Name
DESCRIPTION	String	PTZ Patrol Description

Example of return in text:

```
RESPONSE_CODE=0
RESPONSE_MESSAGE=OK
COUNT=2
PATROL_1_ID=0
PATROL_1_NAME=Day
PATROL_1_DESCRIPTION=Daytime operation
PATROL_2_ID=1
PATROL_2_NAME=Night
PATROL_2_DESCRIPTION=Nighttime operation
```

Example of return in XML:

```
<Response>
  <Code>0</Code>
  <Message>OK</Message>
  <Data>
    <Patrols>
      <Count>2</Count>
      <Patrol>
        <ID>0</ID>
        <Name>Day</Name>
        <Description>Daytime operation</Description>
      </Patrol>
      <Patrol>
        <ID>1</ID>
        <Name>Night</Name>
        <Description>Nighttime operation</Description>
      </Patrol>
    </Patrols>
  </Data>
</Response>
```

Example of return in JSON:

```
{
  "Response": {
    "Code": 0,
    "Message": "OK",
    "Data": {
      "Patrols": [
        {
          "ID": 0,
          "Name": "Day",
          "Description": "Daytime operation"
        },
        {
          "ID": 1,
          "Name": "Night",
          "Description": "Nighttime operation"
        }
      ]
    }
  }
}
```

4.3.12.14.2 Requesting the status of PTZ patrol

Command to query the current status of PTZ Patrol of a given camera

Compatibility: Standard, Professional, Enterprise

Security level: Requires user authentication

Method: HTTP GET

Syntax:

```
http://<server_address>/Interface/Cameras/PTZ/GetPatrolStatus?
<argument=value>[&<argument=value>...][&<general argument>...]
```

Arguments:

Argument	Valid values	Description
Camera*	String	Name of the camera

* Mandatory parameters

Example 1: Query PTZ Patrol status of "Camera1"

```
http://192.168.0.1:8601/Interface/Cameras/PTZ/GetPatrolStatus?
Camera=Camera1
```

Example 2: Query PTZ Patrol status of "Camera1" with response in text

```
http://192.168.0.1:8601/Interface/Cameras/PTZ/GetPatrolStatus?
Camera=Camera1&ResponseFormat=Text
```

Response:

A list of parameter-value pairs is returned

HTTP Return HTTP: 200 OK

Parameters of return:

Parameter	Type	Description								
PATROLTYPE	AUTO MANUAL	Type of PTZ patrol								
CURRENTPATROLID	Integer	ID of current PTZ patrol								
CURRENTPATROLNAME	String	Name of current PTZ patrol								
PAUSED	Boolean	Status of PTZ patrol <table border="1" data-bbox="714 487 1432 593"> <thead> <tr> <th>Status</th><th>Description</th></tr> </thead> <tbody> <tr> <td>TRUE</td><td>PTZ patrol is paused</td></tr> <tr> <td>FALSE</td><td>PTZ patrol is running</td></tr> </tbody> </table>	Status	Description	TRUE	PTZ patrol is paused	FALSE	PTZ patrol is running		
Status	Description									
TRUE	PTZ patrol is paused									
FALSE	PTZ patrol is running									
STATE	STOPPED PAUSED PLAYING	State of PTZ patrol <table border="1" data-bbox="714 692 1432 840"> <thead> <tr> <th>State</th><th>Description</th></tr> </thead> <tbody> <tr> <td>STOPPED</td><td>PTZ Patrol is stopped</td></tr> <tr> <td>PAUSED</td><td>PTZ Patrol is paused</td></tr> <tr> <td>PLAYING</td><td>PTZ Patrol is running</td></tr> </tbody> </table>	State	Description	STOPPED	PTZ Patrol is stopped	PAUSED	PTZ Patrol is paused	PLAYING	PTZ Patrol is running
State	Description									
STOPPED	PTZ Patrol is stopped									
PAUSED	PTZ Patrol is paused									
PLAYING	PTZ Patrol is running									

Example of return in text:

```
RESPONSE_CODE=0
RESPONSE_MESSAGE=OK
PATROLTYPE=AUTO
CURRENTPATROLID=0
CURRENTPATROLNAME=Day
PAUSED=FALSE
STATE=PLAYING
```

Example of return in XML:

```
<Response>
  <Code>0</Code>
  <Message>OK</Message>
  <Data>
    <Status>
      <PatrolType>AUTO</PatrolType>
      <CurrentPatrolID>0</CurrentPatrolID>
      <CurrentPatrolName>Day</CurrentPatrolName>
      <Paused>FALSE</Paused>
      <State>PLAYING</State>
    </Status>
  </Data>
</Response>
```

Example of return in JSON:

```
{
  "Response": {
    "Code": 0,
    "Message": "OK",
    "Data": {
      "Status": {
        "PatrolType": "AUTO",
        "CurrentPatrolID": 0,
        "CurrentPatrolName": "Day",
        "Paused": false,
        "State": "PLAYING"
      }
    }
  }
}
```

4.3.12.14.3 Controlling PTZ Patrol

Command to control the PTZ Patrol of a given camera

Compatibility: Standard, Professional, Enterprise

Security level: Requires user authentication

Method: HTTP GET

Syntax:

```
http://<server_address>/Interface/Cameras/PTZ/Patrol?
<argument=value>[&<argument=value>... ] [&<general argument>... ]
```

Arguments:

Argument	Valid values	Description										
Camera*	String	Name of the camera										
Action*	START PAUSE RESUME STOP	PTZ Patrol Action <table border="1" data-bbox="665 1326 1390 1510"> <thead> <tr> <th>Action</th><th>Description</th></tr> </thead> <tbody> <tr> <td>START</td><td>Start a given patrol</td></tr> <tr> <td>PAUSE</td><td>Pause current patrol operation</td></tr> <tr> <td>RESUME</td><td>Resume current patrol operation</td></tr> <tr> <td>STOP</td><td>Stop current patrol operation</td></tr> </tbody> </table>	Action	Description	START	Start a given patrol	PAUSE	Pause current patrol operation	RESUME	Resume current patrol operation	STOP	Stop current patrol operation
Action	Description											
START	Start a given patrol											
PAUSE	Pause current patrol operation											
RESUME	Resume current patrol operation											
STOP	Stop current patrol operation											
ID**	Integer	ID of the patrol This parameter is mandatory when using Action=START										

* Mandatory parameters

** Conditionally mandatory parameters

Example 1: Start patrol 1 of "Camera1"

```
http://192.168.0.1:8601/Interface/Cameras/PTZ/Patrol?Camera=1&
Action=START&ID=1
```

Example 2: Pause current patrol of "Camera1"

```
http://192.168.0.1:8601/Interface/Cameras/PTZ/Patrol?Camera=1&
Action=PAUSE
```

Example 3: Resume current patrol of "Camera1"

```
http://192.168.0.1:8601/Interface/Cameras/PTZ/Patrol?Camera=1&
Action=RESUME
```

Response:

Default response of API.

HTTP Return: 200 OK

Parameters of return: [Default return of API](#)

4.3.12.15 Supported commands

Returns a list of the PTZ commands that are supported by a given camera.

Compatibility: All editions

Security level: Requires user authentication

Method: HTTP GET

Syntax:

```
http://<server_address>/Interface/Cameras/PTZ/GetSupportedCommands?
<argument=value>[&<argument=value>...][&<general argument>...]
```

Arguments:

Argument	Valid values	Description
Camera*	String	Name of the camera

* Mandatory parameters

Example 1: Requests the supported PTZ commands of a camera with response in text

```
http://192.168.0.1:8601/Interface/Cameras/PTZ/GetSupportedCommands?
Camera=Cameral&ResponseFormat=Text
```

Example 2: Requests the supported PTZ commands PTZ of a camera with response in XML and authentication of the user User1

```
http://192.168.0.1:8601/Interface/Cameras/PTZ/GetSupportedCommands?
Camera=Cameral&ResponseFormat=XML&AuthUser=User1&AuthPass=User1Pass
```

Response:

A list of parameter-value pairs is returned

HTTP Return: 200 OK

Parameters of return:

Parameter	Type	Description
SIMPLE	Boolean	Support for the commands of the type "Simple"
RELATIVE	Boolean	Support for the commands of the type "Relative"
ABSOLUTE	Boolean	Support for the commands of the type "Absolute"
AREAZOOM	Boolean	Support for the commands of the type "AreaZoom"

Parameter	Type	Description
SIMULTANEOUS	Boolean	Support for the commands of the type "Simultaneous"
CONTINUOUS	Boolean	Support for the commands of the type "Continuous"
AUTOFOCUS	Boolean	Support for the commands of the type "Auto Focus"
AUTOIRIS	Boolean	Support for the commands of the type "Auto Iris"
MENUCONTROL	Boolean	Support for the commands of the type "Control of Menu"
CALLPATTERN	Boolean	Support for the commands of the type "Call a Pattern"
CALLPRESET	Boolean	Support for the commands of the type "Call a Preset"
WIPER	Boolean	Support for the commands of the type "Windshield Wiper"
AUXILIARY	Boolean	Support for the commands of the type "Auxiliary"
POSITIONMONITORING	Boolean	Support for position monitoring

Example of return in text:

```
RESPONSE_CODE=0
RESPONSE_MESSAGE=OK
SIMPLE=TRUE
RELATIVE=FALSE
ABSOLUTE=TRUE
AREAZOOM=FALSE
SIMULTANEOUS=FALSE
CONTINUOUS=TRUE
AUTOFOCUS=TRUE
AUTOIRIS=TRUE
MENUCONTROL=TRUE
CALLPATTERN=TRUE
CALLPRESET=TRUE
WIPER=TRUE
AUXILIARY=TRUE
POSITIONMONITORING=FALSE
```

Example of return in XML:

```
<Response>
  <Code>0</Code>
  <Message>OK</Message>
  <Data>
    <Commands>
      <Simple>TRUE</Simple>
      <Relative>FALSE</Relative>
      <Absolute>TRUE</Absolute>
      <AreaZoom>FALSE</AreaZoom>
      <Simultaneous>FALSE</Simultaneous>
      <Continuous>TRUE</Continuous>
      <AutoFocus>TRUE</AutoFocus>
      <AutoIris>TRUE</AutoIris>
      <MenuControl>TRUE</MenuControl>
      <CallPattern>TRUE</CallPattern>
      <CallPreset>TRUE</CallPreset>
      <Wiper>TRUE</Wiper>
      <Auxiliary>TRUE</Auxiliary>
      <PositionMonitoring>FALSE</PositionMonitoring>
    </Commands>
  </Data>
</Response>
```

Example of return in JSON:

```
{
  "Response": {
    "Code": 0,
    "Message": "OK",
    "Data": {
      "Commands": {
        "Simple": true,
        "Relative": false,
        "Absolute": true,
        "AreaZoom": false,
        "Simultaneous": false,
        "Continuous": true,
        "AutoFocus": true,
        "AutoIris": true,
        "MenuControl": true,
        "CallPattern": true,
        "CallPreset": true,
        "Wiper": true,
        "Auxiliary": true,
        "PositionMonitoring": false
      }
    }
  }
}
```

4.3.12.16 Requesting the status of PTZ

Request the status of the PTZ cameras over which the user has rights

Compatibility: All editions

Security level: Requires user authentication

Method: HTTP GET

Syntax:

```
http://<server_address>/Interface/Cameras/PTZ/GetStatus[?<argument=value>
[&<argument=value>...]] [&<general\_argument>...]]
```

Arguments:

Argument	Valid values	Description
Cameras	APIMasks	Mask to filter the results. Specify which cameras must be returned based on the provided masks.
Users	APIMasks	Mask to filter the results. The users mask will filter to include only cameras being used by the specified users.
IPs	APIMasks	Mask to filter the results. The IPs mask will filter to include only cameras being used by the specified IP addresses.

Example 1: Request the status of the PTZ of all cameras with all fields and response in XML

```
http://192.168.0.1:8601/Interface/Cameras/PTZ/GetStatus?ResponseFormat=XML
```

Example 2: Request the status of all cameras with response in text

```
http://192.168.0.1:8601/Interface/Cameras/PTZ/GetStatus?ResponseFormat=Text
```

Example 3: Request the status of all cameras starting with A, with response in text

```
http://192.168.0.1:8601/Interface/Cameras/PTZ/GetStatus?Cameras=A*&ResponseFormat=Text
```

Example 4: Request the status of all active cameras being used by user User1, with response in XML and authentication with admin user (No password)

```
http://192.168.0.1:8601/Interface/Cameras/PTZ/GetStatus?Users=User1&ResponseFormat=XML&AuthUser=admin
```

Response:

A list with the status of the PTZ from all of the cameras that the user has live view or playback rights is returned.

HTTP Return: 200 OK

Parameters of return:**Fixed Parameter:**

Parameter	Type	Description
COUNT	Integer	Total number of cameras

Parameters in the list of status of cameras:

Parameter	Type	Description
NAME	String	Name of the camera
CURRENTPRESET	Integer	Current preset that camera is positioned at
CURRENTPOSITION_SUPPORTED	Boolean	Supports position monitoring
CURRENTPOSITION_INITIALIZED	Boolean	Position monitoring is initialized (If FALSE, the system will activate monitoring automatically, call the command again in a few seconds to retrieve the position)
CURRENTPOSITION_PAN	Integer	Current Pan position -180 to 180 Negative values = Left Positive values = Right
CURRENTPOSITION_TILT	Integer	Current Tilt position -180 to 180 Negative values = Up Positive values = Down
CURRENTPOSITION_ZOOM	Integer	Current Zoom position 1 to 100
USAGE_INUSE	Boolean	Identify if the PTZ is currently being controlled
USAGE_USER	String	Name of the user controlling the camera
USAGE_IP	String	IP of the machine controlling the camera
LOCK_LOCKED	Boolean	Identify if the PTZ is currently locked to an user
LOCK_USER	String	Name of the user that locked the camera

Parameter	Type	Description
LOCK_IP	String	IP of the machine that locked the camera

Example of return in text:

```

RESPONSE_CODE=0
RESPONSE_MESSAGE=OK
COUNT=2
CAMERA_1_NAME=40
CAMERA_1_CURRENTPRESET=-1
CAMERA_1_CURRENTPOSITION_SUPPORTED=TRUE
CAMERA_1_CURRENTPOSITION_INITIALIZED=TRUE
CAMERA_1_CURRENTPOSITION_PAN=105
CAMERA_1_CURRENTPOSITION_TILT=9
CAMERA_1_CURRENTPOSITION_ZOOM=47
CAMERA_1_USAGE_INUSE=FALSE
CAMERA_1_USAGE_USER=admin
CAMERA_1_USAGE_IP=192.168.0.2
CAMERA_1_LOCK_LOCKED=FALSE
CAMERA_1_LOCK_USER=
CAMERA_1_LOCK_IP=
CAMERA_2_NAME=92
CAMERA_2_CURRENTPRESET=-1
CAMERA_2_CURRENTPOSITION_SUPPORTED=TRUE
CAMERA_2_CURRENTPOSITION_INITIALIZED=TRUE
CAMERA_2_CURRENTPOSITION_PAN=114
CAMERA_2_CURRENTPOSITION_TILT=84
CAMERA_2_CURRENTPOSITION_ZOOM=1
CAMERA_2_USAGE_INUSE=FALSE
CAMERA_2_USAGE_USER=
CAMERA_2_USAGE_IP=
CAMERA_2_LOCK_LOCKED=FALSE
CAMERA_2_LOCK_USER=
CAMERA_2_LOCK_IP=

```

Example of return in XML:

```
<Response>
  <Code>0</Code>
  <Message>OK</Message>
  <Data>
    <PTZStatus>
      <Count>2</Count>
      <Camera>
        <Name>40</Name>
        <CurrentPreset>-1</CurrentPreset>
        <CurrentPosition>
          <Supported>TRUE</Supported>
          <Initialized>TRUE</Initialized>
          <Pan>105</Pan>
          <Tilt>9</Tilt>
          <Zoom>47</Zoom>
        </CurrentPosition>
        <Usage>
          <InUse>FALSE</InUse>
          <User>admin</User>
          <IP>192.168.0.2</IP>
        </Usage>
        <Lock>
          <Locked>FALSE</Locked>
          <User />
          <IP />
        </Lock>
      </Camera>
      <Camera>
        <Name>92</Name>
        <CurrentPreset>-1</CurrentPreset>
        <CurrentPosition>
          <Supported>TRUE</Supported>
          <Initialized>TRUE</Initialized>
          <Pan>114</Pan>
          <Tilt>84</Tilt>
          <Zoom>1</Zoom>
        </CurrentPosition>
        <Usage>
          <InUse>FALSE</InUse>
          <User />
          <IP />
        </Usage>
        <Lock>
          <Locked>FALSE</Locked>
          <User />
          <IP />
        </Lock>
      </Camera>
    </PTZStatus>
  </Data>
</Response>
```

Example of return in JSON:

```
{  
    "Response": {  
        "Code": 0,  
        "Message": "OK",  
        "Data": [  
            {"PTZStatus": [  
                {  
                    "Name": "40",  
                    "CurrentPreset": -1,  
                    "CurrentPosition": {  
                        "Supported": true,  
                        "Initialized": true,  
                        "Pan": 105,  
                        "Tilt": 9,  
                        "Zoom": 47  
                    },  
                    "Usage": {  
                        "InUse": false,  
                        "User": "admin",  
                        "IP": "192.168.0.2"  
                    },  
                    "Lock": {  
                        "Locked": false,  
                        "User": "",  
                        "IP": ""  
                    }  
                },  
                {  
                    "Name": "92",  
                    "CurrentPreset": -1,  
                    "CurrentPosition": {  
                        "Supported": true,  
                        "Initialized": true,  
                        "Pan": 114,  
                        "Tilt": 84,  
                        "Zoom": 1  
                    },  
                    "Usage": {  
                        "InUse": false,  
                        "User": "",  
                        "IP": ""  
                    },  
                    "Lock": {  
                        "Locked": false,  
                        "User": "",  
                        "IP": ""  
                    }  
                }  
            ]  
        }  
    }  
}
```

4.3.13 I/O

4.3.13.1 Requesting the list of input events

Requests the list of input events of a camera.

Compatibility: Standard, Professional, Enterprise

Security level: Requires user authentication with rights to camera registration

Method: HTTP GET

Syntax:

```
http://<server_address>/Interface/Cameras/IO/GetInputEvents?
<argument=value>[&<argument=value>...][&<general argument>...]
```

Arguments:

Argument	Valid values	Description
Camera*	String	Name of the camera
Events	APIMasks	Mask to filter the results. Specify which events must be returned based on the provided masks.

* Mandatory parameter

Example 1: Request the list of input events of a camera with response in text

```
http://192.168.0.1:8601/Interface/Cameras/IO/GetInputEvents?
Camera=Cameral&ResponseFormat=Text
```

Example 2: Request the list of input events of a camera with response in XML

```
http://192.168.0.1:8601/Interface/Cameras/IO/GetInputEvents?
Camera=Cameral&ResponseFormat=XML
```

Example 3: Request the list of input events that starts with letter "A" of a camera with response in XML

```
http://192.168.0.1:8601/Interface/Cameras/IO/GetInputEvents?
Camera=Cameral&Events=A*&ResponseFormat=XML
```

Response:

A list with all of the input events of the specified camera is returned.

HTTP Return: 200 OK

Parameters of return:

Fixed parameters:

Parameter	Type	Description
COUNT	Integer	Total number of input events (Not included in JSON response)

Parameters of the list of events:

Parameter	Type	Description
NAME	String	Name of the event
DESCRIPTION	String	Description of the event
LATITUDE	APILatLang	Event latitude with 6 digit precision
LONGITUDE	APILatLang	Event longitude with 6 digit precision
ACTIONS	Event Actions	List of configured Event Actions. Please refer to Event Actions section

Parameter	Type	Description
		for the format of the list.

Example of return in text:

```
RESPONSE_CODE=0
RESPONSE_MESSAGE=OK
COUNT=2
EVENT_1_NAME=Event1
EVENT_1_DESCRIPTION=Event 1
EVENT_1_LATITUDE=48.873501
EVENT_1_LONGITUDE=54.266785
EVENT_1_ACTIONS_SENDEMAIL_ACTIVE=TRUE
EVENT_1_ACTIONS_SENDEMAIL_EMAILGROUP=Administration
EVENT_1_ACTIONS_SENDEMAIL_MESSAGETEXT=
EVENT_1_ACTIONS_SENDEMAIL_SMSFORMAT=Default
EVENT_2_NAME=Event2
EVENT_2_DESCRIPTION=Event 2
EVENT_2_LATITUDE=0.000000
EVENT_2_LONGITUDE=0.000000
EVENT_2_ACTIONS_CALLPRESET_ACTIVE=TRUE
EVENT_2_ACTIONS_CALLPRESET_PRESETS_COUNT=1
EVENT_2_ACTIONS_CALLPRESET_PRESETS_PRESET_1_CAMERA=29
EVENT_2_ACTIONS_CALLPRESET_PRESETS_PRESET_1_PRESETNUMBER=4
```

Example of return in XML:

```
<Response>
  <Code>0</Code>
  <Message>OK</Message>
  <Data>
    <Events>
      <Count>2</Count>
      <Event>
        <Name>Event1</Name>
        <Description>Event 1</Description>
        <Latitude>48.873501</Latitude>
        <Longitude>54.266785</Longitude>
        <Actions>
          <SendEmail>
            <Active>TRUE</Active>
            <EmailGroup>Administration</EmailGroup>
            <MessageText />
            <SMSFormat>Default</SMSFormat>
          </SendEmail>
        </Actions>
      </Event>
      <Event>
        <Name>Event2</Name>
        <Description>Event 2</Description>
        <Latitude>0.000000</Latitude>
        <Longitude>0.000000</Longitude>
        <Actions>
          <CallPreset>
            <Active>TRUE</Active>
            <Presets>
              <Count>1</Count>
              <Preset>
                <Camera>29</Camera>
                <PresetNumber>4</PresetNumber>
              </Preset>
            </Presets>
          </CallPreset>
        </Actions>
      </Event>
    </Events>
  </Data>
</Response>
```

Example of return in JSON:

```
{
  "Response": {
    "Code": 0,
    "Message": "OK",
    "Data": {
      "Events": [
        {
          "Name": "Event1",
          "Description": "Event 1",
          "Latitude": "48.873501",
          "Longitude": "54.266785",
          "Actions": {
            "SendEmail": {
              "Active": true,
              "EmailGroup": "Administration",
              "MessageText": "",
              "SMSFormat": "Default"
            }
          }
        },
        {
          "Name": "Event2",
          "Description": "Event 2",
          "Latitude": "0.000000",
          "Longitude": "0.000000",
          "Actions": {
            "CallPreset": {
              "Active": true,
              "Presets": [
                {
                  "Camera": "29",
                  "PresetNumber": 4
                }
              ]
            }
          }
        }
      ]
    }
  }
}
```

4.3.13.2 Requesting the status of the input ports

Requests the status of the alarm input ports of a camera.

Compatibility: Standard, Professional, Enterprise

Security level: Requires user authentication with rights to camera status

Method: HTTP GET

Syntax:

```
http://<server_address>/Interface/Cameras/IO/GetInputPortStatus?  
<argument=value>[&<argument=value>...][&<general argument>...]
```

Arguments:

Argument	Valid values	Description
Camera*	String	Name of the camera

* Mandatory parameters

Example 1: Request the status of the alarm input ports of a camera with response in text

```
http://192.168.0.1:8601/Interface/Cameras/IO/GetInputPortStatus?  
Camera=Cameral&ResponseFormat=Text
```

Example 2: Request the status of the alarm input ports of a camera with response in XML

```
http://192.168.0.1:8601/Interface/Cameras/IO/GetInputPortStatus?  
Camera=Cameral&ResponseFormat=XML
```

Response:

A list with the status of all of the alarm input ports of the specified camera is returned.

HTTP Return: 200 OK

Parameters of return:**Fixed Parameters:**

Parameter	Type	Description
COUNT	Integer	Total number of alarm input ports (Not included in JSON response)

Parameters of the list of ports:

Parameter	Type	Description
STATE	String	State of the input port
Type		Description
SHORT		Port closed
OPEN		Port open
UNKNOWN		State unknown

Example of return in text:

```
RESPONSE_CODE=0  
RESPONSE_MESSAGE=OK  
COUNT=4  
PORT_1_STATE=OPEN  
PORT_2_STATE=OPEN  
PORT_3_STATE=SHORT  
PORT_4_STATE=OPEN
```

Example of return in XML:

```
<Response>
  <Code>0</Code>
  <Message>OK</Message>
  <Data>
    <Ports>
      <Count>4</Count>
      <Port>
        <State>OPEN</State>
      </Port>
      <Port>
        <State>OPEN</State>
      </Port>
      <Port>
        <State>SHORT</State>
      </Port>
      <Port>
        <State>OPEN</State>
      </Port>
    </Ports>
  </Data>
</Response>
```

Example of return in JSON:

```
{
  "Response": {
    "Code": 0,
    "Message": "OK",
    "Data": {
      "Ports": [
        {
          "State": "OPEN"
        },
        {
          "State": "OPEN"
        },
        {
          "State": "SHORT"
        },
        {
          "State": "OPEN"
        }
      ]
    }
  }
}
```

4.3.13.3 Requesting the status of input events

Requests the status of alarm input events of a camera.

Compatibility: Standard, Professional, Enterprise

Security level: Requires user authentication with rights to camera status

Method: HTTP GET**Syntax:**

```
http://<server_address>/Interface/Cameras/IO/GetInputEventStatus?
<argument=value>[&<argument=value>...][&<general\_argument>...]
```

Arguments:

Argument	Valid values	Description
Camera*	String	Name of the camera

* Mandatory parameters

Example 1: Request the status of alarm input events of camera with response in text

```
http://192.168.0.1:8601/Interface/Cameras/IO/GetInputEventStatus?
Camera=Cameral&ResponseFormat=Text
```

Example 2: Request the status of the alarm input events of camera with response in XML

```
http://192.168.0.1:8601/Interface/Cameras/IO/GetInputEventStatus?
Camera=Cameral&ResponseFormat=XML
```

Response:

A list with the status of all of the alarm input events of the specified camera is returned.

HTTP Return: 200 OK**Parameters of return:****Fixed parameters:**

Parameter	Type	Description
COUNT	Integer	Total number of alarm input events (Not included in JSON response)

Parameters of the list of events:

Parameter	Type	Description						
NAME	String	Name of the event						
STATE	String	State of the input event <table border="1" style="margin-left: 20px;"> <thead> <tr> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>ACTIVE</td> <td>Event active (Occurring at the moment)</td> </tr> <tr> <td>INACTIVE</td> <td>Event inactive (Not occurring)</td> </tr> </tbody> </table>	Type	Description	ACTIVE	Event active (Occurring at the moment)	INACTIVE	Event inactive (Not occurring)
Type	Description							
ACTIVE	Event active (Occurring at the moment)							
INACTIVE	Event inactive (Not occurring)							

Example of return in text:

```
RESPONSE_CODE=0
RESPONSE_MESSAGE=OK
COUNT=2
EVENT_1_NAME=Event 1
EVENT_1_STATE=ACTIVE
EVENT_2_NAME=Event 2
EVENT_2_STATE=INACTIVE
```

Example of return in XML:

```

<Response>
  <Code>0</Code>
  <Message>OK</Message>
  <Data>
    <Events>
      <Count>2</Count>
      <Event>
        <Name>Event 1</Name>
        <State>ACTIVE</State>
      </Event>
      <Event>
        <Name>Event 2</Name>
        <State>INACTIVE</State>
      </Event>
    </Events>
  </Data>
</Response>

```

Example of return in JSON:

```
{
  "Response": {
    "Code": 0,
    "Message": "OK",
    "Data": {
      "Events": [
        {
          "Name": "Event 1",
          "State": "ACTIVE"
        },
        {
          "Name": "Event 2",
          "State": "INACTIVE"
        }
      ]
    }
  }
}
```

4.3.13.4 Requesting the status of output ports

Requests the status of the alarm output ports of a camera.

Compatibility: Standard, Professional, Enterprise

Security level: Requires user authentication with rights to camera status

Method: HTTP GET

Syntax:

```
http://<server_address>/Interface/Cameras/IO/GetOutputPortStatus?
<argument=value>[&<argument=value>... ] [&<general_argument>... ]
```

Arguments:

Argument	Valid values	Description
Camera*	String	Name of the camera

* Mandatory parameters

Example 1: Request the status of the alarm output ports of a camera with response in text

```
http://192.168.0.1:8601/Interface/Cameras/IO/GetOutputPortStatus?
Camera=Cameral&ResponseFormat=Text
```

Example 2: Request the status of alarm output ports of a camera with response in XML

```
http://192.168.0.1:8601/Interface/Cameras/IO/GetOutputPortStatus?
Camera=Cameral&ResponseFormat=XML
```

Response:

A list with the status of all of the alarm output ports of the specified camera is returned.

HTTP Return: 200 OK

Parameters of return:

Fixed parameters:

Parameter	Type	Description
COUNT	Integer	Total number of alarm output ports (Not included in JSON response)

Parameters of the list of ports:

Parameter	Type	Description								
STATE	String	State of the output port <table border="1" data-bbox="693 1108 1330 1235"> <thead> <tr> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>SHORT</td> <td>Port closed</td> </tr> <tr> <td>OPEN</td> <td>Port open</td> </tr> <tr> <td>UNKNOWN</td> <td>State unknown</td> </tr> </tbody> </table>	Type	Description	SHORT	Port closed	OPEN	Port open	UNKNOWN	State unknown
Type	Description									
SHORT	Port closed									
OPEN	Port open									
UNKNOWN	State unknown									

Example of return in text:

```
RESPONSE_CODE=0
RESPONSE_MESSAGE=OK
COUNT=4
PORT_1_STATE=OPEN
PORT_2_STATE=OPEN
PORT_3_STATE=SHORT
PORT_4_STATE=OPEN
```

Example of return in XML:

```
<Response>
  <Code>0</Code>
  <Message>OK</Message>
  <Data>
    <Ports>
      <Count>4</Count>
      <Port>
        <State>OPEN</State>
      </Port>
      <Port>
        <State>OPEN</State>
      </Port>
      <Port>
        <State>SHORT</State>
      </Port>
      <Port>
        <State>OPEN</State>
      </Port>
    </Ports>
  </Data>
</Response>
```

Example of return in JSON:

```
{
  "Response": {
    "Code": 0,
    "Message": "OK",
    "Data": {
      "Ports": [
        {
          "State": "OPEN"
        },
        {
          "State": "OPEN"
        },
        {
          "State": "SHORT"
        },
        {
          "State": "OPEN"
        }
      ]
    }
  }
}
```

4.3.13.5 Requesting the list of output actions

Requests the list of output actions of a camera.

Compatibility: Standard, Professional, Enterprise

Security level: Requires user authentication with rights to camera registration

Method: HTTP GET**Syntax:**

```
http://<server_address>/Interface/Cameras/IO/GetOutputActions?
<argument=value>[&<argument=value>...][&<general\_argument>...]
```

Arguments:

Argument	Valid values	Description
Camera*	String	Name of the camera
Actions	APIMasks	Mask to filter the results. Specify which actions must be returned based on the provided masks.

* Mandatory parameter

Example 1: Request the list of output actions of a camera with response in text

```
http://192.168.0.1:8601/Interface/Cameras/IO/GetOutputActions?
Camera=Cameral&ResponseFormat=Text
```

Example 2: Request the list of output actions of a camera with response in XML

```
http://192.168.0.1:8601/Interface/Cameras/IO/GetOutputActions?
Camera=Cameral&ResponseFormat=XML
```

Example 3: Request the list of output actions that starts with letter "A" of a camera with response in XML

```
http://192.168.0.1:8601/Interface/Cameras/IO/GetOutputActions?
Camera=Cameral&Actions=A*&ResponseFormat=XML
```

Response:

A list with all of the output actions of the specified camera is returned.

HTTP Return: 200 OK

Parameters of return:**Fixed parameters:**

Parameter	Type	Description
COUNT	Integer	Total number of output actions (Not included in JSON response)

Parameters of the list of actions:

Parameter	Type	Description
NAME	String	Name of the action

Example of return in text:

```
RESPONSE_CODE=0
RESPONSE_MESSAGE=OK
COUNT=2
ACTION_1_NAME=Action 1
ACTION_2_NAME=Action 2
```

Example of return in XML:

```
<Response>
  <Code>0</Code>
  <Message>OK</Message>
  <Data>
    <Actions>
      <Count>2</Count>
      <Action>
        <Name>Action 1</Name>
      </Action>
      <Action>
        <Name>Action 2</Name>
      </Action>
    </Actions>
  </Data>
</Response>
```

Example of return in JSON:

```
{
  "Response": {
    "Code": 0,
    "Message": "OK",
    "Data": {
      "Actions": [
        {
          "Name": "Action 1"
        },
        {
          "Name": "Action 2"
        }
      ]
    }
  }
}
```

4.3.13.6 Triggering an output action

By way of this command you will be able to trigger a script of output actions.

Compatibility: Standard, Professional, Enterprise

Security level: Requires user authentication with rights to camera registration

Method: HTTP GET

Syntax:

```
http://<server_address>/Interface/Cameras/IO/TriggerOutputAction?
<argument=value>[&<argument=value>...][&<general\_argument>...]
```

Arguments:

Argument	Valid values	Description
Camera*	String	Name of the camera
Action*	String	Name of the action

* Mandatory parameters

Example 1: Activate an output action of a camera with response in text

```
http://192.168.0.1:8601/Interface/Cameras/IO/TriggerOutputAction?
Camera=Cameral&Action=Action1&ResponseFormat=Text
```

Example 2: Activate an output action of a camera with response in XML

```
http://192.168.0.1:8601/Interface/Cameras/IO/TriggerOutputAction?
Camera=Cameral&Action=Action1&ResponseFormat=XML
```

Response:

Default response of API.

HTTP Return: 200 OK

Parameters of return: [Default return of API](#)

4.3.13.7 Requesting the status of virtual ports

Requests the status of the virtual ports of a camera.

Compatibility: Standard, Professional, Enterprise

Security level: Requires user authentication with rights to camera status

Method: HTTP GET

Syntax:

```
http://<server_address>/Interface/Cameras/IO/GetVirtualPortStatus?
<argument=value>[&<argument=value>... ] [&<general argument>... ]
```

Arguments:

Argument	Valid values	Description
Camera*	String	Name of the camera

* Mandatory parameters

Example 1: Requests the status of the virtual ports of a camera with response in text

```
http://192.168.0.1:8601/Interface/Cameras/IO/GetVirtualPortStatus?
Camera=Cameral&ResponseFormat=Text
```

Example 2: Requests the status of the virtual ports of a camera with response in XML

```
http://192.168.0.1:8601/Interface/Cameras/IO/GetVirtualPortStatus?
Camera=Cameral&ResponseFormat=XML
```

Example 2: Requests the status of the virtual ports of a camera with response in JSON

```
http://192.168.0.1:8601/Interface/Cameras/IO/GetVirtualPortStatus?
Camera=Cameral&ResponseFormat=JSON
```

Response:

A list with the status of all of the virtual ports of the specified camera is returned.

HTTP Return: 200 OK

Parameters of return:

Fixed parameters:

Parameter	Type	Description
COUNT	Integer	Total number of virtual ports (Not included in JSON response)

Parameters of the list of ports:

Parameter	Type	Description								
STATE	String	State of the virtual port <table border="1" style="margin-left: auto; margin-right: auto;"> <thead> <tr> <th>Type</th><th>Description</th></tr> </thead> <tbody> <tr> <td>SHORT</td><td>Port closed</td></tr> <tr> <td>OPEN</td><td>Port open</td></tr> <tr> <td>UNKNOWN</td><td>State unknown</td></tr> </tbody> </table>	Type	Description	SHORT	Port closed	OPEN	Port open	UNKNOWN	State unknown
Type	Description									
SHORT	Port closed									
OPEN	Port open									
UNKNOWN	State unknown									

Example of return in text:

```
RESPONSE_CODE=0
RESPONSE_MESSAGE=OK
COUNT=4
PORT_1_STATE=OPEN
PORT_2_STATE=OPEN
PORT_3_STATE=SHORT
PORT_4_STATE=OPEN
```

Example of return in XML:

```
<Response>
  <Code>0</Code>
  <Message>OK</Message>
  <Data>
    <Ports>
      <Count>4</Count>
      <Port>
        <State>OPEN</State>
      </Port>
      <Port>
        <State>OPEN</State>
      </Port>
      <Port>
        <State>SHORT</State>
      </Port>
      <Port>
        <State>OPEN</State>
      </Port>
    </Ports>
  </Data>
</Response>
```

Example of return in JSON:

```
{
  "Response": {
    "Code": 0,
    "Message": "OK",
    "Data": {
      "Ports": [
        {
          "State": "OPEN"
        },
        {
          "State": "OPEN"
        },
        {
          "State": "SHORT"
        },
        {
          "State": "OPEN"
        }
      ]
    }
  }
}
```

4.3.13.8 Setting the state of a virtual port

By way of this command you will be set the state a Camera Virtual Port

Compatibility: Standard, Professional, Enterprise

Security level: Requires user authentication with rights to camera registration

Method: HTTP GET

Syntax:

```
http://<server_address>/Interface/Cameras/IO/SetVirtualPortStatus?  
<argument=value>[&<argument=value>...][&<general\_argument>...]
```

Arguments:

Argument	Valid values	Description
Camera*	String	Name of the camera
Port*	Integer	Virtual Port number
State*	State	New state of the Virtual Port

State	Description
SHORT	Port closed
OPEN	Port open

* Mandatory parameters

Example 1: Close Virtual Port 1 of a camera with response in text

```
http://192.168.0.1:8601/Interface/Cameras/IO/SetVirtualPortStatus?  
Camera=Cameral&Port=1&State=SHORT&ResponseFormat=Text
```

Example 2: Open Virtual Port 1 of a camera with response in XML

```
http://192.168.0.1:8601/Interface/Cameras/IO/SetVirtualPortStatus?
Camera=Camera1&Port=1&State=OPEN&ResponseFormat=Text
```

Response:

Default response of API.

HTTP Return: 200 OK

Parameters of return: [Default return of API](#)

4.3.14 Motion detection

4.3.14.1 Notifying motion detection

Notify movement from a camera.

Note: This command must be used directly by cameras which support motion detection notification by HTTP

Tip: To better understand the mechanism of camera motion detection notification, consult the document "Using Hardware Motion Detection"

Compatibility: All editions

Security level: Use the camera username and password configured in its register on the system

Method: HTTP GET

Syntax:

```
http://<server_address>/Interface/Cameras/MotionDetection/Notify?
<argument=value>[&<argument=value>...][&<general_argument>...]
```

Arguments:

Arguments	Valid values	Description								
Camera*	String	Camera name								
Motion	Start End Instant	Type of motion detection notification <table border="1" data-bbox="734 1425 1321 1573"> <thead> <tr> <th>Name</th><th>Description</th></tr> </thead> <tbody> <tr> <td>Start</td><td>Motion started</td></tr> <tr> <td>End</td><td>Motion ended</td></tr> <tr> <td>Instant</td><td>Motion (Must be sent constantly)</td></tr> </tbody> </table> If this parameter is omitted, the default value INSTANT will be used	Name	Description	Start	Motion started	End	Motion ended	Instant	Motion (Must be sent constantly)
Name	Description									
Start	Motion started									
End	Motion ended									
Instant	Motion (Must be sent constantly)									

* Mandatory fields

Usage:

There are two ways of using this command: By using Motion=Start and Motion=End parameters and by using Motion=Instant parameter.

Motion=Start and Motion=End parameters must be used by cameras that notifies the start and end of motion. When the system receives Motion=Start parameter, it will start the camera recording and will keep recording it until Motion=End parameter is received, therefore if the system doesn't receive

Motion=End parameter, the recording will not stop.

Motion=Instant, on the other hand, will start the camera recording and will stop it as soon as the post alarm buffer (configured on system) is full, therefore the cameras that doesn't notify the start and end of motion, must notify motion by using this parameter on a frequency not inferior than the post alarm buffer size.

Example 1: Notify motion detection on camera Camera1, using its access credentials ROOT and PASS, with response in XML

```
http://192.168.0.1:8601/Interface/Cameras/MotionDetection/Notify?
Camera=Camera1&AuthUser=root&AuthPass=pass&ResponseFormat=XML
```

Example 2: Notify the start of motion on camera Camera1, using its access credentials ROOT and PASS, with response in text

```
http://192.168.0.1:8601/Interface/Cameras/MotionDetection/Notify?
Camera=Camera1&Motion=Start&AuthUser=root&AuthPass=pass&ResponseFormat=Text
```

Example 3: Notify the end of motion on camera Camera1, using its access credentials ROOT and PASS, with response in XML

```
http://192.168.0.1:8601/Interface/Cameras/MotionDetection/Notify?
Camera=Camera1&Motion=End&AuthUser=root&AuthPass=pass&ResponseFormat=XML
```

Response:

Default response of API.

HTTP Return: 200 OK

Parameters of return: [Default return of API](#)

4.3.15 Manual events

4.3.15.1 Requesting the list of manual events

Requests the list of global events over which the user has rights to access.

Compatibility: Standard, Professional, Enterprise

Security level: Requires user authentication with rights to trigger manual events

Method: HTTP GET

Syntax:

```
http://<server_address>/Interface/Cameras/ManualEvents/GetManualEvents
[?<argument=value> [&<argument=value>...]] [&<general\_argument>...]
```

Arguments:

Argument	Valid values	Description
Camera *	String	Camera name
ManualEvents	APIMasks	Mask to filter the results. Specify which manual events must be returned based on the provided masks.
Fields	Name Description Actions	Specifies the list of desired fields. If this parameter is omitted, all of the fields will be sent.

		The fields must be separated by commas								
<table border="1"> <thead> <tr> <th>Name</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>Name</td> <td>Name of the manual event</td> </tr> <tr> <td>Description</td> <td>Description of the manual event</td> </tr> <tr> <td>Actions</td> <td>List of configured actions</td> </tr> </tbody> </table>			Name	Description	Name	Name of the manual event	Description	Description of the manual event	Actions	List of configured actions
Name	Description									
Name	Name of the manual event									
Description	Description of the manual event									
Actions	List of configured actions									

Example 1: Request the list of manual events from camera 1 with all of the fields and response in XML
 http://192.168.0.1:8601/Interface/Cameras/ManualEvents/GetManualEvents?
 Camera=01&ResponseFormat=XML

Example 2: Request the list of manual events from camera 1 with all of the fields and response in text
 http://192.168.0.1:8601/Interface/Cameras/ManualEvents/GetManualEvents?
 Camera=01&ResponseFormat=Text

Example 3: Request the list of manual events from camera 2 with only name, response in XML and authentication of the user Admin

http://192.168.0.1:8601/Interface/Cameras/ManualEvents/GetManualEvents?
 Camera=02&Fields=Name&ResponseFormat=XML&AuthUser=admin&AuthPass=pass

Example 4: Request the list of manual events starting with letter "A" from camera 2, with only name, response in XML and authentication with Admin user

http://192.168.0.1:8601/Interface/Cameras/ManualEvents/GetManualEvents?
 Camera=02&ManualEvents=A*&Fields=Name&ResponseFormat=XML&AuthUser=admin&
 AuthPass=pass

Response:

A list with all of the manual events from the specified camera is returned. The fields returned in the list will depend on the values informed in the argument **Fields**

HTTP Return: 200 OK

Parameters of return:

Fixed parameters:

Parameter	Type	Description
COUNT	Integer	Total number of manual events

Parameters of the list of manual events:

Parameter	Type	Description
NAME	String	Name of the manual event
DESCRIPTION	String	Description of the manual event
ACTIONS	Event Actions	List of configured Event Actions. Please refer to Event Actions section for the format of the list.

Example of return in text:

```
RESPONSE_CODE=0
RESPONSE_MESSAGE=OK
COUNT=2
MANUALEVENT_1_NAME=Event1
MANUALEVENT_1_DESCRIPTION=Event 1
MANUALEVENT_1_ACTIONS_SENDEMAIL_ACTIVE=TRUE
MANUALEVENT_1_ACTIONS_SENDEMAIL_EMAILGROUP=Administration
MANUALEVENT_1_ACTIONS_SENDEMAIL_MESSAGETEXT=
MANUALEVENT_1_ACTIONS_SENDEMAIL_SMSFORMAT=Default
MANUALEVENT_2_NAME=Event2
MANUALEVENT_2_DESCRIPTION=Event 2
MANUALEVENT_2_ACTIONS_CALLPRESET_ACTIVE=TRUE
MANUALEVENT_2_ACTIONS_CALLPRESET_PRESETS_COUNT=1
MANUALEVENT_2_ACTIONS_CALLPRESET_PRESETS_PRESET_1_CAMERA=29
MANUALEVENT_2_ACTIONS_CALLPRESET_PRESETS_PRESET_1_PRESETNUMBER=4
```

Example of return in XML:

```
<Response>
  <Code>0</Code>
  <Message>OK</Message>
  <Data>
    <ManualEvents>
      <Count>2</Count>
      <ManualEvent>
        <Name>Event1</Name>
        <Description>Event 1</Description>
        <Actions>
          <SendEmail>
            <Active>TRUE</Active>
            <EmailGroup>Administration</EmailGroup>
            <MessageText />
            <SMSFormat>Default</SMSFormat>
          </SendEmail>
        </Actions>
      </ManualEvent>
      <ManualEvent>
        <Name>Event2</Name>
        <Description>Event 2</Description>
        <Actions>
          <CallPreset>
            <Active>TRUE</Active>
            <Presets>
              <Count>1</Count>
              <Preset>
                <Camera>29</Camera>
                <PresetNumber>4</PresetNumber>
              </Preset>
            </Presets>
          </CallPreset>
        </Actions>
      </ManualEvent>
    </ManualEvents>
  </Data>
</Response>
```

Example of return in JSON:

```
{
  "Response": {
    "Code": 0,
    "Message": "OK",
    "Data": {
      "ManualEvents": [
        {
          "Name": "Event1",
          "Description": "Event 1",
          "Actions": {
            "SendEmail": {
              "Active": true,
              "EmailGroup": "Administration",
              "MessageText": "",
              "SMSFormat": "Default"
            }
          }
        },
        {
          "Name": "Event2",
          "Description": "Event 2",
          "Actions": {
            "CallPreset": {
              "Active": true,
              "Presets": [
                {
                  "Camera": "29",
                  "PresetNumber": 4
                }
              ]
            }
          }
        }
      ]
    }
  }
}
```

4.3.15.2 Triggering a manual event

By way of this command, you will be able to trigger a manual event from a camera.

Compatibility: Standard, Professional, Enterprise

Security level: Requires user authentication with rights to trigger manual events

Method: HTTP GET

Syntax:

```
http://<server_address>/Interface/Cameras/ManualEvents/TriggerManualEvent?
<argument=value>[&<argument=value>... ] [&<general argument>... ]
```

Arguments:

Argument	Valid values	Description						
Camera*	String	Name of the camera						
Event*	String	Name of the global event						
OverrideEmailMessage	String	Overrides an e-mail message, in case the action of e-mail transmission for the event is selected						
OverrideEmailMessageFormat	text/plain text/base64	Select the format of the message being passed on OverrideEmailMessage parameter. The following options are available: <table border="1" style="margin-left: auto; margin-right: auto;"> <thead> <tr> <th>Name</th><th>Description</th></tr> </thead> <tbody> <tr> <td>text/plain</td><td>Message will be passed as plain text</td></tr> <tr> <td>text/base64</td><td>Message will be passed using base64 encoding</td></tr> </tbody> </table> If this parameter is omitted, the value text/plain will be used as default	Name	Description	text/plain	Message will be passed as plain text	text/base64	Message will be passed using base64 encoding
Name	Description							
text/plain	Message will be passed as plain text							
text/base64	Message will be passed using base64 encoding							
OverrideOperatorMessage	String	Overrides an instant message sent to the operators, in case the action of message transmission to the operators is selected						
OverrideOperatorMessageFormat	text/plain text/base64	Select the format of the message being passed on OverrideOperatorMessage parameter. The following options are available: <table border="1" style="margin-left: auto; margin-right: auto;"> <thead> <tr> <th>Name</th><th>Description</th></tr> </thead> <tbody> <tr> <td>text/plain</td><td>Message will be passed as plain text</td></tr> <tr> <td>text/base64</td><td>Message will be passed using base64 encoding</td></tr> </tbody> </table> If this parameter is omitted, the value text/plain will be used as default	Name	Description	text/plain	Message will be passed as plain text	text/base64	Message will be passed using base64 encoding
Name	Description							
text/plain	Message will be passed as plain text							
text/base64	Message will be passed using base64 encoding							
OverrideShowObjects	String	Overrides the list of objects to be sent to the operator inside an alarm popup. See below how the string must be generated.						
OverrideShowPlaybackLoop	String	Overrides the list of playback cameras to be sent to the operator inside an alarm popup. See below how the string must be generated.						
OverrideShowSnapshots	String	Overrides the list of snapshots to be sent to the operator inside an alarm popup. See below how the string must be generated.						
OverrideLatitude	APILatLng	Override the latitude from the event and provide a new latitude value.						
OverrideLongitude	APILatLng	Override the longitude from the event and provide a new longitude value.						

* Mandatory parameters

OverrideShowObjects

The list of objects to must be formated in the following way:

```
OBJECT_ID_1;OBJECT_NAME_1,OBJECT_ID_2;OBJECT_NAME_2
```

Being, OBJECT_ID the ID of the object and OBJECT_NAME the name of the objects. The below table displays the possible values for OBJECT_ID:

OBJECT_ID	Description
1	Camera
9	Map
12	Analytics configuration
13	LPR configuration

For example, to show cameras Camera1, Camera2 and the maps Map1 and Map2, the list must be defined as:

```
1;Camera1,1;Camera2,9;Map1,9;Map2
```

OverrideShowSnapshots

This parameter will override the list of objects for snapshots. It uses the same structure as OverrideShowObjects but only supports Camera Object ID at this moment.

The list of objects to must be formated in the following way:

```
OBJECT_ID_1;OBJECT_NAME_1,OBJECT_ID_2;OBJECT_NAME_2
```

Being, OBJECT_ID the ID of the object and OBJECT_NAME the name of the objects. The below table displays the possible values for OBJECT_ID:

OBJECT_ID	Description
1	Camera

OverrideShowPlaybackLoop

This parameter will override the list of cameras for playback loop. It uses the same structure as OverrideShowObjects but only supports Camera Object ID at this moment.

The list of objects to must be formated in the following way:

```
OBJECT_ID_1;OBJECT_NAME_1,OBJECT_ID_2;OBJECT_NAME_2
```

Being, OBJECT_ID the ID of the object and OBJECT_NAME the name of the objects. The below table displays the possible values for OBJECT_ID:

OBJECT_ID	Description
1	Camera

For example, to show snapshots from Camera1 and Camera2 the list must be defined as:

```
1;Camera1,1;Camera2
```

Example 1: Trigger the manual event "Event1" of camera 1 with response in text

```
http://192.168.0.1:8601/Interface/Cameras/ManualEvents/TriggerManualEvent?
Camera=01&Event=Event1&ResponseFormat=Text
```

Example 2: Trigger the manual event "Event2" of camera 1 with response in XML

```
http://192.168.0.1:8601/Interface/Cameras/ManualEvents/TriggerManualEvent?
Camera=01&Event=Event2&ResponseFormat=XML
```

Example 3: Trigger the manual event "Event1" of camera 2 with response in XML and authentication with user Admin

```
http://192.168.0.1:8601/Interface/Cameras/ManualEvents/TriggerManualEvent?
Camera=02&Event=Event1&ResponseFormat=XML&AuthUser=admin&AuthPass=pass
```

Example 4: Trigger the manual event "Event1" of camera 2 overriding the message to be sent to the operators with the message "External alarm" and overriding the objects to be displayed by Camera1, Camera2, Map1, Map2, with response in XML and authentication with user Admin

```
http://192.168.0.1:8601/Interface/Cameras/ManualEvents/TriggerManualEvent?
Camera=02&Event=Event1&OverrideOperatorMessage=External alarm&
OverrideShowObjects=1;Camera1,1;Camera2,9;Map1,9;Map2&
ResponseFormat=XML&AuthUser=admin&AuthPass=pass
```

Response:

Default response of API.

HTTP Return: 200 OK

Parameters of return: [Default return of API](#)

4.3.16 Bookmarks

4.3.16.1 Adding a new bookmark

Create a new bookmark in camera recordings

Compatibility: Professional, Enterprise

Security level: Requires user authentication with rights to create bookmarks

Method: HTTP GET

Syntax:

```
http://<server_address>/Interface/Cameras/Bookmarks/Add?
<argument=value>[&<argument=value>...][&<general argument>...]
```

Arguments:

Argument	Valid values	Description
Title *	String	Bookmark title
Color *	APIColor	Bookmark color
StartDate*	APIDate	Bookmark start date
StartTime*	APITime	Bookmark start time
Cameras *	String	List of cameras that will be part of the new bookmark. Camera names must be comma-separated.
EndDate	APIDate	Bookmark end date. If omitted, the start date will be used
EndTime	APITime	Bookmark end time. If omitted, the start time will be used
Remarks	String	Bookmark remarks

* Mandatory parameters

Example 1: Add a punctual bookmark (Containing only start date and time) "Alarm" of red color, with start date and time of March 07, 2014 15:48:34.450 for cameras "Camera1" and "Camera2"

```
http://192.168.0.1:8601/Interface/Cameras/Bookmarks/Add?Title=Alarm&
Color=Red&StartDate=2014.03.07&StartTime=15.58.34.450&
Cameras=Camera1, Camera2
```

Example 2: Add a ranged bookmark with title "Door open" of blue color with start date and time of March 07, 2014 16:00:04.510 and final date and time of March 07, 2014 16:10:23.100 for cameras "Camera1" and "Camera2" and remark "Front door is open"

```
http://192.168.0.1:8601/Interface/Cameras/Bookmarks/Add?Title=Door%20open&
Color=Blue&StartDate=2014.03.07&StartTime=16.00.04.510&
EndDate=2014.03.07&EndTime=16.10.23.100&Cameras=Camera1,Camera2&
Remarks=Front%20door%20is%20open
```

Response:

Default response of API.

HTTP Return: 200 OK

Parameters of return: [Default return of API](#)

4.3.16.2 Searching bookmarks

Perform a search for bookmarks in the database

Compatibility: Professional, Enterprise

Security level: Requires user authentication with rights to search for bookmarks

Method: HTTP GET

Syntax:

```
http://<server_address>/Interface/Cameras/Bookmarks/Search
[?<argument=value> [&<argument=value>...]] [&<general\_argument>...]]
```

Arguments:

Argument	Valid values	Description
Keyword	String	Keyword to search on bookmarks
KeywordExact	TRUE FALSE	TRUE - Search for exact keyword only FALSE - Search for keyword contained inside bookmark If omitted, the default value FALSE will be used
SearchRemarks	TRUE FALSE	Search in remarks as well (Instead of only searching by title) If omitted, the default value FALSE will be used Note: Search might be slower when activating this parameter
Colors	APIColor	Search for bookmarks of the specified colors only. You can specify multiple colors. In this case, the color names must be comma-separated.
StartDate	APIDate	Start date for date / time range
StartTime	APITime	Start time for date / time range
EndDate	APIDate	End date for date / time range. If this parameter is omitted, the value specified in <code>StartTime</code> parameter will be used
EndTime	APITime	End time for date / time range. If this parameter is omitted, the value specified in <code>StartTime</code> parameter will be used
Cameras	String	List of cameras from bookmark. The list with camera names must be comma-separated.

Example 1: Search for all bookmarks that contains the word "Alarm"

```
http://192.168.0.1:8601/Interface/Cameras/Bookmarks/Search?Keyword=Alarm
```

Example 2: Search for all red and blue bookmarks

```
http://192.168.0.1:8601/Interface/Cameras/Bookmarks/Search?Color=Red,Blue
```

Example 3: Search for all red bookmarks of camera "Camera1" that has the keyword "Alarm", with response in text

```
http://192.168.0.1:8601/Interface/Cameras/Bookmarks/Search?Keyword=Alarm&Cameras=Camera1&Color=Red&ResponseFormat=Text
```

Response:

A list with all found bookmarks will be returned.

HTTP Return: 200 OK

Return parameters:

Fixed parameters:

Parameter	Type	Description
COUNT	Integer	Bookmark count (Not included in JSON response)

Bookmark list parameters:

Parameter	Type	Description
TITLE	String	Bookmark title
COLOR	APIColor	Bookmark color
STARTDATE	APITimestamp	Bookmark start date and time
ENDDATE	APITimestamp	Bookmark end date and time
REMARKS	String	Bookmark remarks
USER	String	User that created the bookmark
CAMERAS	String	List of cameras of which the bookmark belongs to (Comma-separated)

Example of result in text format:

```
RESPONSE_CODE=0
RESPONSE_MESSAGE=OK
COUNT=2
BOOKMARK_1_TITLE=Test
BOOKMARK_1_COLOR=Red
BOOKMARK_1_STARTDATE=2014-06-03 10:10:01.302
BOOKMARK_1_ENDDATE=2014-06-03 10:10:01.302
BOOKMARK_1_REMARKS=
BOOKMARK_1_USER=admin
BOOKMARK_1_CAMERAS=Camera1
BOOKMARK_2_TITLE=Alarm
BOOKMARK_2_COLOR=Red
BOOKMARK_2_STARTDATE=2014-03-07 15:58:34.450
BOOKMARK_2_ENDDATE=2014-03-07 15:58:34.450
BOOKMARK_2_REMARKS=
BOOKMARK_2_USER=admin
BOOKMARK_2_CAMERAS=Camera1, Camera2
```

Example of return in XML:

```
<Response>
  <Code>0</Code>
  <Message>OK</Message>
  <Data>
    <Bookmarks>
      <Count>2</Count>
      <Bookmark>
        <Title>Test</Title>
        <Color>Red</Color>
        <StartDate>2014-06-03 10:10:01.302</StartDate>
        <EndDate>2014-06-03 10:10:01.302</EndDate>
        <Remarks/>
        <User>admin</User>
        <Cameras>Camera1</Cameras>
      </Bookmark>
      <Bookmark>
        <Title>Alarm</Title>
        <Color>Red</Color>
        <StartDate>2014-03-07 15:58:34.450</StartDate>
        <EndDate>2014-03-07 15:58:34.450</EndDate>
        <Remarks/>
        <User>admin</User>
        <Cameras>Camera1, Camera2</Cameras>
      </Bookmark>
    </Bookmarks>
  </Data>
</Response>
```

Example of return in JSON:

```
{
  "Response": {
    "Code": 0,
    "Message": "OK",
    "Data": {
      "Bookmarks": [
        {
          "Title": "Test",
          "Color": "Red",
          "StartDate": "2014-06-03T10:10:01.302Z",
          "EndDate": "2014-06-03T10:10:01.302Z",
          "Remarks": "",
          "User": "admin",
          "Cameras": "Camera1"
        },
        {
          "Title": "Alarm",
          "Color": "Red",
          "StartDate": "2014-03-07T15:58:34.450Z",
          "EndDate": "2014-03-07T15:58:34.450Z",
          "Remarks": "",
          "User": "admin",
          "Cameras": "Camera1,Camera2"
        }
      ]
    }
  }
}
```

4.3.17 Privacy Mode

4.3.17.1 Controlling privacy mode

Command to activate or deactivate privacy mode of a given camera

Compatibility: Professional, Enterprise

Security level: Requires authentication of user with rights to control privacy mode

Method: HTTP GET

Syntax:

```
http://<server_address>/Interface/Cameras/PrivacyMode/SetPrivacyMode?
<argument=value>[&<argument=value>... ] [&<general_argument>... ]
```

Arguments:

Argument	Valid values	Description
Camera*	String	Name of the camera
Active *	TRUE FALSE	TRUE - Activate privacy mode FALSE - Deactivate privacy mode

* Mandatory parameters

Example 1: Activate privacy mode of "Camera1"

```
http://192.168.0.1:8601/Interface/Cameras/PrivacyMode/SetPrivacyMode?
Camera=Camera1&Active=TRUE
```

Example 2: Deactivate privacy mode of "Camera1" with response in text

```
http://192.168.0.1:8601/Interface/Cameras/PrivacyMode/SetPrivacyMode?
Camera=Camera1&Active=FALSE&ResponseFormat=Text
```

Response:

Default response of API.

HTTP Return: 200 OK

Parameters of return: [Default return of API](#)

4.3.17.2 Requesting the status of privacy mode

Command to query the current status of privacy mode of a given camera

Compatibility: Professional, Enterprise

Security level: Requires user authentication

Method: HTTP GET

Syntax:

```
http://<server_address>/Interface/Cameras/PrivacyMode/GetStatus?
<argument=value>[&<argument=value>...][&<general\_argument>...]
```

Arguments:

Argument	Valid values	Description
Camera*	String	Name of the camera

* Mandatory parameters

Example 1: Query privacy mode status of "Camera1"

```
http://192.168.0.1:8601/Interface/Cameras/PrivacyMode/GetStatus?
Camera=Camera1
```

Example 2: Query privacy mode status of "Camera1" with response in text

```
http://192.168.0.1:8601/Interface/Cameras/PrivacyMode/GetStatus?
Camera=Camera1&ResponseFormat=Text
```

Response:

A list of parameter-value pairs is returned

HTTP Return HTTP: 200 OK

Parameters of return:

Parameter	Type	Description
ACTIVE	Boolean	Privacy mode activation status

Example of return in text:

```
RESPONSE_CODE=0
RESPONSE_MESSAGE=OK
ACTIVE=FALSE
```

Example of return in XML:

```
<Response>
  <Code>0</Code>
  <Message>OK</Message>
  <Data>
    <Status>
      <Active>FALSE</Active>
    </Status>
  </Data>
</Response>
```

Example of return in JSON:

```
{
  "Response": {
    "Code": 0,
    "Message": "OK",
    "Data": {
      "Status": {
        "Active": false
      }
    }
  }
}
```

4.3.18 Recording

4.3.18.1 Notifying an event for event recording

Command to notify the default recording event when a camera is set to record by event. The system has an internal event that can only be triggered by the API and does not need to be selected in camera recording scheduling when recording by event, this event will start or stop the recording if the camera is set to record by event.

Compatibility: Standard, Professional, Enterprise

Security level: Requires user authentication with rights to Camera Registration

Method: HTTP GET

Syntax:

```
http://<server_address>/Interface/Cameras/Recording/NotifyEvent?
<argument=value>[&<argument=value>...][&<general argument>...]
```

Arguments:

Argument	Valid values	Description
Cameras*	String	List of cameras to notify the recording event. Camera names must be separated by comma
Event *	Start End	The following values are supported for Event parameter:

		Name	Description
		Start	Notify that the recording event has started
		End	Notify that the recording event has ended

* Mandatory parameters

Example 1: Start recording event of camera "Camera1"

```
http://192.168.0.1:8601/Interface/Cameras/Recording/NotifyEvent?
Cameras=Camera1&Event=Start
```

Example 2: End recording event of cameras "Camera1" and "Camera2" with response in text

```
http://192.168.0.1:8601/Interface/Cameras/Recording/NotifyEvent?
Cameras=Camera1,Camera2&Event=End&ResponseFormat=Text
```

Response:

Default response of API.

HTTP Return: 200 OK

Parameters of return: [Default return of API](#)

4.3.18.2 Locks

4.3.18.2.1 Searching Recording Locks

Perform a search for Recording Locks

Compatibility: Professional, Enterprise

Security level: Requires user authentication with rights to Search for Recording Locks

Method: HTTP GET

Syntax:

```
http://<server_address>/Interface/Cameras/Recording/Locks/Search
[?<argument=value>[&<argument=value>...][&<general_argument>...]]
```

Arguments:

Argument	Valid values	Description
StartDate	APIDate	Start date for date / time range
StartTime	APITime	Start time for date / time range
EndDate	APIDate	End date for date / time range
EndTime	APITime	End time for date / time range
Cameras	String	List of cameras. The list with camera names must be comma-separated.
Users	String	List of users. This list with user names must be comma-separated

Example 1: Search for all recording locks

```
http://192.168.0.1:8601/Interface/Cameras/Recording/Locks/Search
```

Example 2: Search for all recording locks from cameras 01 and 02

```
http://192.168.0.1:8601/Interface/Cameras/Recording/Locks/Search?
Cameras=01,02
```

Example 3: Search for all recording locks created by User1 with response in text

```
http://192.168.0.1:8601/Interface/Cameras/Recording/Locks/Search?
Users=User1&ResponseFormat=Text
```

Response:

A list with all found locks will be returned.

HTTP Return: 200 OK

Return parameters:**Fixed parameters:**

Parameter	Type	Description
COUNT	Integer	Recording Lock count (Not included in JSON response)

Recording Lock list parameters:

Parameter	Type	Description
LOCKID	String	ID of the Lock
CAMERA	String	Camera Name
DESCRIPTION	String	Lock Description
STARTDATE	APITimestamp	Lock start date and time
ENDDATE	APITimestamp	Lock end date and time
USER	String	User that created the lock

Example of result in text format:

```
RESPONSE_CODE=0
RESPONSE_MESSAGE=OK
COUNT=2
LOCK_1_LOCKID=F5720523-CDC9-476F-8549-A0A598CAA711
LOCK_1_CAMERA=01
LOCK_1_DESCRIPTION=Suspect of Robbery
LOCK_1_STARTDATE=2019-10-08 19:48:06.000
LOCK_1_ENDDATE=2019-10-08 19:50:00.000
LOCK_1_USER=admin
LOCK_2_LOCKID=CB40BCD8-671D-432E-939B-6E3191805A62
LOCK_2_CAMERA=02
LOCK_2_DESCRIPTION=Car Crash
LOCK_2_STARTDATE=2019-10-08 19:48:38.000
LOCK_2_ENDDATE=2019-10-08 19:55:00.000
LOCK_2_USER=admin
```

Example of return in XML:

```

<Response>
  <Code>0</Code>
  <Message>OK</Message>
  <Data>
    <Locks>
      <Count>2</Count>
      <Lock>
        <LockID>F5720523-CDC9-476F-8549-A0A598CAA711</LockID>
        <Camera>01</Camera>
        <Description>Suspect of Robbery</Description>
        <StartDate>2019-10-08 19:48:06.000</StartDate>
        <EndDate>2019-10-08 19:50:00.000</EndDate>
        <User>admin</User>
      </Lock>
      <Lock>
        <LockID>CB40BCD8-671D-432E-939B-6E3191805A62</LockID>
        <Camera>02</Camera>
        <Description>Car Crash</Description>
        <StartDate>2019-10-08 19:48:38.000</StartDate>
        <EndDate>2019-10-08 19:55:00.000</EndDate>
        <User>admin</User>
      </Lock>
    </Locks>
  </Data>
</Response>

```

Example of return in JSON:

```
{
  "Response": {
    "Code": 0,
    "Message": "OK",
    "Data": {
      "Locks": [
        {
          "LockID": "F5720523-CDC9-476F-8549-A0A598CAA711",
          "Camera": "01",
          "Description": "Suspect of Robbery",
          "StartDate": "2019-10-08T19:48:06.000Z",
          "EndDate": "2019-10-08T19:50:00.000Z",
          "User": "admin"
        },
        {
          "LockID": "CB40BCD8-671D-432E-939B-6E3191805A62",
          "Camera": "02",
          "Description": "Car Crash",
          "StartDate": "2019-10-08T19:48:38.000Z",
          "EndDate": "2019-10-08T19:55:00.000Z",
          "User": "admin"
        }
      ]
    }
  }
}
```

4.3.18.2.2 Adding a new Recording Lock

Create a new recording lock for a camera

Compatibility: Professional, Enterprise

Security level: Requires user authentication with rights to Create Recording Locks

Method: HTTP GET

Syntax:

```
http://<server_address>/Interface/Cameras/Recording/Locks/Add?
<argument=value>[&<argument=value>...][&<general argument>...]
```

Arguments:

Argument	Valid values	Description
Camera*	String	Camera name
Description*	String	Lock description
StartDate*	APIDate	Lock start date
StartTime*	APITime	Lock start time
EndDate*	APIDate	Lock end date
EndTime*	APITime	Lock end time
ExpirationDate	APIDate	Lock expiration date (If applicable)

* Mandatory parameters

Example 1: Add a recording lock for camera 01 with start date and time of October 08, 2019

15:00:00.000 and final date time of October 08, 2019, 16:00:00.000 with no expiration date

```
http://192.168.0.1:8601/Interface/Cameras/Recording/Locks/Add?Camera=01&
Description=Suspect of Robbery&StartDate=2019.10.08&StartTime=15.00.00.0000&
EndDate=2019.10.08&EndTime=16.00.00.000
```

Example 2: Add a recording lock for camera 01 with start date and time of October 08, 2019

15:00:00.000 and final date time of October 08, 2019, 16:00:00.000 with no expiration date for October 29, 2019

```
http://192.168.0.1:8601/Interface/Cameras/Recording/Locks/Add?Camera=01&
Description=Suspect of Robbery&StartDate=2019.10.08&StartTime=15.00.00.0000&
EndDate=2019.10.08&EndTime=16.00.00.0000&ExpirationDate=2019.10.29
```

Response:

If the lock was created successfully, the LockID will be returned, otherwise the [result codes 10600 to 10603](#) will be returned

HTTP Return: 200 OK

Return parameters:

Fixed parameters:

Parameter	Type	Description
LockID	String	ID of the created lock

4.3.18.3 Creating a new Self-Healing Job

Create a new Recording Self-Healing Job

Compatibility: Enterprise

Security level: Requires user authentication with rights to Camera Registration

Method: HTTP GET

Syntax:

```
http://<server_address>/Interface/Cameras/Recording/SelfHealing/
CreateDownloadJob?<argument=value>[&<argument=value>...]
[&<general_argument>...]
```

Arguments:

Argument	Valid values	Description
Camera*	String	Camera name
StartDate*	APIDate	Download period start date
StartTime*	APITime	Download period start time
EndDate*	APIDate	Download period end date
EndTime*	APITime	Download period end time

* Mandatory parameters

Example 1: Add a download job for camera 01 with start date and time of October 08, 2019

15:00:00.000 and final date time of October 08, 2019, 16:00:00.000

```
http://192.168.0.1:8601/Interface/Cameras/Recording/SelfHealing/
CreateDownloadJob?Camera=01&StartDate=2019.10.08&StartTime=15.00.00.000&
EndDate=2019.10.08&EndTime=16.00.00.000
```

Example 2: Add a download job for camera 02 with start date and time of October 09, 2019

16:00:00.000 and final date time of October 09, 2019, 16:30:00.000 with response in JSON

```
http://192.168.0.1:8601/Interface/Cameras/Recording/SelfHealing/
CreateDownloadJob?Camera=02&StartDate=2019.10.09&StartTime=16.00.00.000&
EndDate=2019.10.09&EndTime=16.30.00.000&ResponseFormat=JSON
```

Response:

Default response of API.

HTTP Return: 200 OK

Parameters of return: [Default return of API](#)

4.4 I/O Devices

4.4.1 Requesting the list of I/O devices

Requests the list of I/O devices registered in the server.

Compatibility: Standard, Professional, Enterprise

Security level: Requires user authentication with rights to I/O device registration

Method: HTTP GET

Syntax:

```
http://<server_address>/Interface/IODevices/GetIODevices
[?<argument=value> [&<argument=value>...]] [&<general\_argument>...]
```

Arguments:

Argument	Valid values	Description														
IODevices	APIMasks	Mask to filter the results. Specify which I/O devices must be returned based on the provided masks.														
Fields	Name Description Active Model Latitude Longitude	Specifies the list of desired fields. In case this parameter is omitted, all of the fields will be sent. The fields must be separated by commas <table border="1" data-bbox="677 718 1395 960"> <thead> <tr> <th>Name</th><th>Description</th></tr> </thead> <tbody> <tr> <td>Name</td><td>Name of the I/O device</td></tr> <tr> <td>Description</td><td>Description of the I/O device</td></tr> <tr> <td>Active</td><td>I/O device activated / deactivated</td></tr> <tr> <td>Model</td><td>Model of the I/O device</td></tr> <tr> <td>Latitude</td><td>Configured device latitude</td></tr> <tr> <td>Longitude</td><td>Configured device longitude</td></tr> </tbody> </table>	Name	Description	Name	Name of the I/O device	Description	Description of the I/O device	Active	I/O device activated / deactivated	Model	Model of the I/O device	Latitude	Configured device latitude	Longitude	Configured device longitude
Name	Description															
Name	Name of the I/O device															
Description	Description of the I/O device															
Active	I/O device activated / deactivated															
Model	Model of the I/O device															
Latitude	Configured device latitude															
Longitude	Configured device longitude															

Example 1: Requests the list of I/O devices with all of the fields and response in XML

```
http://192.168.0.1:8601/Interface/IODevices/GetIODevices?
ResponseFormat=XML
```

Example 2: Requests the list of I/O devices with all of the fields and response in text

```
http://192.168.0.1:8601/Interface/IODevices/GetIODevices?
ResponseFormat=Text
```

Example 3: Requests the list of I/O devices with only name and description, response in XML and authentication of the user Admin

```
http://192.168.0.1:8601/Interface/IODevices/GetIODevices?Fields=Name,
Description&ResponseFormat=XML&AuthUser=admin
```

Example 4: Request the list of I/O devices starting with "A", with only name and description, response in XML format and authentication with Admin user

```
http://192.168.0.1:8601/Interface/IODevices/GetIODevices?
IODevices=A*&Fields=Name,Description&ResponseFormat=XML&AuthUser=admin
```

Response:

A list with all of the I/O devices registered in the system is returned. The fields returned in the list will depend on the values informed in the argument **Fields**

HTTP Return: 200 OK

Parameters of return:

Fixed parameters:

Parameter	Type	Description
COUNT	Integer	Total number of I/O devices (Not included in JSON response)

Parameters of the list of I/O devices:

Parameter	Type	Description
NAME	String	Name of the I/O device
DESCRIPTION	String	Description of the I/O device
ACTIVE	Boolean	I/O device activated / deactivated
MODEL	String	Model of the de I/O device
LATITUDE	APILatLng	Device latitude with 6 digit precision
LONGITUDE	APILatLng	Device longitude with 6 digit precision

Example of return in text:

```

RESPONSE_CODE=0
RESPONSE_MESSAGE=OK
COUNT=2
IODEVICE_1_NAME=Device 1
IODEVICE_1_DESCRIPTION=I/O Device 1
IODEVICE_1_ACTIVE=TRUE
IODEVICE_1_MODEL=Generic
IODEVICE_1_LATITUDE=0.000000
IODEVICE_1_LONGITUDE=0.000000
IODEVICE_2_NAME=Device 2
IODEVICE_2_DESCRIPTION=I/O Device 2
IODEVICE_2_ACTIVE=FALSE
IODEVICE_2_MODEL=Generic
IODEVICE_2_LATITUDE=-23.630363
IODEVICE_2_LONGITUDE=-46.554916

```

Example of return in XML:

```

<Response>
  <Code>0</Code>
  <Message>OK</Message>
  <Data>
    <IODevices>
      <Count>2</Count>
      <IODevice>
        <Name>Device 1</Name>
        <Description>I/O device 1</Description>
        <Active>TRUE</Active>
        <Model>Generic</Model>
        <Latitude>0.000000</Latitude>
        <Longitude>0.000000</Longitude>
      </IODevice>
      <IODevice>
        <Name>Device 2</Name>
        <Description>I/O device 2</Description>
        <Active>FALSE</Active>
        <Model>Generic</Model>
        <Latitude>-23.630363</Latitude>
        <Longitude>-46.554916</Longitude>
      </IODevice>
    </IODevices>
  </Data>
</Response>

```

Example of return in JSON:

```

{
  "Response": {
    "Code": 0,
    "Message": "OK",
    "Data": {
      "IODevices": [
        {
          "Name": "Device 1",
          "Description": "I/O device 1",
          "Active": true,
          "Model": "Generic",
          "Latitude": "0.000000",
          "Longitude": "0.000000"
        },
        {
          "Name": "Device 2",
          "Description": "I/O device 2",
          "Active": false,
          "Model": "Generic",
          "Latitude": "-23.630363",
          "Longitude": "-46.554916"
        }
      ]
    }
  }
}

```

4.4.2 Requesting the status of I/O devices

Requests the status of the I/O devices from the system.

Compatibility: Standard, Professional, Enterprise

Security level: Requires user authentication with rights to I/O device status

Method: HTTP GET

Syntax:

```
http://<server_address>/Interface/IODevices/GetStatus [<argument=value>
[&<argument=value>...]] [&<general argument>...]]
```

Arguments:

Argument	Valid values	Description						
IODevices	APIMasks	Mask to filter the results. Specify which I/O devices must be returned based on the provided masks.						
Fields	Active Working	Specifies the list of desired fields. In case this parameter is omitted, all of the fields will be sent. The fields must be separated by commas <table border="1" data-bbox="660 967 1411 1072"> <thead> <tr> <th>Name</th><th>Description</th></tr> </thead> <tbody> <tr> <td>Active</td><td>Identify if the I/O device is active</td></tr> <tr> <td>Working</td><td>Identify if the I/O device is working</td></tr> </tbody> </table>	Name	Description	Active	Identify if the I/O device is active	Working	Identify if the I/O device is working
Name	Description							
Active	Identify if the I/O device is active							
Working	Identify if the I/O device is working							
Active	TRUE FALSE	In case this parameter is specified, a filter will be applied and only the I/O devices that matches this filter will be returned, thus, in case the value Active=TRUE is specified, the result will only include the active I/O devices, while if the value Active=FALSE is specified, the result will only include the deactivated I/O devices.						
Working	TRUE FALSE	In case this parameter is specified, a filter will be applied and only the I/O devices that matches this filter will be returned, thus, in case the value Working=TRUE is specified, the result will only include the working I/O devices, while if the value Working=FALSE is specified, the result will only include the I/O devices that are out of order.						

Example 1: Request the status of all I/O devices with all fields and response in XML

```
http://192.168.0.1:8601/Interface/IODevices/GetStatus?ResponseFormat=XML
```

Example 2: Request the status of all active I/O devices with response in text

```
http://192.168.0.1:8601/Interface/IODevices/GetStatus?Active=TRUE&
ResponseFormat=Text
```

Example 3: Request the status of all active devices starting with "A", with response in text

```
http://192.168.0.1:8601/Interface/IODevices/GetStatus?IODevices=A*&
Active=TRUE&ResponseFormat=Text
```

Example 4: Request the status of all active I/O devices that are not working, with response in XML and authentication with admin user (No password)

```
http://192.168.0.1:8601/Interface/IODevices/GetStatus?Active=TRUE&
```

```
Working=FALSE&ResponseFormat=XML&AuthUser=admin
```

Response:

A list with the status of all of I/O devices is returned. The fields returned in the will depend on the values informed in the argument Fields

HTTP Return: 200 OK

Parameters of return:**Fixed Parameter:**

Parameter	Type	Description
COUNT	Integer	Total number of I/O devices (Not included in JSON response)

Parameters in the list of status of I/O devices:

Parameter	Type	Description
NAME	String	Name of the I/O device
ACTIVE	Boolean	Identify if the I/O device is active
WORKING	Boolean	Identify if the I/O device is working

Example of return in text:

```
RESPONSE_CODE=0
RESPONSE_MESSAGE=OK
COUNT=2
IODEVICE_1_NAME=Device 1
IODEVICE_1_ACTIVE=TRUE
IODEVICE_1_WORKING=TRUE
IODEVICE_2_NAME=Device 2
IODEVICE_2_ACTIVE=TRUE
IODEVICE_2_WORKING=FALSE
```

Example of return in XML:

```
<Response>
  <Code>0</Code>
  <Message>OK</Message>
  <Data>
    <IODevices>
      <Count>2</Count>
      <IODevice>
        <Name>Device 1</Name>
        <Active>TRUE</Active>
        <Working>TRUE</Working>
      </IODevice>
      <IODevice>
        <Name>Device 2</Name>
        <Active>TRUE</Active>
        <Working>FALSE</Working>
      </IODevice>
    </IODevices>
  </Data>
</Response>
```

Example of return in JSON:

```
{
  "Response": {
    "Code": 0,
    "Message": "OK",
    "Data": {
      "IODevices": [
        {
          "Name": "Device 1",
          "Active": true,
          "Working": true
        },
        {
          "Name": "Device 2",
          "Active": true,
          "Working": false
        }
      ]
    }
  }
}
```

4.4.3 Activating / Deactivating I/O devices

Allows the activation or deactivation of multiple I/O devices simultaneously

Compatibility: Standard, Professional, Enterprise

Security level: Requires user authentication with rights to I/O device registration

Method: HTTP GET

Syntax:

```
http://<server_address>/Interface/IODevices/Activation?<argument=value>
[&<argument=value>...] [&<general argument>...]
```

Arguments:

Argument	Valid values	Description										
IODevices*	String	<p>List of devices to activate or deactivate.</p> <p>The device names must be separated by commas.</p> <p>This command is only mandatory when Action=Activate or Action=Deactivate</p>										
Action*	Activate Deactivate ActivateAll DeactivateAll	<p>Action to be executed</p> <table border="1"> <thead> <tr> <th>Operation</th><th>Description</th></tr> </thead> <tbody> <tr> <td>Activate</td><td>Activate the devices on IODevices argument</td></tr> <tr> <td>Deactivate</td><td>Deactivate the devices on IODevices argument</td></tr> <tr> <td>ActivateAll</td><td>Activate all I/O devices from the server</td></tr> <tr> <td>DeactivateAll</td><td>Deactivate all I/O devices from the server</td></tr> </tbody> </table>	Operation	Description	Activate	Activate the devices on IODevices argument	Deactivate	Deactivate the devices on IODevices argument	ActivateAll	Activate all I/O devices from the server	DeactivateAll	Deactivate all I/O devices from the server
Operation	Description											
Activate	Activate the devices on IODevices argument											
Deactivate	Deactivate the devices on IODevices argument											
ActivateAll	Activate all I/O devices from the server											
DeactivateAll	Deactivate all I/O devices from the server											

* Mandatory parameters

Example 1: Activate all I/O devices from the server

```
http://192.168.0.1:8601/Interface/IODevices/Activation?Action=ActivateAll
```

Example 2: Activate I/O devices Device1 and Device2

```
http://192.168.0.1:8601/Interface/IODevices/Activation?Action=Activate&IODevices=Device1,Device2
```

Example 3: Deactivate all I/O devices from the server

```
http://192.168.0.1:8601/Interface/IODevices/Activation?Action=DeactivateAll
```

Response:

Default response of API.

HTTP Return HTTP: 200 OK

Parameters of return: [Default return of API](#)

4.4.4 I/O

4.4.4.1 Requesting the list of input events

Requests the list of input events of an I/O device.

Compatibility: Standard, Professional, Enterprise

Security level: Requires user authentication with rights to I/O device registration

Method: HTTP GET

Syntax:

```
http://<server_address>/Interface/IODevices/IO/GetInputEvents?
<argument=value>[&<argument=value>...][&<general\_argument>...]
```

Arguments:

Argument	Valid values	Description
Device*	String	Name of the device (Not included in JSON response)
Events	APIMasks	Mask to filter the results. Specify which events must be returned based on the provided masks.

* Mandatory parameter

Example 1: Request the list of input events of a device with response in text

```
http://192.168.0.1:8601/Interface/IODevices/IO/GetInputEvents?
Device=Device1&ResponseFormat=Text
```

Example 2: Request the list of input events of a device with response in XML

```
http://192.168.0.1:8601/Interface/IODevices/IO/GetInputEvents?
Device=Device1&ResponseFormat=XML
```

Example 3: Request the list of input events that starts with letter "A" of a device with response in XML

```
http://192.168.0.1:8601/Interface/IODevices/IO/GetInputEvents?
Device=Device1&Events=A*&ResponseFormat=XML
```

Response:

A list with all of the input events of the specified device is returned.

HTTP Return: 200 OK

Parameters of return:**Fixed parameters:**

Parameter	Type	Description
COUNT	Integer	Total number of input events (Not included in JSON response)

Parameters of the list of events:

Parameter	Type	Description
NAME	String	Name of the event
DESCRIPTION	String	Description of the event
LATITUDE	APILatLnG	Event latitude with 6 digit precision
LONGITUDE	APILatLnG	Event longitude with 6 digit precision
ACTIONS	Event Actions	List of configured Event Actions. Please refer to Event Actions section for the format of the list.

Example of return in text:

```
RESPONSE_CODE=0
RESPONSE_MESSAGE=OK
COUNT=2
EVENT_1_NAME=Event1
EVENT_1_DESCRIPTION=Event 1
EVENT_1_LATITUDE=48.873501
EVENT_1_LONGITUDE=54.266785
EVENT_1_ACTIONS_SENDEMAIL_ACTIVE=TRUE
EVENT_1_ACTIONS_SENDEMAIL_EMAILGROUP=Administration
EVENT_1_ACTIONS_SENDEMAIL_MESSAGETEXT=
EVENT_1_ACTIONS_SENDEMAIL_SMSFORMAT=Default
EVENT_2_NAME=Event2
EVENT_2_DESCRIPTION=Event 2
EVENT_2_LATITUDE=0.000000
EVENT_2_LONGITUDE=0.000000
EVENT_2_ACTIONS_CALLPRESET_ACTIVE=TRUE
EVENT_2_ACTIONS_CALLPRESET_PRESETS_COUNT=1
EVENT_2_ACTIONS_CALLPRESET_PRESETS_PRESET_1_CAMERA=29
EVENT_2_ACTIONS_CALLPRESET_PRESETS_PRESET_1_PRESETNUMBER=4
```

Example of return in XML:

```
<Response>
  <Code>0</Code>
  <Message>OK</Message>
  <Data>
    <Events>
      <Count>2</Count>
      <Event>
        <Name>Event1</Name>
        <Description>Event 1</Description>
        <Latitude>48.873501</Latitude>
        <Longitude>54.266785</Longitude>
        <Actions>
          <SendEmail>
            <Active>TRUE</Active>
            <EmailGroup>Administration</EmailGroup>
            <MessageText />
            <SMSFormat>Default</SMSFormat>
          </SendEmail>
        </Actions>
      </Event>
      <Event>
        <Name>Event2</Name>
        <Description>Event 2</Description>
        <Latitude>0.000000</Latitude>
        <Longitude>0.000000</Longitude>
        <Actions>
          <CallPreset>
            <Active>TRUE</Active>
            <Presets>
              <Count>1</Count>
              <Preset>
                <Camera>29</Camera>
                <PresetNumber>4</PresetNumber>
              </Preset>
            </Presets>
          </CallPreset>
        </Actions>
      </Event>
    </Events>
  </Data>
</Response>
```

Example of return in JSON:

```
{
  "Response": {
    "Code": 0,
    "Message": "OK",
    "Data": {
      "Events": [
        {
          "Name": "Event1",
          "Description": "Event 1",
          "Latitude": "48.873501",
          "Longitude": "54.266785",
          "Actions": {
            "SendEmail": {
              "Active": true,
              "EmailGroup": "Administration",
              "MessageText": "",
              "SMSFormat": "Default"
            }
          }
        },
        {
          "Name": "Event2",
          "Description": "Event 2",
          "Latitude": "0.000000",
          "Longitude": "0.000000",
          "Actions": {
            "CallPreset": {
              "Active": true,
              "Presets": [
                {
                  "Camera": "29",
                  "PresetNumber": 4
                }
              ]
            }
          }
        }
      ]
    }
  }
}
```

4.4.4.2 Requesting the status of input ports

Requests the status of the alarm input ports of an I/O device.

Compatibility: Standard, Professional, Enterprise

Security level: Requires user authentication with rights to I/O device status

Method: HTTP GET

Syntax:

```
http://<server_address>/Interface/IODevices/IO/GetInputPortStatus?  
<argument=value>[&<argument=value>...][&<general argument>...]
```

Arguments:

Argument	Valid values	Description
Device*	String	Name of the I/O device

* Mandatory parameters

Example 1: Requests the status of the alarm input ports of an I/O device with response in text

```
http://192.168.0.1:8601/Interface/IODevices/IO/GetInputPortStatus?  
Device=Device1&ResponseFormat=Text
```

Example 2: Requests the status of the alarm input ports of an I/O device with response in XML

```
http://192.168.0.1:8601/Interface/IODevices/IO/GetInputPortStatus?  
Device=Device1&ResponseFormat=XML
```

Response:

A list with the status of all of the alarm input ports of the specified I/O device is returned.

HTTP Return: 200 OK

Parameters of return:**Fixed parameters:**

Parameter	Type	Description
COUNT	Integer	Total number of alarm input ports (Not included in JSON response)

Parameters of the list of ports:

Parameter	Type	Description							
STATE	String	State of the input port							
<table border="1"> <thead> <tr> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>SHORT</td> <td>Port closed</td> </tr> <tr> <td>OPEN</td> <td>Port open</td> </tr> <tr> <td>UNKNOWN</td> <td>State unknown</td> </tr> </tbody> </table>		Type	Description	SHORT	Port closed	OPEN	Port open	UNKNOWN	State unknown
Type	Description								
SHORT	Port closed								
OPEN	Port open								
UNKNOWN	State unknown								

Example of return in text:

```
RESPONSE_CODE=0  
RESPONSE_MESSAGE=OK  
COUNT=4  
PORT_1_STATE=OPEN  
PORT_2_STATE=OPEN  
PORT_3_STATE=SHORT  
PORT_4_STATE=OPEN
```

Example of return in XML:

```

<Response>
  <Code>0</Code>
  <Message>OK</Message>
  <Data>
    <Ports>
      <Count>4</Count>
      <Port>
        <State>OPEN</State>
      </Port>
      <Port>
        <State>OPEN</State>
      </Port>
      <Port>
        <State>SHORT</State>
      </Port>
      <Port>
        <State>OPEN</State>
      </Port>
    </Ports>
  </Data>
</Response>

```

Example of return in JSON:

```
{
  "Response": {
    "Code": 0,
    "Message": "OK",
    "Data": {
      "Ports": [
        {
          "State": "OPEN"
        },
        {
          "State": "OPEN"
        },
        {
          "State": "SHORT"
        },
        {
          "State": "OPEN"
        }
      ]
    }
  }
}
```

4.4.4.3 Requesting the status of input events

Requests the status of the events of alarm input of an I/O device.

Compatibility: Standard, Professional, Enterprise

Security level: Requires user authentication with rights to I/O device status

Method: HTTP GET

Syntax:

```
http://<server_address>/Interface/IODevices/IO/GetInputEventStatus?  
<argument=value>[&<argument=value>...][&<general\_argument>...]
```

Arguments:

Argument	Valid values	Description
Device*	String	Name of the I/O device

* Mandatory parameters

Example 1: Requests the status of the alarm input events of an I/O device with response in text

```
http://192.168.0.1:8601/Interface/IODevices/IO/GetInputEventStatus?  
Device=Device1&ResponseFormat=Text
```

Example 2: Requests the status of the alarm input events of an I/O device with response in XML

```
http://192.168.0.1:8601/Interface/IODevices/IO/GetInputEventStatus?  
Device=Device1&ResponseFormat=XML
```

Response:

A list of the status of all of the alarm input events of the specified I/O device is returned.

HTTP Return: 200 OK

Parameters of return:

Fixed parameters:

Parameter	Type	Description
COUNT	Integer	Total number of alarm input events (Not included in JSON response)

Parameters of the list of events:

Parameter	Type	Description						
NAME	String	Name of the event						
STATE	String	State of the input event						
		<table border="1"> <thead> <tr> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>ACTIVE</td> <td>Event active (Occurring at this moment)</td> </tr> <tr> <td>INACTIVE</td> <td>Event inactive (Not occurring)</td> </tr> </tbody> </table>	Type	Description	ACTIVE	Event active (Occurring at this moment)	INACTIVE	Event inactive (Not occurring)
Type	Description							
ACTIVE	Event active (Occurring at this moment)							
INACTIVE	Event inactive (Not occurring)							

Example of return in text:

```
RESPONSE_CODE=0  
RESPONSE_MESSAGE=OK  
COUNT=2  
EVENT_1_NAME=Event 1  
EVENT_1_STATE=ACTIVE  
EVENT_2_NAME=Event 2  
EVENT_2_STATE=INACTIVE
```

Example of return in XML:

```

<Response>
  <Code>0</Code>
  <Message>OK</Message>
  <Data>
    <Events>
      <Count>2</Count>
      <Event>
        <Name>Event 1</Name>
        <State>ACTIVE</State>
      </Event>
      <Event>
        <Name>Event 2</Name>
        <State>INACTIVE</State>
      </Event>
    </Events>
  </Data>
</Response>

```

Example of return in JSON:

```
{
  "Response": {
    "Code": 0,
    "Message": "OK",
    "Data": {
      "Events": [
        {
          "Name": "Event 1",
          "State": "ACTIVE"
        },
        {
          "Name": "Event 2",
          "State": "INACTIVE"
        }
      ]
    }
  }
}
```

4.4.4.4 Requesting the status of the output ports

Requests the status of the alarm output ports of an I/O device.

Compatibility: Standard, Professional, Enterprise

Security level: Requires user authentication with rights to I/O device status

Method: HTTP GET

Syntax:

```
http://<server_address>/Interface/IODevices/IO/GetOutputPortStatus?
<argument=value>[&<argument=value>... ] [&<general argument>... ]
```

Arguments:

Argument	Valid values	Description
Device*	String	Name of the I/O device

* Mandatory parameters

Example 1: Requests the status of the alarm output ports of an I/O device with response in text

```
http://192.168.0.1:8601/Interface/IODevices/IO/GetOutputPortStatus?
```

```
Device=Device1&ResponseFormat=Text
```

Example 2: Requests the status of the alarm output ports of an I/O device with response in XML

```
http://192.168.0.1:8601/Interface/IODevices/IO/GetOutputPortStatus?
```

```
Device=Device1&ResponseFormat=XML
```

Response:

A list with the status of all of the alarm output ports of a specified I/O device is returned.

HTTP Return: 200 OK

Parameters of return:

Fixed parameters:

Parameter	Type	Description
COUNT	Integer	Total number of alarm output ports (Not included in JSON response)

Parameters of the list of ports:

Parameter	Type	Description								
STATE	String	State of the output port <table border="1" style="margin-left: 20px;"> <thead> <tr> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>SHORT</td> <td>Port closed</td> </tr> <tr> <td>OPEN</td> <td>Port open</td> </tr> <tr> <td>UNKNOWN</td> <td>State unknown</td> </tr> </tbody> </table>	Type	Description	SHORT	Port closed	OPEN	Port open	UNKNOWN	State unknown
Type	Description									
SHORT	Port closed									
OPEN	Port open									
UNKNOWN	State unknown									

Example of return in text:

```
RESPONSE_CODE=0
RESPONSE_MESSAGE=OK
COUNT=4
PORT_1_STATE=OPEN
PORT_2_STATE=OPEN
PORT_3_STATE=SHORT
PORT_4_STATE=OPEN
```

Example of return in XML:

```

<Response>
  <Code>0</Code>
  <Message>OK</Message>
  <Data>
    <Ports>
      <Count>4</Count>
      <Port>
        <State>OPEN</State>
      </Port>
      <Port>
        <State>OPEN</State>
      </Port>
      <Port>
        <State>SHORT</State>
      </Port>
      <Port>
        <State>OPEN</State>
      </Port>
    </Ports>
  </Data>
</Response>

```

Example of return in JSON:

```
{
  "Response": {
    "Code": 0,
    "Message": "OK",
    "Data": {
      "Ports": [
        {
          "State": "OPEN"
        },
        {
          "State": "OPEN"
        },
        {
          "State": "SHORT"
        },
        {
          "State": "OPEN"
        }
      ]
    }
  }
}
```

4.4.4.5 Requesting the list of output actions

Requests the list of output actions of an I/O device.

Compatibility: Standard, Professional, Enterprise

Security level: Requires user authentication with rights to I/O device registration

Method: HTTP GET

Syntax:

```
http://<server_address>/Interface/IODevices/IO/GetOutputActions?  
<argument=value>[&<argument=value>...][&<general\_argument>...]
```

Arguments:

Argument	Valid values	Description
Device*	String	Name of the I/O device
Actions	APIMasks	Mask to filter the results. Specify which actions must be returned based on the provided masks.

* Mandatory parameters

Example 1: Request the list of output actions of an I/O device with response in text

```
http://192.168.0.1:8601/Interface/IODevices/IO/GetOutputActions?  
Device=Device1&ResponseFormat=Text
```

Example 2: Request the list of output actions of an I/O device with response in XML

```
http://192.168.0.1:8601/Interface/IODevices/IO/GetOutputActions?  
Device=Device1&ResponseFormat=XML
```

Example 3: Request the list of output actions starting with "A" of an I/O device, with response in XML

```
http://192.168.0.1:8601/Interface/IODevices/IO/GetOutputActions?  
Device=Device1&Actions=A*&ResponseFormat=XML
```

Response:

A list with all of the output actions of the specified I/O device is returned.

HTTP Return: 200 OK

Parameters of return:

Fixed parameters:

Parameter	Type	Description
COUNT	Integer	Total number of output actions (Not included in JSON response)

Parameters of the list of actions:

Parameter	Type	Description
NAME	String	Name of the action

Example of return in text:

```
RESPONSE_CODE=0  
RESPONSE_MESSAGE=OK  
COUNT=2  
ACTION_1_NAME=Action 1  
ACTION_2_NAME=Action 2
```

Example of return in XML:

```
<Response>
  <Code>0</Code>
  <Message>OK</Message>
  <Data>
    <Actions>
      <Count>2</Count>
      <Action>
        <Name>Action 1</Name>
      </Action>
      <Action>
        <Name>Action 2</Name>
      </Action>
    </Actions>
  </Data>
</Response>
```

Example of return in JSON:

```
{
  "Response": {
    "Code": 0,
    "Message": "OK",
    "Data": {
      "Actions": [
        {
          "Name": "Action 1"
        },
        {
          "Name": "Action 2"
        }
      ]
    }
  }
}
```

4.4.4.6 Triggering an output action

By way of this command you will be able to trigger a script output actions.

Compatibility: Standard, Professional, Enterprise

Security level: Requires user authentication with rights to I/O device registration

Method: HTTP GET

Syntax:

```
http://<server_address>/Interface/IODevices/IO/TriggerOutputAction?
<argument=value>[&<argument=value>...][&<general\_argument>...]
```

Arguments:

Argument	Valid values	Description
Device*	String	Name of the I/O device
Action*	String	Name of the action

* Mandatory parameters

Example 1: Activate an output action of an I/O device with response in text

```
http://192.168.0.1:8601/Interface/IODevices/IO/TriggerOutputAction?
Device=Device1&Action=Action1&ResponseFormat=Text
```

Example 2: Activate an output action of an I/O device with response in XML

```
http://192.168.0.1:8601/Interface/IODevices/IO/TriggerOutputAction?
Device=Device1&Action=Action1&ResponseFormat=XML
```

Response:

Default response of API.

HTTP Return: 200 OK

Parameters of return: [Default return of API](#)

4.4.4.7 Requesting the status of virtual ports

Requests the status of the virtual ports of an I/O device.

Compatibility: Standard, Professional, Enterprise

Security level: Requires user authentication with rights to I/O device status

Method: HTTP GET

Syntax:

```
http://<server_address>/Interface/IODevices/IO/GetVirtualPortStatus?
<argument=value>[&<argument=value>... ] [&<general argument>... ]
```

Arguments:

Argument	Valid values	Description
Device*	String	Name of the I/O device

* Mandatory parameters

Example 1: Requests the status of the virtual ports of an I/O device with response in text

```
http://192.168.0.1:8601/Interface/IODevices/IO/GetVirtualPortStatus?
Device=Device1&ResponseFormat=Text
```

Example 2: Requests the status of the virtual ports of an I/O device with response in XML

```
http://192.168.0.1:8601/Interface/IODevices/IO/GetVirtualPortStatus?
Device=Device1&ResponseFormat=XML
```

Example 2: Requests the status of the virtual ports of an I/O device with response in JSON

```
http://192.168.0.1:8601/Interface/IODevices/IO/GetVirtualPortStatus?
Device=Device1&ResponseFormat=JSON
```

Response:

A list with the status of all of the virtual ports of the specified I/O device is returned.

HTTP Return: 200 OK

Parameters of return:

Fixed parameters:

Parameter	Type	Description
COUNT	Integer	Total number of virtual ports (Not included in JSON response)

Parameters of the list of ports:

Parameter	Type	Description								
STATE	String	State of the virtual port <table border="1" style="margin-left: auto; margin-right: auto;"> <thead> <tr> <th>Type</th><th>Description</th></tr> </thead> <tbody> <tr> <td>SHORT</td><td>Port closed</td></tr> <tr> <td>OPEN</td><td>Port open</td></tr> <tr> <td>UNKNOWN</td><td>State unknown</td></tr> </tbody> </table>	Type	Description	SHORT	Port closed	OPEN	Port open	UNKNOWN	State unknown
Type	Description									
SHORT	Port closed									
OPEN	Port open									
UNKNOWN	State unknown									

Example of return in text:

```
RESPONSE_CODE=0
RESPONSE_MESSAGE=OK
COUNT=4
PORT_1_STATE=OPEN
PORT_2_STATE=OPEN
PORT_3_STATE=SHORT
PORT_4_STATE=OPEN
```

Example of return in XML:

```
<Response>
  <Code>0</Code>
  <Message>OK</Message>
  <Data>
    <Ports>
      <Count>4</Count>
      <Port>
        <State>OPEN</State>
      </Port>
      <Port>
        <State>OPEN</State>
      </Port>
      <Port>
        <State>SHORT</State>
      </Port>
      <Port>
        <State>OPEN</State>
      </Port>
    </Ports>
  </Data>
</Response>
```

Example of return in JSON:

```
{
  "Response": {
    "Code": 0,
    "Message": "OK",
    "Data": {
      "Ports": [
        {
          "State": "OPEN"
        },
        {
          "State": "OPEN"
        },
        {
          "State": "SHORT"
        },
        {
          "State": "OPEN"
        }
      ]
    }
  }
}
```

4.4.4.8 Setting the state of a virtual port

By way of this command you will be set the state an I/O Device Virtual Port

Compatibility: Standard, Professional, Enterprise

Security level: Requires user authentication with rights to I/O device registration

Method: HTTP GET

Syntax:

```
http://<server_address>/Interface/IODevices/IO/SetVirtualPortStatus?  
&argument=value [&<argument=value>... ] [&<general\_argument>... ]
```

Arguments:

Argument	Valid values	Description						
Device*	String	Name of the I/O device						
Port*	Integer	Virtual Port number						
State*	State	New state of the Virtual Port <table border="1" data-bbox="714 1600 1357 1700"> <thead> <tr> <th>State</th><th>Description</th></tr> </thead> <tbody> <tr> <td>SHORT</td><td>Port closed</td></tr> <tr> <td>OPEN</td><td>Port open</td></tr> </tbody> </table>	State	Description	SHORT	Port closed	OPEN	Port open
State	Description							
SHORT	Port closed							
OPEN	Port open							

* Mandatory parameters

Example 1: Close Virtual Port 1 of an I/O Device with response in text

```
http://192.168.0.1:8601/Interface/IODevices/IO/SetVirtualPortStatus?  
Device=Device1&Port=1&State=SHORT&ResponseFormat=Text
```

Example 2: Open Virtual Port 1 of an I/O device with response in XML

```
http://192.168.0.1:8601/Interface/IODevices/IO/SetVirtualPortStatus?
Device=Device1&Port=1&State=OPEN&ResponseFormat=Text
```

Response:

Default response of API.

HTTP Return: 200 OK

Parameters of return: [Default return of API](#)

4.5 Users

4.5.1 Requesting the list of users

Requests the list of user registered in the server.

Compatibility: All editions

Security level: Admin

Method: HTTP GET

Syntax:

```
http://<server_address>/Interface/Users/GetUsers[?<argument=value>
[&<argument=value>...]] [&<general\_argument>...]
```

Arguments:

Argument	Valid values	Description												
Users	APIMasks	Mask to filter the results. Specify which users must be returned based on the provided masks.												
Fields	Name Description Memo Groups ObjectRights	Specifies the list of desired fields. In case this parameter is omitted, all of the fields will be sent. The fields must be separated by commas <table border="1"> <thead> <tr> <th>Name</th><th>Description</th></tr> </thead> <tbody> <tr> <td>Name</td><td>Name of the user</td></tr> <tr> <td>Description</td><td>Description of the user</td></tr> <tr> <td>Memo</td><td>General-use memo of the user</td></tr> <tr> <td>Groups</td><td>Comma-separated list of groups</td></tr> <tr> <td>ObjectRights</td><td>Objects of which user has access to</td></tr> </tbody> </table>	Name	Description	Name	Name of the user	Description	Description of the user	Memo	General-use memo of the user	Groups	Comma-separated list of groups	ObjectRights	Objects of which user has access to
Name	Description													
Name	Name of the user													
Description	Description of the user													
Memo	General-use memo of the user													
Groups	Comma-separated list of groups													
ObjectRights	Objects of which user has access to													

Example 1: Request the list of users with all of the fields and response in XML

```
http://192.168.0.1:8601/Interface/Users/GetUsers?ResponseFormat=XML
```

Example 2: Request the list of users with all of the fields and response in text

```
http://192.168.0.1:8601/Interface/Users/GetUsers?ResponseFormat=Text
```

Example 3: Request the list of users with only name, response in XML and authentication of the user Admin

```
http://192.168.0.1:8601/Interface/Users/GetUsers?Fields=Name&
ResponseFormat=XML&AuthUser=admin&AuthPass=pass
```

Example 4: Request the list of users starting with "A", with only name, response in XML and authentication with Admin user

```
http://192.168.0.1:8601/Interface/Users/GetUsers?Users=A*&Fields=Name&
ResponseFormat=XML&AuthUser=admin&AuthPass=pass
```

Response:

A list with all of the users registered in the system is returned. The fields returned in the will depend on the values informed in the argument Fields

HTTP Return: 200 OK

Parameters of return:

Fixed parameters:

Parameter	Type	Description
COUNT	Integer	Total number of users (Not included in JSON response)

Parameters in the list of users:

Parameter	Type	Description
NAME	String	Name of the user
DESCRIPTION	String	Description of the user
MEMO	String	User memo in Base64 format. Base64 format is used to properly encode special characters and line breaks that might be added to the memo by the user
GROUPS	String	Comma-separated list of groups over which the user belongs to
OBJECTRIGHTS	List of Rights	List of rights to objects (Please refer below)

List of rights (Comma-delimited):

Parameter	Type	Description
MAPS	String	List with the name of maps over which the user has rights
GLOBALEVENTS	String	List with the name of global events over which the user has rights
ANALYTICSCONFIGURATIONS	String	List with the name of analytics configurations over which the user has rights
LPRCONFIGURATIONS	String	List with the name of LPR configurations over which the user has rights
LPRZONES	String	List with the name of LPR Zones over which the user has rights
LPRZONEGROUPS	String	List with the name of LPR Zone Groups over which the user has rights
CAMERASPLAYBACK	String	List with the name of cameras over which the user has rights to playback
CAMERASLIVE	String	List with the name of cameras over which the user has rights to live view
CAMERASLIVEAUDIOLISTEN	String	List with the name of cameras over which the user has rights to listen to audio
CAMERASLIVEAUDIOSPEAK	String	List with the name of cameras over which the user has rights to send audio

Parameter	Type	Description
CAMERASPTZ	String	List with the name of cameras over which the user has rights to control its PTZ
WEBPAGES	String	List with the name of web pages over which the user has rights
OPERATIONALMAPS	String	List with the name of operational maps over which the user has rights

Example of return in text:

```

RESPONSE_CODE=0
RESPONSE_MESSAGE=OK
COUNT=2
USER_1_NAME=admin
USER_1_DESCRIPTION=System administration account
USER_1_MEMO=
USER_1_GROUPS=Administrators
USER_1_OBJECTRIGHTS_MAPS=01,02,03
USER_1_OBJECTRIGHTS_GLOBALEVENTS=Event1,Event2
USER_1_OBJECTRIGHTS_ANALYTICSCONFIGURATIONS=01,02,03
USER_1_OBJECTRIGHTS_LPRCONFIGURATIONS=01,02,03
USER_1_OBJECTRIGHTS_LPRZONES=01,02,03
USER_1_OBJECTRIGHTS_LPRZONEGROUPS=01,02,03
USER_1_OBJECTRIGHTS_CAMERASPLAYBACK=01,02,03
USER_1_OBJECTRIGHTS_CAMERASLIVE=01,02,03
USER_1_OBJECTRIGHTS_CAMERASLIVEAUDIOLISTEN=01,02,03
USER_1_OBJECTRIGHTS_CAMERASLISTAUDIOSSPEAK=01,02,03
USER_1_OBJECTRIGHTS_CAMERASPTZ=01,02,03
USER_1_OBJECTRIGHTS_WEBPAGES=01,02,03
USER_1_OBJECTRIGHTS_OPERATIONALMAPS=01,02,03
USER_2_NAME=Guest
USER_2_DESCRIPTION=Guest user
USER_2_MEMO=VXNlciB3aXR0IHZJlc3RyawN0ZWQgYWNjZXNz
USER_2_GROUPS=Restricted, Guests
USER_2_OBJECTRIGHTS_MAPS=01,02
USER_2_OBJECTRIGHTS_GLOBALEVENTS=Event1
USER_2_OBJECTRIGHTS_ANALYTICSCONFIGURATIONS=01,02
USER_2_OBJECTRIGHTS_LPRCONFIGURATIONS=01,02
USER_2_OBJECTRIGHTS_LPRZONES=01,02,03
USER_2_OBJECTRIGHTS_LPRZONEGROUPS=01,02,03
USER_2_OBJECTRIGHTS_CAMERASPLAYBACK=01,02
USER_2_OBJECTRIGHTS_CAMERASLIVE=01,02
USER_2_OBJECTRIGHTS_CAMERASLIVEAUDIOLISTEN=01,02
USER_2_OBJECTRIGHTS_CAMERASLISTAUDIOSSPEAK=01,02
USER_2_OBJECTRIGHTS_CAMERASPTZ=01,02
USER_2_OBJECTRIGHTS_WEBPAGES=01,02,03
USER_2_OBJECTRIGHTS_OPERATIONALMAPS=01,02,03

```

Example of return in XML:

```

<Response>
  <Code>0</Code>
  <Message>OK</Message>
  <Data>
    <Users>
      <User>
        <Name>admin</Name>
        <Description>System administration account</Description>
        <Memo/>
        <Groups>Administrators</Groups>
        <ObjectRights>
          <Maps>01,02,03</Maps>
          <GlobalEvents>Event1,Event2</GlobalEvents>
          <AnalyticsConfigurations>01,02,03</AnalyticsConfigurations>
          <LPRConfigurations>01,02,03</LPRConfigurations>
          <LPRZones>01,02,03</LPRZones>
          <LPRZoneGroups>01,02,03</LPRZoneGroups>
          <CamerasPlayback>01,02,03</CamerasPlayback>
          <CamerasLive>01,02,03</CamerasLive>
          <CamerasLiveAudioListen>01,02,03</CamerasLiveAudioListen>
          <CamerasLiveAudioSpeak>01,02,03</CamerasLiveAudioSpeak>
          <CamerasPTZ>01,02,03</CamerasPTZ>
          <WebPages>01,02,03</WebPages>
          <OperationalMaps>01,02,03</OperationalMaps>
        </ObjectRights>
      </User>
      <User>
        <Name>Guest</Name>
        <Description>Guest user</Description>
        <Memo>VXNlcib3aXRoIHZJlc3RyaWN0ZWQgYWNjZXNz</Memo>
        <Groups>Restricted, Guests</Groups>
        <ObjectRights>
          <Maps>01,02</Maps>
          <GlobalEvents>Event1</GlobalEvents>
          <AnalyticsConfigurations>01,02</AnalyticsConfigurations>
          <LPRConfigurations>01,02</LPRConfigurations>
          <LPRZones>01,02,03</LPRZones>
          <LPRZoneGroups>01,02,03</LPRZoneGroups>
          <CamerasPlayback>01,02</CamerasPlayback>
          <CamerasLive>01,02</CamerasLive>
          <CamerasLiveAudioListen>01,02</CamerasLiveAudioListen>
          <CamerasLiveAudioSpeak>01,02</CamerasLiveAudioSpeak>
          <CamerasPTZ>01,02</CamerasPTZ>
          <WebPages>01,02,03</WebPages>
          <OperationalMaps>01,02,03</OperationalMaps>
        </ObjectRights>
      </User>
    </Users>
  </Data>
</Response>

```

Example of return in JSON:

```
{  
    "Response": {  
        "Code": 0,  
        "Message": "OK",  
        "Data": {  
            "Users": [  
                {  
                    "Name": "admin",  
                    "Description": "System administration account",  
                    "Memo": "",  
                    "Groups": "Administrators",  
                    "ObjectRights": {  
                        "Maps": "01,02,03",  
                        "GlobalEvents": "Event1,Event2",  
                        "AnalyticsConfigurations": "01,02,03",  
                        "LPRConfigurations": "01,02,03",  
                        "LPRZones": "01,02,03",  
                        "LPRZoneGroups": "01,02,03",  
                        "CamerasPlayback": "01,02,03",  
                        "CamerasLive": "01,02,03",  
                        "CamerasLiveAudioListen": "01,02,03",  
                        "CamerasListAudioSpeak": "01,02,03",  
                        "CamerasPTZ": "01,02,03",  
                        "WebPages": "01,02,03",  
                        "OperationalMaps": "01,02,03"  
                    }  
                },  
                {  
                    "Name": "Guest",  
                    "Description": "Guest user",  
                    "Memo": "VXNlciB3aXR0IHJlc3RyaWN0ZWQgYWNjZXNz==",  
                    "Groups": "Restricted,Guests",  
                    "ObjectRights": {  
                        "Maps": "01,02",  
                        "GlobalEvents": "Event1",  
                        "AnalyticsConfigurations": "01,02",  
                        "LPRConfigurations": "01,02",  
                        "LPRZones": "01,02,03",  
                        "LPRZoneGroups": "01,02,03",  
                        "CamerasPlayback": "01,02",  
                        "CamerasLive": "01,02",  
                        "CamerasLiveAudioListen": "01,02",  
                        "CamerasListAudioSpeak": "01,02",  
                        "CamerasPTZ": "01,02",  
                        "WebPages": "01,02,03",  
                        "OperationalMaps": "01,02,03"  
                    }  
                }  
            ]  
        }  
    }  
}
```

4.5.2 Requesting the list of groups

Requests the list of user groups registered in the server.

Compatibility: All editions

Security level: Admin

Method: HTTP GET

Syntax:

```
http://<server_address>/Interface/Users/GetGroups [?<argument=value>
[&<argument=value>...]] [&<general argument>...]
```

Arguments:

Argument	Valid values	Description										
Groups	APIMasks	Mask to filter the results. Specify which groups must be returned based on the provided masks.										
Fields	Name Description Users ObjectRights	Specifies the list of desired fields. In case this parameter is omitted, all of the fields will be sent. The fields must be separated by commas <table border="1"> <thead> <tr> <th>Name</th><th>Description</th></tr> </thead> <tbody> <tr> <td>Name</td><td>Name of the group</td></tr> <tr> <td>Description</td><td>Description of the group</td></tr> <tr> <td>Users</td><td>Comma-separated list of users</td></tr> <tr> <td>ObjectRights</td><td>Objects of which group has access to</td></tr> </tbody> </table>	Name	Description	Name	Name of the group	Description	Description of the group	Users	Comma-separated list of users	ObjectRights	Objects of which group has access to
Name	Description											
Name	Name of the group											
Description	Description of the group											
Users	Comma-separated list of users											
ObjectRights	Objects of which group has access to											

Example 1: Request the list of user groups with all of the fields and response in XML

```
http://192.168.0.1:8601/Interface/Users/GetGroups?ResponseFormat=XML
```

Example 2: Request the list of user groups with all of the fields and response in text

```
http://192.168.0.1:8601/Interface/Users/GetGroups?ResponseFormat=Text
```

Example 3: Request the list of user groups with only name, response in XML and authentication of the user Admin

```
http://192.168.0.1:8601/Interface/Users/GetGroups?Fields=Name&
ResponseFormat=XML&AuthUser=admin&AuthPass=pass
```

Example 4: Request the list of user groups starting with "A", with only name, response in XML and authentication with Admin user

```
http://192.168.0.1:8601/Interface/Users/GetGroups?Users=A*&Fields=Name&
ResponseFormat=XML&AuthUser=admin&AuthPass=pass
```

Response:

A list with all of the user groups registered in the system is returned. The fields returned in the will depend on the values informed in the argument Fields

HTTP Return: 200 OK

Parameters of return:**Fixed parameters:**

Parameter	Type	Description
COUNT	Integer	Total number of groups (Not included in JSON response)

Parameters in the list of groups:

Parameter	Type	Description
NAME	String	Name of the group
DESCRIPTION	String	Description of the group
USERS	String	Comma-separated list of users from the group
OBJECTRIGHTS	List of Rights	List of rights to objects (Please refer below)

List of rights (Comma-delimited):

Parameter	Type	Description
MAPS	String	List with the name of maps of which the group has rights to
GLOBALEVENTS	String	List with the name of global events of which the group has rights to
ANALYTICSCONFIGURATIONS	String	List with the name of analytics configurations of which the group has rights to
LPRCONFIGURATIONS	String	List with the name of LPR configurations of which the group has rights to
LPRZONES	String	List with the name of LPR Zones of which the group has rights to
LPRZONEGROUPS	String	List with the name of LPR Zone Groups of which the group has rights to
CAMERASPLAYBACK	String	List with the name of cameras of which the group has rights to playback
CAMERASLIVE	String	List with the name of cameras of which the group has rights to live view
CAMERASLIVEAUDIOLISTEN	String	List with the name of cameras of which the group has rights to listen to audio
CAMERASLIVEAUDIOSPEAK	String	List with the name of cameras of which the group has rights to send audio
CAMERASPTZ	String	List with the name of cameras of which the group has rights to control its PTZ
WEBPAGES	String	List with the name of web pages of which the group has rights to view
OPERATIONALMAPS	String	List with the name of operational maps of which the group has rights to view

Example of return in text:

```
RESPONSE_CODE=0
RESPONSE_MESSAGE=OK
COUNT=2
GROUP_1_NAME=Administrators
GROUP_1_DESCRIPTION=Group for system administrators
GROUP_1_USERS=admin,user1,user2
GROUP_1_OBJECTRIGHTS_MAPS=01,02,03
GROUP_1_OBJECTRIGHTS_GLOBALEVENTS=Event1,Event2
GROUP_1_OBJECTRIGHTS_ANALYTICSCONFIGURATIONS=01,02,03
GROUP_1_OBJECTRIGHTS_LPRCONFIGURATIONS=01,02,03
GROUP_1_OBJECTRIGHTS_LPRZONES=01,02,03
GROUP_1_OBJECTRIGHTS_LPRZONEGROUPS=01,02,03
GROUP_1_OBJECTRIGHTS_CAMERASPLAYBACK=01,02,03
GROUP_1_OBJECTRIGHTS_CAMERASLIVE=01,02,03
GROUP_1_OBJECTRIGHTS_CAMERASLIVEAUDIOLISTEN=01,02,03
GROUP_1_OBJECTRIGHTS_CAMERASLISTAUDIOSPEAK=01,02,03
GROUP_1_OBJECTRIGHTS_CAMERASPTZ=01,02,03
GROUP_1_OBJECTRIGHTS_WEBPAGES=01,02,03
GROUP_1_OBJECTRIGHTS_OPERATIONALMAPS=01,02,03
GROUP_2_NAME=Operators
GROUP_2_DESCRIPTION=Operational users
GROUP_2_USERS=operator1,operator2
GROUP_2_OBJECTRIGHTS_MAPS=01,02
GROUP_2_OBJECTRIGHTS_GLOBALEVENTS=Event1
GROUP_2_OBJECTRIGHTS_ANALYTICSCONFIGURATIONS=01,02
GROUP_2_OBJECTRIGHTS_LPRCONFIGURATIONS=01,02
GROUP_2_OBJECTRIGHTS_LPRZONES=01,02,03
GROUP_2_OBJECTRIGHTS_LPRZONEGROUPS=01,02,03
GROUP_2_OBJECTRIGHTS_CAMERASPLAYBACK=01,02
GROUP_2_OBJECTRIGHTS_CAMERASLIVE=01,02
GROUP_2_OBJECTRIGHTS_CAMERASLIVEAUDIOLISTEN=01,02
GROUP_2_OBJECTRIGHTS_CAMERASLISTAUDIOSPEAK=01,02
GROUP_2_OBJECTRIGHTS_CAMERASPTZ=01,02
GROUP_2_OBJECTRIGHTS_WEBPAGES=01,02,03
GROUP_2_OBJECTRIGHTS_OPERATIONALMAPS=01,02,03
```

Example of return in XML:

```
<Response>
  <Code>0</Code>
  <Message>OK</Message>
  <Data>
    <Groups>
      <Count>2</Count>
      <Group>
        <Name>Administrators</Name>
        <Description>Group for system administrators</Description>
        <Users>admin,user1,user2</Users>
        <ObjectRights>
          <Maps>01,02,03</Maps>
          <GlobalEvents>Event1,Event2</GlobalEvents>
          <AnalyticsConfigurations>01,02,03</AnalyticsConfigurations>
          <LPRConfigurations>01,02,03</LPRConfigurations>
          <LPRZones>01,02,03</LPRZones>
          <LPRZoneGroups>01,02,03</LPRZoneGroups>
          <CamerasPlayback>01,02,03</CamerasPlayback>
          <CamerasLive>01,02,03</CamerasLive>
          <CamerasLiveAudioListen>01,02,03</CamerasLiveAudioListen>
          <CamerasLiveAudioSpeak>01,02,03</CamerasLiveAudioSpeak>
          <CamerasPTZ>01,02,03</CamerasPTZ>
          <WebPages>01,02,03</WebPages>
          <OperationalMaps>01,02,03</OperationalMaps>
        </ObjectRights>
      </Group>
      <Group>
        <Name>Operators</Name>
        <Description>Operational users</Description>
        <Users>operator1,operator2</Users>
        <ObjectRights>
          <Maps>01,02</Maps>
          <GlobalEvents>Event1</GlobalEvents>
          <AnalyticsConfigurations>01,02</AnalyticsConfigurations>
          <LPRConfigurations>01,02</LPRConfigurations>
          <LPRZones>01,02,03</LPRZones>
          <LPRZoneGroups>01,02,03</LPRZoneGroups>
          <CamerasPlayback>01,02</CamerasPlayback>
          <CamerasLive>01,02</CamerasLive>
          <CamerasLiveAudioListen>01,02</CamerasLiveAudioListen>
          <CamerasLiveAudioSpeak>01,02</CamerasLiveAudioSpeak>
          <CamerasPTZ>01,02</CamerasPTZ>
          <WebPages>01,02,03</WebPages>
          <OperationalMaps>01,02,03</OperationalMaps>
        </ObjectRights>
      </Group>
    </Groups>
  </Data>
</Response>
```

Example of return in JSON:

```
{  
    "Response": {  
        "Code": 0,  
        "Message": "OK",  
        "Data": {  
            "Groups": [  
                {  
                    "Name": "Administrators",  
                    "Description": "Group for system administrators",  
                    "Users": "admin,user1,user2",  
                    "ObjectRights": {  
                        "Maps": "01,02,03",  
                        "GlobalEvents": "Event1,Event2",  
                        "AnalyticsConfigurations": "01,02,03",  
                        "LPRConfigurations": "01,02,03",  
                        "LPRZones": "01,02,03",  
                        "LPRZoneGroups": "01,02,03",  
                        "CamerasPlayback": "01,02,03",  
                        "CamerasLive": "01,02,03",  
                        "CamerasLiveAudioListen": "01,02,03",  
                        "CamerasListAudioSpeak": "01,02,03",  
                        "CamerasPTZ": "01,02,03",  
                        "WebPages": "01,02,03",  
                        "OperationalMaps": "01,02,03"  
                    }  
                },  
                {  
                    "Name": "Operators",  
                    "Description": "Operational users",  
                    "Users": "operator1,operator2",  
                    "ObjectRights": {  
                        "Maps": "01,02",  
                        "GlobalEvents": "Event1",  
                        "AnalyticsConfigurations": "01,02",  
                        "LPRConfigurations": "01,02",  
                        "LPRZones": "01,02,03",  
                        "LPRZoneGroups": "01,02,03",  
                        "CamerasPlayback": "01,02",  
                        "CamerasLive": "01,02",  
                        "CamerasLiveAudioListen": "01,02",  
                        "CamerasListAudioSpeak": "01,02",  
                        "CamerasPTZ": "01,02",  
                        "WebPages": "01,02,03",  
                        "OperationalMaps": "01,02,03"  
                    }  
                }  
            ]  
        }  
    }  
}
```

4.5.3 Requesting the list of current connected users

Request the list of connected users.

Compatibility: All editions

Security level: Requires user authentication with rights to view the status of connected users

Method: HTTP GET

Syntax:

```
http://<server_address>/Interface/Users/GetConnections [?<argument=value>
[&<argument=value>...]] [&<general\_argument>...]]
```

Arguments:

Argument	Valid values	Description
Usernames	APIMasks	Mask to filter the results. Filter the results to include only the connections from the specified usernames
IPs	APIMasks	Mask to filter the results. Filter the results to include only the connections from the specified IPs
ConnectionTypes	CONNECTION ADMINISTRATION_CLIENT SURVEILLANCE_CLIENT CAMERA_LIVE WEB_PLAYBACK_PLUGIN CAMERA_PLAYBACK MAP ANALYTICS_LIVE_VIEW LPR_LIVE_VIEW LPR_ZONE_LIVE_VIEW OPERATIONAL_MAP_LIVE_VIEW DIGIFORT_SERVER WEB_SERVER RTSP_INTERFACE WEB_API_SERVER	Specifies the type of connections to filter.
Details	APIMasks	Mask to filter the results. Filter the results to include only the connections with the specified details

Example 1: Request the list of all connected users with response in XML

```
http://192.168.0.1:8601/Interface/Users/GetConnections?ResponseFormat=XML
```

Example 2: Request the status of all connected users with response in text

```
http://192.168.0.1:8601/Interface/Users/GetConnections?ResponseFormat=Text
```

Example 3: Request the list of all connections for user Operator1 with response in text

```
http://192.168.0.1:8601/Interface/Users/GetConnections?Usernames=Operator1&
ResponseFormat=Text
```

Example 4: Request the list of all connections from live and playback cameras with response in XML

```
http://192.168.0.1:8601/Interface/Users/GetConnections?
ConnectionTypes=CAMERA_LIVE,CAMERA_PLAYBACK&ResponseFormat=XML
```

Response:

A list with the status of all of the connected users.

HTTP Return: 200 OK

Parameters of return:**Fixed Parameter:**

Parameter	Type	Description
COUNT	Integer	Total number of connections (Not included in JSON response)

Parameters in the list of status of cameras:

Parameter	Type	Description																																		
USERNAME	String	Name of the user																																		
IP	String	IP address of the connection																																		
CONNECTIONTIME	Integer	Amount of seconds since the connection was open																																		
CONNECTIONTYPE	String	<p>Type of connection</p> <table border="1"> <thead> <tr> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>CONNECTION</td> <td>Unauthenticated TCP connection</td> </tr> <tr> <td>ADMINISTRATION_CLIENT</td> <td>Connection from Administration Client</td> </tr> <tr> <td>SURVEILLANCE_CLIENT</td> <td>Connection from Surveillance Client</td> </tr> <tr> <td>CAMERA_LIVE</td> <td>Connection from a live camera</td> </tr> <tr> <td>WEB_LIVE_PLUGIN</td> <td>Connection from ActiveX monitoring plugin</td> </tr> <tr> <td>WEB_PLAYBACK_PLUGIN</td> <td>Connection from ActiveX playback plugin</td> </tr> <tr> <td>CAMERA_PLAYBACK</td> <td>Connection for camera playback</td> </tr> <tr> <td>MAP</td> <td>Connection from a map object</td> </tr> <tr> <td>ANALYTICS_LIVE_VIEW</td> <td>Connection from a live analytics object</td> </tr> <tr> <td>LPR_LIVE_VIEW</td> <td>Connection from a live LPR object</td> </tr> <tr> <td>LPR_ZONE_LIVE_VIEW</td> <td>Connection from a live LPR Zone object</td> </tr> <tr> <td>OPERATIONAL_MAP_LIVE_VIEW</td> <td>Connection from an Operational Map object</td> </tr> <tr> <td>DIGIFORT_SERVER</td> <td>Connection from another Digifort server</td> </tr> <tr> <td>WEB_SERVER</td> <td>Connection from web server</td> </tr> <tr> <td>RTSP_INTERFACE</td> <td>Connection from RTSP server</td> </tr> <tr> <td>WEB_API_SERVER</td> <td>Connection from API server</td> </tr> </tbody> </table>	Type	Description	CONNECTION	Unauthenticated TCP connection	ADMINISTRATION_CLIENT	Connection from Administration Client	SURVEILLANCE_CLIENT	Connection from Surveillance Client	CAMERA_LIVE	Connection from a live camera	WEB_LIVE_PLUGIN	Connection from ActiveX monitoring plugin	WEB_PLAYBACK_PLUGIN	Connection from ActiveX playback plugin	CAMERA_PLAYBACK	Connection for camera playback	MAP	Connection from a map object	ANALYTICS_LIVE_VIEW	Connection from a live analytics object	LPR_LIVE_VIEW	Connection from a live LPR object	LPR_ZONE_LIVE_VIEW	Connection from a live LPR Zone object	OPERATIONAL_MAP_LIVE_VIEW	Connection from an Operational Map object	DIGIFORT_SERVER	Connection from another Digifort server	WEB_SERVER	Connection from web server	RTSP_INTERFACE	Connection from RTSP server	WEB_API_SERVER	Connection from API server
Type	Description																																			
CONNECTION	Unauthenticated TCP connection																																			
ADMINISTRATION_CLIENT	Connection from Administration Client																																			
SURVEILLANCE_CLIENT	Connection from Surveillance Client																																			
CAMERA_LIVE	Connection from a live camera																																			
WEB_LIVE_PLUGIN	Connection from ActiveX monitoring plugin																																			
WEB_PLAYBACK_PLUGIN	Connection from ActiveX playback plugin																																			
CAMERA_PLAYBACK	Connection for camera playback																																			
MAP	Connection from a map object																																			
ANALYTICS_LIVE_VIEW	Connection from a live analytics object																																			
LPR_LIVE_VIEW	Connection from a live LPR object																																			
LPR_ZONE_LIVE_VIEW	Connection from a live LPR Zone object																																			
OPERATIONAL_MAP_LIVE_VIEW	Connection from an Operational Map object																																			
DIGIFORT_SERVER	Connection from another Digifort server																																			
WEB_SERVER	Connection from web server																																			
RTSP_INTERFACE	Connection from RTSP server																																			
WEB_API_SERVER	Connection from API server																																			
DETAILS	String	Details of the connection. Some connection types such as CAMERA_LIVE will return the name of the object being visualized.																																		

Example of return in text:

```

RESPONSE_CODE=0
RESPONSE_MESSAGE=OK
COUNT=2
CONNECTION_1_USERNAME=videowall
CONNECTION_1_IP=192.168.0.10
CONNECTION_1_CONNECTIONTIME=1683046
CONNECTION_1_CONNECTIONTYPE=SURVEILLANCE_CLIENT
CONNECTION_1_DETAILS=
CONNECTION_2_USERNAME=videowall
CONNECTION_2_IP=192.168.0.10
CONNECTION_2_CONNECTIONTIME=1682987
CONNECTION_2_CONNECTIONTYPE=CAMERA_LIVE
CONNECTION_2_DETAILS=Cameral

```

Example of return in XML:

```

<Response>
  <Code>0</Code>
  <Message>OK</Message>
  <Data>
    <Connections>
      <Count>2</Count>
      <Connection>
        <Username>videowall</Username>
        <IP>192.168.0.10</IP>
        <ConnectionTime>1683046</ConnectionTime>
        <ConnectionType>SURVEILLANCE_CLIENT</ConnectionType>
        <Details/>
      </Connection>
      <Connection>
        <Username>videowall</Username>
        <IP>192.168.0.10</IP>
        <ConnectionTime>1682987</ConnectionTime>
        <ConnectionType>CAMERA_LIVE</ConnectionType>
        <Details>Cameral</Details>
      </Connection>
    </Connections>
  </Data>
</Response>

```

Example of return in JSON:

```
{
  "Response": {
    "Code": 0,
    "Message": "OK",
    "Data": {
      "Connections": [
        {
          "Username": "videowall",
          "IP": "192.168.0.10",
          "ConnectionTime": 1683046,
          "ConnectionType": "SURVEILLANCE_CLIENT",
          "Details": ""
        },
        {
          "Username": "videowall",
          "IP": "192.168.0.10",
          "ConnectionTime": 1682987,
          "ConnectionType": "CAMERA_LIVE",
          "Details": "Cameral"
        }
      ]
    }
  }
}
```

4.5.4 Blocking / Unblocking users

Allows the blocking or unblocking of multiple users simultaneously

Compatibility: All editions

Security level: Requires user authentication with rights to configure users

Method: HTTP GET

Syntax:

```
http://<server_address>/Interface/Users/Block?<argument=value>
[&<argument=value>...] [&<general_argument>...]
```

Arguments:

Argument	Valid values	Description								
Users*	String	<p>List of users to block or unblock.</p> <p>The user names must be separated by commas.</p> <p>This command is only mandatory when Action=Block or Action=Unblock</p>								
Action*	Block Unblock BlockAll UnblockAll	<p>Action to be executed</p> <table border="1"> <thead> <tr> <th>Operation</th><th>Description</th></tr> </thead> <tbody> <tr> <td>Block</td><td>Block the users on Users argument</td></tr> <tr> <td>Unblock</td><td>Unblock the users on Users argument</td></tr> <tr> <td>BlockAll</td><td>Block all users from the server</td></tr> </tbody> </table>	Operation	Description	Block	Block the users on Users argument	Unblock	Unblock the users on Users argument	BlockAll	Block all users from the server
Operation	Description									
Block	Block the users on Users argument									
Unblock	Unblock the users on Users argument									
BlockAll	Block all users from the server									

		UnblockAll	Unblock all users from the server
--	--	-------------------	-----------------------------------

* Mandatory parameters

Note: The Admin user can never be blocked. This command will not block the Admin user.

Example 1: Block all users from the server

```
http://192.168.0.1:8601/Interface/Users/Block?Action=BlockAll
```

Example 2: Block users User1 and User2

```
http://192.168.0.1:8601/Interface/Users/Block?Action=Block&Users=User1,User2
```

Example 3: Unblock all users from the server

```
http://192.168.0.1:8601/Interface/Users/Block?Action=UnblockAll
```

Response:

Default response of API.

HTTP Return HTTP: 200 OK

Parameters of return: [Default return of API](#)

4.6 Screenstyles

4.6.1 Requesting the list of screenstyles

Requests the list of screenstyles registered in the server.

Compatibility: All editions

Security level: Requires user authentication

Method: HTTP GET

Syntax:

```
http://<server_address>/Interface/ScreenStyles/GetScreenStyles  
[?<general_argument>[&<general_argument>...]]
```

Example 1: Requests the list of screenstyles with response in XML

```
http://192.168.0.1:8601/Interface/ScreenStyles/GetScreenStyles?  
ResponseFormat=XML
```

Example 2: Requests the list of screenstyles with response in text

```
http://192.168.0.1:8601/Interface/ScreenStyles/GetScreenStyles?  
ResponseFormat=Text
```

Example 3: Requests the list of screenstyles with response in JSON

```
http://192.168.0.1:8601/Interface/ScreenStyles/GetScreenStyles?  
ResponseFormat=JSON
```

Response:

A list with all of the screenstyles registered in the system is returned.

HTTP Return: 200 OK

Parameters of return:

Fixed parameters:

Parameter	Type	Description
COUNT	Integer	Total number of screenstyles (Not included in JSON response)

Parameters of the list of screenstyles:

Parameter	Type	Description
ID	Integer	Identification of the screenstyle
Data	String	Internal structure data (For specific integration, ask our development team on how to use this data structure)

Example of return in text:

```
RESPONSE_CODE=0
RESPONSE_MESSAGE=OK
COUNT=5
SCREENSTYLE_1_ID=1399
SCREENSTYLE_1_DATA=1,1
SCREENSTYLE_2_ID=6278
SCREENSTYLE_2_DATA=2,1,2,3,4
SCREENSTYLE_3_ID=7186
SCREENSTYLE_3_DATA=4,1,1,2,2,1,1,2,2,3,3,2,2,3,3,2,2,
SCREENSTYLE_4_ID=8320
SCREENSTYLE_4_DATA=4,1,1,1,1,1,1,1,1,1,1,1,2,3,4,5,
SCREENSTYLE_5_ID=9983
SCREENSTYLE_5_DATA=3,1,1,2,1,1,3,4,5,6
```

Example of return in XML:

```
<Response>
  <Code>0</Code>
  <Message>OK</Message>
  <Data>
    <ScreenStyles>
      <Count>5</Count>
      <ScreenStyle>
        <ID>1399</ID>
        <Data>1,1</Data>
      </ScreenStyle>
      <ScreenStyle>
        <ID>6278</ID>
        <Data>2,1,2,3,4</Data>
      </ScreenStyle>
      <ScreenStyle>
        <ID>7186</ID>
        <Data>4,1,1,2,2,1,1,2,2,3,3,2,2,3,3,2,2,</Data>
      </ScreenStyle>
      <ScreenStyle>
        <ID>8320</ID>
        <Data>4,1,1,1,1,1,1,1,1,1,1,2,3,4,5,</Data>
      </ScreenStyle>
      <ScreenStyle>
        <ID>9983</ID>
        <Data>3,1,1,2,1,1,3,4,5,6</Data>
      </ScreenStyle>
    </ScreenStyles>
  </Data>
</Response>
```

Example of return in JSON:

```
{
  "Response": {
    "Code": 0,
    "Message": "OK",
    "Data": {
      "ScreenStyles": [
        {
          "ID": 1399,
          "Data": "1,1"
        },
        {
          "ID": 6278,
          "Data": "2,1,2,3,4"
        },
        {
          "ID": 7186,
          "Data": "4,1,1,2,2,1,1,2,2,3,3,2,2,3,3,2,2,"
        },
        {
          "ID": 8320,
          "Data": "4,1,1,1,1,1,1,1,1,1,1,1,2,3,4,5,"
        },
        {
          "ID": 9983,
          "Data": "3,1,1,2,1,1,3,4,5,6"
        }
      ]
    }
  }
}
```

4.6.2 Requesting the image of a screenstyle

Requesting the illustrative image of a screenstyle in JPEG.

Compatibility: All editions

Security level: Requires user authentication

Method: HTTP GET

Syntax:

```
http://<server_address>/Interface/ScreenStyles/GetScreenStyleImage?
<argument=value>[&<argument=value>...] [&<general argument>...]
```

Arguments:

Argument	Valid values	Description
ID*	Integer	ID of the screenstyle Special styles: 0 - AUTO 1 - TIMER
Selected	0, 1	Indicates whether the image will be displayed with a normal border (in

	<p>black) or a selected border (in red). if this parameter is omitted, the default value 0 will be used.</p> <table border="1"> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>0</td><td>Normal border (black)</td></tr> <tr> <td>1</td><td>Selected border (red)</td></tr> </tbody> </table>	Value	Description	0	Normal border (black)	1	Selected border (red)
Value	Description						
0	Normal border (black)						
1	Selected border (red)						

* Mandatory parameters

Example 1: Requests the image of screenstyle 1399 with error response in XML

```
http://192.168.0.1:8601/Interface/ScreenStyles/GetScreenStyleImage?
ID=1399&ResponseFormat=XML
```

Example 2: Requests the image of screenstyle 6278 with error response in text

```
http://192.168.0.1:8601/Interface/ScreenStyles/GetScreenStyleImage?
ID=6278&ResponseFormat=Text
```

Example 3: Requests the image of screenstyle 6278 with a selected border and error response in text

```
http://192.168.0.1:8601/Interface/ScreenStyles/GetScreenStyleImage?
ID=6278&Selected=1&ResponseFormat=Text
```

Example 3: Requests the image of AUTO screenstyle with a selected border and error response in text

```
http://192.168.0.1:8601/Interface/ScreenStyles/GetScreenStyleImage?
ID=0&Selected=1&ResponseFormat=Text
```

Response:

In case of success, a JPEG image will be returned. In case of error, a code and an error message will be returned. An error can be returned if the screenstyle is not found.

HTTP Return: 200 OK

Parameters of return: [Default return of API](#)

4.7 Screen views

4.7.1 Requesting the list of screen views of the user

Request the list of screen views of the user used to request the command

Compatibility: All editions

Security level: Requires user authentication

Method: HTTP GET

Syntax:

```
http://<server_address>/Interface/ScreenViews/ GetUserScreenViews
[?<argument=value> [&<argument=value>...]] [&<general\_argument>...]
```

Arguments:

Argument	Valid values	Description
ScreenStyleID	Integer	Filters the results to display only the screen views of the specified

		screenstyle. If this parameter is omitted, all of the screen views of the user will be listed.
ScreenViews	APIMasks	Mask to filter the results. Specify which screen views must be returned based on the provided masks.

Example 1: Request the list of screen views of the user with response in XML

```
http://192.168.0.1:8601/Interface/ScreenViews/GetUserScreenViews?
ResponseFormat=XML
```

Example 2: Request the list of screen view of the user guest1 for the screenstyle 9983 and response in text

```
http://192.168.0.1:8601/Interface/ScreenViews/GetUserScreenViews?
ScreenStyleID=9983&AuthUser=guest1&AuthPass=guestpass&ResponseFormat=Text
```

Example 3: Request the list of screen views staring with "A", from user "guest1" for screen style 9983 with response in text

```
http://192.168.0.1:8601/Interface/ScreenViews/GetUserScreenViews?
ScreenStyleID=9983&ScreenViews=A*&AuthUser=guest1&AuthPass=guestpass&
ResponseFormat=Text
```

Response:

A list with all of the screen views of the user is returned

HTTP Return: 200 OK

Parameters of return:**Fixed parameters:**

Parameter	Type	Description
COUNT	Integer	Total number of screen views (Not included in JSON response)

Parameters of the list of screen views:

Parameter	Type	Description
NAME	String	Name of the screen view
SCREENSTYLEID	Integer	ID of the screenstyle of the screen view

Example of return in text:

```
RESPONSE_CODE=0
RESPONSE_MESSAGE=OK
COUNT=3
SCREENVIEW_1_NAME=View 1
SCREENVIEW_1_SCREENSTYLEID=6278
SCREENVIEW_2_NAME=View 2
SCREENVIEW_2_SCREENSTYLEID=9983
SCREENVIEW_3_NAME=View 3
SCREENVIEW_3_SCREENSTYLEID=9983
```

Example of return in XML:

```
<Response>
  <Code>0</Code>
  <Message>OK</Message>
  <Data>
    <ScreenViews>
      <Count>3</Count>
      <ScreenView>
        <Name>View 1</Name>
        <ScreenStyleID>6278</ScreenStyleID>
      </ScreenView>
      <ScreenView>
        <Name>View 2</Name>
        <ScreenStyleID>9983</ScreenStyleID>
      </ScreenView>
      <ScreenView>
        <Name>View 3</Name>
        <ScreenStyleID>9983</ScreenStyleID>
      </ScreenView>
    </ScreenViews>
  </Data>
</Response>
```

Example of return in JSON:

```
{
  "Response": {
    "Code": 0,
    "Message": "OK",
    "Data": {
      "ScreenViews": [
        {
          "Name": "View 1",
          "ScreenStyleID": 6278
        },
        {
          "Name": "View 2",
          "ScreenStyleID": 9983
        },
        {
          "Name": "View 3",
          "ScreenStyleID": 9983
        }
      ]
    }
  }
}
```

4.7.2 Requesting the list of public screen views

Requests the list of public screen views of the system.

Compatibility: All editions

Security level: Requires user authentication

Method: HTTP GET**Syntax:**

```
http://<server_address>/Interface/ScreenViews/GetPublicScreenViews
[?<argument=value> [&<argument=value>...]] [&<general\_argument>...]
```

Arguments:

Argument	Valid values	Description
ScreenStyleID	Integer	Filters the results to display only the screen views of the specified screenstyle. If this parameter is omitted, all of the public screen views will be listed.
ScreenViews	APIMasks	Mask to filter the results. Specify which screen views must be returned based on the provided masks.

Example 1: Request the list of public screen views with response in XML

```
http://192.168.0.1:8601/Interface/ScreenViews/GetPublicScreenViews?
ResponseFormat=XML
```

Example 2: Request the list of public screen view for the screenstyle 9983 and response in text

```
http://192.168.0.1:8601/Interface/ScreenViews/GetPublicScreenViews?
ScreenStyleID=9983&ResponseFormat=Text
```

Example 3: Request the list of public screen views for screenstyle 9983 that starts with letter "A", with response in text

```
http://192.168.0.1:8601/Interface/ScreenViews/GetPublicScreenViews?
ScreenStyleID=9983&ScreenViews=A*&ResponseFormat=Text
```

Response:

A list with all of the public screen views is returned

HTTP Return: 200 OK

Parameters of return:**Fixed parameters:**

Parameter	Type	Description
COUNT	Integer	Total number of screen views (Not included in JSON response)

Parameters of the list of screen views:

Parameter	Type	Description
NAME	String	Name of the screen view
SCREENSTYLEID	Integer	ID of the screenstyle of the screen view

Example of return in text:

```
RESPONSE_CODE=0
RESPONSE_MESSAGE=OK
COUNT=3
SCREENVIEW_1_NAME=View 1
SCREENVIEW_1_SCREENSTYLEID=6278
SCREENVIEW_2_NAME=View 2
SCREENVIEW_2_SCREENSTYLEID=9983
SCREENVIEW_3_NAME=View 3
SCREENVIEW_3_SCREENSTYLEID=9983
```

Example of return in XML:

```
<Response>
  <Code>0</Code>
  <Message>OK</Message>
  <Data>
    <ScreenViews>
      <Count>3</Count>
      <ScreenView>
        <Name>View 1</Name>
        <ScreenStyleID>6278</ScreenStyleID>
      </ScreenView>
      <ScreenView>
        <Name>View 2</Name>
        <ScreenStyleID>9983</ScreenStyleID>
      </ScreenView>
      <ScreenView>
        <Name>View 3</Name>
        <ScreenStyleID>9983</ScreenStyleID>
      </ScreenView>
    </ScreenViews>
  </Data>
</Response>
```

Example of return in JSON:

```
{
  "Response": {
    "Code": 0,
    "Message": "OK",
    "Data": {
      "ScreenViews": [
        {
          "Name": "View 1",
          "ScreenStyleID": 6278
        },
        {
          "Name": "View 2",
          "ScreenStyleID": 9983
        },
        {
          "Name": "View 3",
          "ScreenStyleID": 9983
        }
      ]
    }
  }
}
```

4.7.3 Reading the contents of an user screen view

This command will read and return the contents of a saved user screen view

Compatibility: All editions

Security level: Requires user authentication

Method: HTTP GET

Syntax:

```
http://<server_address>/Interface/ScreenViews/ReadUserScreenView?  
&argument=value [&argument=value] ... [&<general_argument>...]
```

Arguments:

Argument	Valid values	Description
ScreenStyleID*	Integer	ID of the Screen Style
ScreenView*	String	Name of the screen view to read

* Mandatory parameters

Example 1: Reading the screen view Entrance from Screen Style 6278 and response in XML

```
http://192.168.0.1:8601/Interface/ScreenViews/ReadUserScreenView?  
ScreenStyleID=6278&ScreenView=Entrance&ResponseFormat=XML
```

Example 2: Request the screen view Exit from Screen Style 6278 and response in text

```
http://192.168.0.1:8601/Interface/ScreenViews/ReadUserScreenView?  
ScreenStyleID=6278&ScreenView=Exit&ResponseFormat=Text
```

Response:

A list with all of the objects inside the screen view is returned

HTTP Return: 200 OK

Parameters of return:

Screen View parameters:

Parameter	Type	Description
NAME	String	Name of the screen view
SCREENSTYLE	Integer	ID of the screen style for the view

Objects parameters:

Parameter	Type	Description
COUNT	Integer	Total number of objects in the view (Not included in JSON response)

Parameters of the list of objects:

Parameter	Type	Description
NAME	String	Name of the object
SERVER	String	ID of the server which the object belongs to
TYPE	Integer	Type of the object. Consult the system objects type section to verify the object type.

Example of return in text:

```
RESPONSE_CODE=0
RESPONSE_MESSAGE=OK
NAME=Entrance
SCREENSTYLE=6278
COUNT=4
OBJECT_1_NAME=Entrance
OBJECT_1_SERVER=1D1F5CADE44029D452B41EA966E6AB56
OBJECT_1_TYPE=13
OBJECT_2_NAME=Entrance
OBJECT_2_SERVER=1D1F5CADE44029D452B41EA966E6AB56
OBJECT_2_TYPE=1
OBJECT_3_NAME=Entrance
OBJECT_3_SERVER=D8CAD018B6443ADA4DFB654ADE9AC23F
OBJECT_3_TYPE=12
OBJECT_4_NAME=Entrance
OBJECT_4_SERVER=D8CAD018B6443ADA4DFB654ADE9AC23F
OBJECT_4_TYPE=9
```

Example of return in XML:

```
<Response>
  <Code>0</Code>
  <Message>OK</Message>
  <Data>
    <ScreenView>
      <Name>Entrance</Name>
      <ScreenStyle>6278</ScreenStyle>
      <Objects>
        <Count>4</Count>
        <Object>
          <Name>Entrance</Name>
          <Server>1D1F5CADE44029D452B41EA966E6AB56</Server>
          <Type>13</Type>
        </Object>
        <Object>
          <Name>Entrance</Name>
          <Server>1D1F5CADE44029D452B41EA966E6AB56</Server>
          <Type>1</Type>
        </Object>
        <Object>
          <Name>Entrance</Name>
          <Server>1D1F5CADE44029D452B41EA966E6AB56</Server>
          <Type>12</Type>
        </Object>
        <Object>
          <Name>Entrance</Name>
          <Server>1D1F5CADE44029D452B41EA966E6AB56</Server>
          <Type>9</Type>
        </Object>
      </Objects>
    </ScreenView>
  </Data>
</Response>
```

Example of return in XML:

```
{
  "Response": {
    "Code": 0,
    "Message": "OK",
    "Data": {
      "ScreenView": {
        "Name": "Entrance",
        "ScreenStyle": 6278,
        "Objects": [
          {
            "Name": "Entrance",
            "Server": "1D1F5CADE44029D452B41EA966E6AB56",
            "Type": 13
          },
          {
            "Name": "Entrance",
            "Server": "1D1F5CADE44029D452B41EA966E6AB56",
            "Type": 1
          },
          {
            "Name": "Entrance",
            "Server": "1D1F5CADE44029D452B41EA966E6AB56",
            "Type": 12
          },
          {
            "Name": "Entrance",
            "Server": "1D1F5CADE44029D452B41EA966E6AB56",
            "Type": 9
          }
        ]
      }
    }
  }
}
```

4.7.4 Reading the contents of a public screen view

This command will read and return the contents of a saved public screen view

Compatibility: All editions

Security level: Requires user authentication

Method: HTTP GET

Syntax:

```
http://<server_address>/Interface/ScreenViews/ReadPublicScreenView?  
<argument=value>[&<argument=value>...][&<general_argument>...]
```

Arguments:

Argument	Valid values	Description
ScreenStyleID*	Integer	ID of the Screen Style
ScreenView*	String	Name of the screen view to read

* Mandatory parameters

Example 1: Reading the screen view Entrance from Screen Style 6278 and response in XML

```
http://192.168.0.1:8601/Interface/ScreenViews/ReadPublicScreenView?
ScreenStyleID=6278&ScreenView=Entrance&ResponseFormat=XML
```

Example 2: Request the screen view Exit from Screen Style 6278 and response in text

```
http://192.168.0.1:8601/Interface/ScreenViews/ReadPublicScreenView?
ScreenStyleID=6278&ScreenView=Exit&ResponseFormat=Text
```

Response:

A list with all of the objects inside the screen view is returned

HTTP Return: 200 OK

Parameters of return:

Screen View parameters:

Parameter	Type	Description
NAME	String	Name of the screen view
SCREENSTYLE	Integer	ID of the screen style for the view

Objects parameters:

Parameter	Type	Description
COUNT	Integer	Total number of objects in the view (Not included in JSON response)

Parameters of the list of objects:

Parameter	Type	Description
NAME	String	Name of the object
SERVER	String	ID of the server which the object belongs to
TYPE	Integer	Type of the object. Consult the system objects type section to verify the object type.

Example of return in text:

```
RESPONSE_CODE=0
RESPONSE_MESSAGE=OK
NAME=Entrance
SCREENSTYLE=6278
COUNT=4
OBJECT_1_NAME=Entrance
OBJECT_1_SERVER=1D1F5CADE44029D452B41EA966E6AB56
OBJECT_1_TYPE=13
OBJECT_2_NAME=Entrance
OBJECT_2_SERVER=1D1F5CADE44029D452B41EA966E6AB56
OBJECT_2_TYPE=1
OBJECT_3_NAME=Entrance
OBJECT_3_SERVER=D8CAD018B6443ADA4DFB654ADE9AC23F
OBJECT_3_TYPE=12
OBJECT_4_NAME=Entrance
OBJECT_4_SERVER=D8CAD018B6443ADA4DFB654ADE9AC23F
OBJECT_4_TYPE=9
```

Example of return in XML:

```
<Response>
  <Code>0</Code>
  <Message>OK</Message>
  <Data>
    <ScreenView>
      <Name>Entrance</Name>
      <ScreenStyle>6278</ScreenStyle>
      <Objects>
        <Count>4</Count>
        <Object>
          <Name>Entrance</Name>
          <Server>1D1F5CADE44029D452B41EA966E6AB56</Server>
          <Type>13</Type>
        </Object>
        <Object>
          <Name>Entrance</Name>
          <Server>1D1F5CADE44029D452B41EA966E6AB56</Server>
          <Type>1</Type>
        </Object>
        <Object>
          <Name>Entrance</Name>
          <Server>1D1F5CADE44029D452B41EA966E6AB56</Server>
          <Type>12</Type>
        </Object>
        <Object>
          <Name>Entrance</Name>
          <Server>1D1F5CADE44029D452B41EA966E6AB56</Server>
          <Type>9</Type>
        </Object>
      </Objects>
    </ScreenView>
  </Data>
</Response>
```

Example of return in XML:

```
{
  "Response": {
    "Code": 0,
    "Message": "OK",
    "Data": {
      "ScreenView": {
        "Name": "Entrance",
        "ScreenStyle": 6278,
        "Objects": [
          {
            "Name": "Entrance",
            "Server": "1D1F5CADE44029D452B41EA966E6AB56",
            "Type": 13
          },
          {
            "Name": "Entrance",
            "Server": "1D1F5CADE44029D452B41EA966E6AB56",
            "Type": 1
          },
          {
            "Name": "Entrance",
            "Server": "1D1F5CADE44029D452B41EA966E6AB56",
            "Type": 12
          },
          {
            "Name": "Entrance",
            "Server": "1D1F5CADE44029D452B41EA966E6AB56",
            "Type": 9
          }
        ]
      }
    }
  }
}
```

4.8 Maps

4.8.1 Requesting the list of maps

Requests the list of maps over which the user has rights to access.

Compatibility: Standard, Professional, Enterprise

Security level: Requires user authentication

Method: HTTP GET

Syntax:

```
http://<server_address>/Interface/Maps/GetMaps [ ?<argument=value>
[ &<argument=value>... ] ] [ &<general argument>... ]
```

Arguments:

Argument	Valid values	Description
----------	--------------	-------------

Maps	APIMasks	Mask to filter the results. Specify which maps must be returned based on the provided masks.												
Fields	Name Description Active Latitude Longitude	<p>Specifies the list of desired fields. If this parameter is omitted, all of the fields will be sent.</p> <p>The fields must be separated by commas</p> <table border="1"> <thead> <tr> <th>Name</th><th>Description</th></tr> </thead> <tbody> <tr> <td>Name</td><td>Name of the map</td></tr> <tr> <td>Description</td><td>Description of the map</td></tr> <tr> <td>Active</td><td>Map activated / deactivated</td></tr> <tr> <td>Latitude</td><td>Configured map latitude</td></tr> <tr> <td>Longitude</td><td>Configured map longitude</td></tr> </tbody> </table>	Name	Description	Name	Name of the map	Description	Description of the map	Active	Map activated / deactivated	Latitude	Configured map latitude	Longitude	Configured map longitude
Name	Description													
Name	Name of the map													
Description	Description of the map													
Active	Map activated / deactivated													
Latitude	Configured map latitude													
Longitude	Configured map longitude													

Example 1: Request the list of maps with all of the fields and response in XML

```
http://192.168.0.1:8601/Interface/Maps/GetMaps?ResponseFormat=XML
```

Example 2: Request the list of maps with all of the fields and response in text

```
http://192.168.0.1:8601/Interface/Maps/GetMaps?ResponseFormat=Text
```

Example 3: Request the list of maps with only name, response in XML and authentication of the user Admin

```
http://192.168.0.1:8601/Interface/Maps/GetMaps?Fields=Name&ResponseFormat=XML&AuthUser=admin&AuthPass=pass
```

Example 4: Request the list of maps starting with "A", with only name, response in XML and authentication with Admin user

```
http://192.168.0.1:8601/Interface/Maps/GetMaps?Maps=A*&Fields=Name&ResponseFormat=XML&AuthUser=admin&AuthPass=pass
```

Response:

A list with all of the maps over which the user has rights to access is returned. The fields returned in the list will depend on the values informed in the argument Fields

HTTP Return: 200 OK

Parameters of return:

Fixed parameters:

Parameter	Type	Description
COUNT	Integer	Total number of maps (Not included in JSON response)

Parameters of the list of maps:

Parameter	Type	Description
NAME	String	Name of the map
DESCRIPTION	String	Description of the map
ACTIVE	Boolean	Map activated / deactivated
LATITUDE	APILatLng	Map latitude with 6 digit precision
LONGITUDE	APILatLng	Map longitude with 6 digit precision

Example of return in text:

```
RESPONSE_CODE=0
RESPONSE_MESSAGE=OK
COUNT=2
MAP_1_NAME=Map1
MAP_1_DESCRIPTION=Map 1
MAP_1_ACTIVE=TRUE
MAP_1_LATITUDE=0.000000
MAP_1_LONGITUDE=0.000000
MAP_2_NAME=Map2
MAP_2_DESCRIPTION=Map 2
MAP_2_ACTIVE=FALSE
MAP_2_LATITUDE=-23.630363
MAP_2_LONGITUDE=-46.554916
```

Example of return in XML:

```
<Response>
  <Code>0</Code>
  <Message>OK</Message>
  <Data>
    <Maps>
      <Count>2</Count>
      <Map>
        <Name>Map1</Name>
        <Description>Map 1</Description>
        <Active>TRUE</Active>
        <Latitude>0.000000</Latitude>
        <Longitude>0.000000</Longitude>
      </Map>
      <Map>
        <Name>Map2</Name>
        <Description>Map 2</Description>
        <Active>FALSE</Active>
        <Latitude>-23.630363</Latitude>
        <Longitude>-46.554916</Longitude>
      </Map>
    </Maps>
  </Data>
</Response>
```

Example of return in JSON:

```
{
  "Response": {
    "Code": 0,
    "Message": "OK",
    "Data": {
      "Maps": [
        {
          "Name": "Map1",
          "Description": "Map 1",
          "Active": true,
          "Latitude": "0.000000",
          "Longitude": "0.000000"
        },
        {
          "Name": "Map1",
          "Description": "Map 1",
          "Active": false,
          "Latitude": "-23.630363",
          "Longitude": "-46.554916"
        }
      ]
    }
  }
}
```

4.8.2 Activating / Deactivating maps

Allows the activation or deactivation of multiple maps simultaneously

Compatibility: Standard, Professional, Enterprise

Security level: Requires user authentication with rights to configure Maps

Method: HTTP GET

Syntax:

```
http://<server_address>/Interface/Maps/Activation?<argument=value>
[&<argument=value>...] [&<general_argument>...]
```

Arguments:

Argument	Valid values	Description								
Maps*	String	<p>List of maps to activate or deactivate.</p> <p>The map names must be separated by commas.</p> <p>This command is only mandatory when Action=Activate or Action=Deactivate</p>								
Action*	Activate Deactivate ActivateAll DeactivateAll	<p>Action to be executed</p> <table border="1"> <thead> <tr> <th>Operation</th><th>Description</th></tr> </thead> <tbody> <tr> <td>Activate</td><td>Activate the maps on Maps argument</td></tr> <tr> <td>Deactivate</td><td>Deactivate the maps on Maps argument</td></tr> <tr> <td>ActivateAll</td><td>Activate all maps from the server</td></tr> </tbody> </table>	Operation	Description	Activate	Activate the maps on Maps argument	Deactivate	Deactivate the maps on Maps argument	ActivateAll	Activate all maps from the server
Operation	Description									
Activate	Activate the maps on Maps argument									
Deactivate	Deactivate the maps on Maps argument									
ActivateAll	Activate all maps from the server									

		DeactivateAll	Deactivate all maps from the server
--	--	---------------	-------------------------------------

* Mandatory parameters

Example 1: Activate all maps from the server

```
http://192.168.0.1:8601/Interface/Maps/Activation?Action=ActivateAll
```

Example 2: Activate maps Map1 and Map2

```
http://192.168.0.1:8601/Interface/Maps/Activation?Action=Activate&Maps=Map1,Map2
```

Example 3: Deactivate all maps from the server

```
http://192.168.0.1:8601/Interface/Maps/Activation?Action=DeactivateAll
```

Response:

Default response of API.

HTTP Return HTTP: 200 OK

Parameters of return: [Default return of API](#)

4.9 Operational Maps

4.9.1 Requesting the list of Operational Maps

Requests the list of Operational Maps over which the user has rights to access.

Compatibility: Professional, Enterprise

Security level: Requires user authentication

Method: HTTP GET

Syntax:

```
http://<server_address>/Interface/OperationalMaps/GetOperationalMaps  
[?<argument=value> [&<argument=value>...]] [ &<general\_argument>... ]
```

Arguments:

Argument	Valid values	Description										
OperationalMaps	APIMasks	Mask to filter the results. Specify which Operational Maps must be returned based on the provided masks. Both DUID and Name will be compared with mask.										
Fields	DUID Name Description Active	Specifies the list of desired fields. If this parameter is omitted, all of the fields will be sent. The fields must be separated by commas <table border="1" data-bbox="750 1721 1395 1890"> <thead> <tr> <th>Name</th><th>Description</th></tr> </thead> <tbody> <tr> <td>DUID</td><td>DUID of the map</td></tr> <tr> <td>Name</td><td>Name of the map</td></tr> <tr> <td>Description</td><td>Description of the map</td></tr> <tr> <td>Active</td><td>Map activated / deactivated</td></tr> </tbody> </table>	Name	Description	DUID	DUID of the map	Name	Name of the map	Description	Description of the map	Active	Map activated / deactivated
Name	Description											
DUID	DUID of the map											
Name	Name of the map											
Description	Description of the map											
Active	Map activated / deactivated											

Example 1: Request the list of Operational Maps with all of the fields and response in XML

```
http://192.168.0.1:8601/Interface/OperationalMaps/GetOperationalMaps?
ResponseFormat=XML
```

Example 2: Request the list of Operational Maps with all of the fields and response in text

```
http://192.168.0.1:8601/Interface/OperationalMaps/GetOperationalMaps?
ResponseFormat=Text
```

Example 3: Request the list of Operational Maps with only name, response in XML and authentication of the user Admin

```
http://192.168.0.1:8601/Interface/OperationalMaps/GetOperationalMaps?
Fields=Name&ResponseFormat=XML&AuthUser=admin&AuthPass=pass
```

Example 4: Request the list of Operational Maps starting with "A", with only name, response in XML and authentication with Admin user

```
http://192.168.0.1:8601/Interface/OperationalMaps/GetOperationalMaps?
OperationalMaps=A*&Fields=Name&ResponseFormat=XML&AuthUser=admin&
AuthPass=pass
```

Response:

A list with all of the Operational Maps over which the user has rights to access is returned. The fields returned in the list will depend on the values informed in the argument `Fields`

HTTP Return: 200 OK

Parameters of return:**Fixed parameters:**

Parameter	Type	Description
COUNT	Integer	Total number of Operational Maps (Not included in JSON response)

Parameters of the list of maps:

Parameter	Type	Description
DUID	DUID	DUID of the Operational Map
NAME	String	Name of the Operational Map
DESCRIPTION	String	Description of the Operational Map
ACTIVE	Boolean	Map activated / deactivated

Example of return in text:

```
RESPONSE_CODE=0
RESPONSE_MESSAGE=OK
COUNT=2
OPERATIONALMAP_1_DUID=17C4670E-82BD-44CA-A902-C255B2E721E3
OPERATIONALMAP_1_NAME=Americas
OPERATIONALMAP_1_DESCRIPTION=Americas Navigational Map
OPERATIONALMAP_1_ACTIVE=TRUE
OPERATIONALMAP_2_DUID=51605AA3-EB1E-4CAA-99C0-FD237B3A0CFA
OPERATIONALMAP_2_NAME=Australia
OPERATIONALMAP_2_DESCRIPTION=Australia Map
OPERATIONALMAP_2_ACTIVE=FALSE
```

Example of return in XML:

```
<Response>
  <Code>0</Code>
  <Message>OK</Message>
  <Data>
    <OperationalMaps>
      <Count>2</Count>
      <OperationalMap>
        <DUID>17C4670E-82BD-44CA-A902-C255B2E721E3</DUID>
        <Name>Americas</Name>
        <Description>Americas Navigational Map</Description>
        <Active>TRUE</Active>
      </OperationalMap>
      <OperationalMap>
        <DUID>51605AA3-EB1E-4CAA-99C0-FD237B3A0CFA</DUID>
        <Name>Australia</Name>
        <Description>Australia Map</Description>
        <Active>FALSE</Active>
      </OperationalMap>
    </OperationalMaps>
  </Data>
</Response>
```

Example of return in JSON:

```
{
  "Response": {
    "Code": 0,
    "Message": "OK",
    "Data": {
      "OperationalMaps": [
        {
          "DUID": "17C4670E-82BD-44CA-A902-C255B2E721E3",
          "Name": "Americas",
          "Description": "Americas Navigational Map",
          "Active": true
        },
        {
          "DUID": "51605AA3-EB1E-4CAA-99C0-FD237B3A0CFA",
          "Name": "Australia",
          "Description": "Australia Map",
          "Active": false
        }
      ]
    }
  }
}
```

4.9.2 Activating / Deactivating Operational Maps

Allows the activation or deactivation of multiple Operational Maps simultaneously

Compatibility: Professional, Enterprise

Security level: Requires user authentication with rights to configure Operational Maps

Method: HTTP GET

Syntax:

```
http://<server_address>/Interface/OperationalMaps/Activation?  
<argument=value>[&<argument=value>...][&<general_argument>...]
```

Arguments:

Argument	Valid values	Description										
OperationalMaps*	String	<p>List of Operational Maps (DUIDs or Names) to activate or deactivate.</p> <p>The Operational Map DUIDs or Names must be separated by commas.</p> <p>This command is only mandatory when Action=Activate or Action=Deactivate</p>										
Action*	Activate Deactivate ActivateAll DeactivateAll	<p>Action to be executed</p> <table border="1"> <thead> <tr> <th>Operation</th><th>Description</th></tr> </thead> <tbody> <tr> <td>Activate</td><td>Activate the Operational Maps on OperationalMaps argument</td></tr> <tr> <td>Deactivate</td><td>Deactivate the Operational Maps on OperationalMaps argument</td></tr> <tr> <td>ActivateAll</td><td>Activate all Operational Maps from the server</td></tr> <tr> <td>DeactivateAll</td><td>Deactivate all Operational Maps from the server</td></tr> </tbody> </table>	Operation	Description	Activate	Activate the Operational Maps on OperationalMaps argument	Deactivate	Deactivate the Operational Maps on OperationalMaps argument	ActivateAll	Activate all Operational Maps from the server	DeactivateAll	Deactivate all Operational Maps from the server
Operation	Description											
Activate	Activate the Operational Maps on OperationalMaps argument											
Deactivate	Deactivate the Operational Maps on OperationalMaps argument											
ActivateAll	Activate all Operational Maps from the server											
DeactivateAll	Deactivate all Operational Maps from the server											

* Mandatory parameters

Example 1: Activate all Operational Maps from the server

```
http://192.168.0.1:8601/Interface/OperationalMaps/Activation?  
Action=ActivateAll
```

Example 2: Activate Operational Maps 3346826D-3CE3-4239-A56A-D81757A9C309 and 3EC95C91-E414-4B25-8D8B-D6DB68922127

```
http://192.168.0.1:8601/Interface/OperationalMaps/Activation?  
Action=Activate&OperationalMaps=3346826D-3CE3-4239-A56A-D81757A9C309,  
3EC95C91-E414-4B25-8D8B-D6DB68922127
```

Example 3: Activate Operational Maps Map1 and Map2

```
http://192.168.0.1:8601/Interface/OperationalMaps/Activation?  
Action=Activate&OperationalMaps=Map1,Map2
```

Example 4: Deactivate all Operational Maps from the server

```
http://192.168.0.1:8601/Interface/OperationalMaps/Activation?  
Action=DeactivateAll
```

Response:

Default response of API.

HTTP Return HTTP: 200 OK

Parameters of return: [Default return of API](#)

4.9.3 Requesting the list of Custom Objects

Requests the list of Operational Maps Custom Objects over which the user has rights to access.

Compatibility: Professional, Enterprise

Security level: Requires user authentication

Method: HTTP GET

Syntax:

```
http://<server_address>/Interface/OperationalMaps/CustomObjects/
GetCustomObjects[?<argument=value>[&<argument=value>...]]
[&<general\_argument>...]
```

Arguments:

Argument	Valid values	Description																
CustomObjects	APIMasks	Mask to filter the results. Specify which Custom Objects must be returned based on the provided masks. Both DUID and Name will be compared with mask.																
Fields	DUID Name Description Memo Latitude Longitude Active	<p>Specifies the list of desired fields. If this parameter is omitted, all of the fields will be sent.</p> <p>The fields must be separated by commas</p> <table border="1"> <thead> <tr> <th>Name</th><th>Description</th></tr> </thead> <tbody> <tr> <td>DUID</td><td>DUID of the object</td></tr> <tr> <td>Name</td><td>Name of the object</td></tr> <tr> <td>Description</td><td>Description of the object</td></tr> <tr> <td>Memo</td><td>General memo for the object</td></tr> <tr> <td>Latitude</td><td>Object latitude</td></tr> <tr> <td>Longitude</td><td>Object longitude</td></tr> <tr> <td>Active</td><td>Object activated / deactivated</td></tr> </tbody> </table>	Name	Description	DUID	DUID of the object	Name	Name of the object	Description	Description of the object	Memo	General memo for the object	Latitude	Object latitude	Longitude	Object longitude	Active	Object activated / deactivated
Name	Description																	
DUID	DUID of the object																	
Name	Name of the object																	
Description	Description of the object																	
Memo	General memo for the object																	
Latitude	Object latitude																	
Longitude	Object longitude																	
Active	Object activated / deactivated																	

Example 1: Request the list of Operational Map Custom Objects with all of the fields and response in XML

```
http://192.168.0.1:8601/Interface/OperationalMaps/CustomObjects/
GetCustomObjects?ResponseFormat=XML
```

Example 2: Request the list of Operational Map Custom Objects with all of the fields and response in text

```
http://192.168.0.1:8601/Interface/OperationalMaps/CustomObjects/
GetCustomObjects?ResponseFormat=Text
```

Example 3: Request the list of Operational Map Custom Objects with only name, response in XML and authentication of the user Admin

```
http://192.168.0.1:8601/Interface/OperationalMaps/CustomObjects/
GetCustomObjects?Fields=Name&ResponseFormat=XML&AuthUser=admin&
AuthPass=pass
```

Example 4: Request the list of Operational Map Custom Objects starting with "A", with only name, response in XML and authentication with Admin user

```
http://192.168.0.1:8601/Interface/OperationalMaps/CustomObjects/
GetCustomObjects?CustomObjects=A*&Fields=Name&ResponseFormat=XML&
AuthUser=admin&AuthPass=pass
```

Response:

A list with all of the Operational Map Custom Objects over which the user has rights to access is returned. The fields returned in the list will depend on the values informed in the argument `Fields`

HTTP Return: 200 OK

Parameters of return:

Fixed parameters:

Parameter	Type	Description
COUNT	Integer	Total number of Custom Objects (Not included in JSON response)

Parameters of the list of Custom Objects:

Parameter	Type	Description
DUID	DUID	DUID of the Custom Object
NAME	String	Name of the Custom Object
DESCRIPTION	String	Description of the Custom Object
MEMO	String	Object memo in Base64 format. Base64 format is used to properly encode special characters and line breaks that might be added to the memo by the user
LATITUDE	APILatLng	Object latitude with 6 digit precision
LONGITUDE	APILatLng	Object longitude with 6 digit precision
ACTIVE	Boolean	Custom Object activated / deactivated

Example of return in text:

```
RESPONSE_CODE=0
RESPONSE_MESSAGE=OK
COUNT=2
CUSTOMOBJECT_1_DUID=17C4670E-82BD-44CA-A902-C255B2E721E3
CUSTOMOBJECT_1_NAME=Police Car 1
CUSTOMOBJECT_1_DESCRIPTION=First Police Car
CUSTOMOBJECT_1_MEMO=
CUSTOMOBJECT_1_LATITUDE=-14.604847
CUSTOMOBJECT_1_LONGITUDE=-49.570313
CUSTOMOBJECT_1_ACTIVE=TRUE
CUSTOMOBJECT_2_DUID=51605AA3-EB1E-4CAA-99C0-FD237B3A0CFA
CUSTOMOBJECT_2_NAME=Ambulance 1
CUSTOMOBJECT_2_DESCRIPTION=First Ambulance
CUSTOMOBJECT_2_MEMO=
CUSTOMOBJECT_2_LATITUDE=-19.023453
CUSTOMOBJECT_2_LONGITUDE=-41.287212
CUSTOMOBJECT_2_ACTIVE=FALSE
```

Example of return in XML:

```
<Response>
  <Code>0</Code>
  <Message>OK</Message>
  <Data>
    <CustomObjects>
      <Count>2</Count>
      <CustomObject>
        <DUID>17C4670E-82BD-44CA-A902-C255B2E721E3</DUID>
        <Name>Police Car 1</Name>
        <Description>First Police Car</Description>
        <Memo/>
        <Active>TRUE</Active>
        <Latitude>-14.604847</Latitude>
        <Longitude>-49.570313</Longitude>
      </CustomObject>
      <CustomObject>
        <DUID>51605AA3-EB1E-4CAA-99C0-FD237B3A0CFA</DUID>
        <Name>Ambulance 1</Name>
        <Description>First Ambulance</Description>
        <Memo/>
        <Latitude>-19.023453</Latitude>
        <Longitude>-41.287212</Longitude>
        <Active>FALSE</Active>
      </CustomObject>
    </CustomObjects>
  </Data>
</Response>
```

Example of return in JSON:

```
{
  "Response": {
    "Code": 0,
    "Message": "OK",
    "Data": {
      "CustomObjects": [
        {
          "DUID": "17C4670E-82BD-44CA-A902-C255B2E721E3",
          "Name": "Police Car 1",
          "Description": "First Police Car",
          "Memo": "",
          "Latitude": "-14.604847",
          "Longitude": "-49.570313",
          "Active": true
        },
        {
          "DUID": "51605AA3-EB1E-4CAA-99C0-FD237B3A0CFA",
          "Name": "Ambulance 1",
          "Description": "First Ambulance",
          "Memo": "",
          "Latitude": "-19.023453",
          "Longitude": "-41.287212",
          "Active": false
        }
      ]
    }
  }
}
```

4.9.4 Updating coordinates and memo of a Custom Object

Update the coordinates and/or memo of an Operational Map Custom Object

Compatibility: Professional, Enterprise

Security level: Requires user authentication with rights to configure Operational Maps

Method: HTTP GET

Syntax:

```
http://<server_address>/Interface/OperationalMaps/CustomObjects/
UpdateCustomObject?<argument=value>[&<argument=value>...]
[&<general_argument>...]
```

Arguments:

Argument	Valid values	Description
CustomObject*	String	DUID of the Custom Object to Update
Memo	String	Memo data in Base64 format
Latitude+	APILatLng	Object Latitude
Longitude+	APILatLng	Object Longitude

* Mandatory parameters

+ Parameters are mutually dependent

Example 1: Update Latitude and Longitude of Custom Object 3346826D-3CE3-4239-A56A-D81757A9C309

```
http://192.168.0.1:8601/Interface/OperationalMaps/CustomObjects/
UpdateCustomObject?CustomObject=C2C0FCC9-EE02-437A-A096-5BC37C18D8DC&
Latitude=-14.604847&Longitude=-49.570313
```

Example 2: Update Memo of Custom Object 3346826D-3CE3-4239-A56A-D81757A9C309

```
http://192.168.0.1:8601/Interface/OperationalMaps/CustomObjects/
UpdateCustomObject?CustomObject=C2C0FCC9-EE02-437A-A096-5BC37C18D8DC&
Memo=TmV3IE1lbW8=
```

Response:

Default response of API.

HTTP Return HTTP: 200 OK

Parameters of return: [Default return of API](#)

4.9.5 Activating / Deactivating Custom Objects

Allows the activation or deactivation of multiple Custom Objects simultaneously

Compatibility: Professional, Enterprise

Security level: Requires user authentication with rights to configure Operational Maps

Method: HTTP GET

Syntax:

```
http://<server_address>/Interface/OperationalMaps/CustomObjects/Activation?
<argument=value>[&<argument=value>...][&<general_argument>...]
```

Arguments:

Argument	Valid values	Description										
CustomObjects*	String	<p>List of Custom Objects (DUIDs or Names) to activate or deactivate.</p> <p>The Custom Object DUIDs or Names must be separated by commas.</p> <p>This command is only mandatory when Action=Activate or Action=Deactivate</p>										
Action*	Activate Deactivate ActivateAll DeactivateAll	<p>Action to be executed</p> <table border="1"> <thead> <tr> <th>Operation</th><th>Description</th></tr> </thead> <tbody> <tr> <td>Activate</td><td>Activate the Custom Objects on CustomObjects argument</td></tr> <tr> <td>Deactivate</td><td>Deactivate the Custom Objects on CustomObjects argument</td></tr> <tr> <td>ActivateAll</td><td>Activate all Custom Objects from the server</td></tr> <tr> <td>DeactivateAll</td><td>Deactivate all Custom Objects from the</td></tr> </tbody> </table>	Operation	Description	Activate	Activate the Custom Objects on CustomObjects argument	Deactivate	Deactivate the Custom Objects on CustomObjects argument	ActivateAll	Activate all Custom Objects from the server	DeactivateAll	Deactivate all Custom Objects from the
Operation	Description											
Activate	Activate the Custom Objects on CustomObjects argument											
Deactivate	Deactivate the Custom Objects on CustomObjects argument											
ActivateAll	Activate all Custom Objects from the server											
DeactivateAll	Deactivate all Custom Objects from the											

		server
--	--	--------

* Mandatory parameters

Example 1: Activate all Custom Objects from the server

```
http://192.168.0.1:8601/Interface/OperationalMaps/CustomObjects/Activation?
Action=ActivateAll
```

Example 2: Activate Custom Objects 3346826D-3CE3-4239-A56A-D81757A9C309 and 3EC95C91-E414-4B25-8D8B-D6DB68922127

```
http://192.168.0.1:8601/Interface/OperationalMaps/CustomObjects/Activation?
Action=Activate&CustomObjects=3346826D-3CE3-4239-A56A-D81757A9C309,
3EC95C91-E414-4B25-8D8B-D6DB68922127
```

Example 3: Activate Custom Objects Object1 and Object2

```
http://192.168.0.1:8601/Interface/OperationalMaps/CustomObjects/Activation?
Action=Activate&CustomObjects=Object1, Object2
```

Example 4: Deactivate all Custom Objects from the server

```
http://192.168.0.1:8601/Interface/OperationalMaps/CustomObjects/Activation?
Action=DeactivateAll
```

Response:

Default response of API.

HTTP Return HTTP: 200 OK

Parameters of return: [Default return of API](#)

4.10 Global Events

4.10.1 Requesting the list of Global Events

Requests the list of global events over which the user has rights to access.

Compatibility: Standard, Professional, Enterprise

Security level: Requires user authentication

Method: HTTP GET

Syntax:

```
http://<server_address>/Interface/GlobalEvents/GetGlobalEvents
[?<argument=value> [&<argument=value>...]] [&<general\_argument>...]
```

Arguments:

Argument	Valid values	Description
GlobalEvents	APIMasks	Mask to filter the results. Specify which global events must be returned based on the provided masks.
Fields	DUID Name Description Active	Specifies the list of desired fields. If this parameter is omitted, all of the fields will be sent. The fields must be separated by commas

	Latitude Longitude Actions	Name	Description
DUID	DUID of the global event		
Name	Name of the global event		
Description	Description of the global event		
Active	Global event activated / deactivated		
Latitude	Configured event latitude		
Longitude	Configured event longitude		
Actions	List of configured actions		

Example 1: Requests the list of global events with all of the fields and response in XML

```
http://192.168.0.1:8601/Interface/GlobalEvents/GetGlobalEvents?
ResponseFormat=XML
```

Example 2: Requests the list of global events with all of the fields and response in text

```
http://192.168.0.1:8601/Interface/GlobalEvents/GetGlobalEvents?
ResponseFormat=Text
```

Example 3: Requests the list of global events with only name, response in XML and authentication of the user Admin

```
http://192.168.0.1:8601/Interface/GlobalEvents/GetGlobalEvents?Fields=Name&
ResponseFormat=XML&AuthUser=admin&AuthPass=pass
```

Example 4: Request the list of global events starting with "A", with only name, response in XML and authentication with Admin user

```
http://192.168.0.1:8601/Interface/GlobalEvents/GetGlobalEvents?
GlobalEvents=A*&Fields=Name&ResponseFormat=XML&AuthUser=admin&
AuthPass=pass
```

Response:

A list with all of the global events over which the user has rights of accessing is returned. The fields returned in the list will depend on the values informed in the argument `Fields`

HTTP Return: 200 OK

Parameters of return:

Fixed parameters:

Parameter	Type	Description
COUNT	Integer	Total number of global events (Not included in JSON response)

Parameters of the list of global events:

Parameter	Type	Description
DUID	DUID	DUID of the global event
NAME	String	Name of the global event
DESCRIPTION	String	Description of the global event
ACTIVE	Boolean	Global event activated / deactivated
LATITUDE	APILatLnq	Event latitude with 6 digit precision
LONGITUDE	APILatLnq	Event longitude with 6 digit precision
ACTIONS	Event Actions	List of configured Event Actions. Please refer to Event Actions section for the format of the list.

Example of return in text:

```
RESPONSE_CODE=0
RESPONSE_MESSAGE=OK
COUNT=2
GLOBALEVENT_1_DUID=3CBCF7E2-59FB-4C11-8635-60FC1C4F6104
GLOBALEVENT_1_NAME=Event1
GLOBALEVENT_1_DESCRIPTION=Event 1
GLOBALEVENT_1_ACTIVE=TRUE
GLOBALEVENT_1_LATITUDE=0.000000
GLOBALEVENT_1_LONGITUDE=0.000000
GLOBALEVENT_1_ACTIONS_SENDEMAIL_ACTIVE=TRUE
GLOBALEVENT_1_ACTIONS_SENDEMAIL_EMAILGROUP=Administration
GLOBALEVENT_1_ACTIONS_SENDEMAIL_MESSAGETEXT=
GLOBALEVENT_1_ACTIONS_SENDEMAIL_SMSFORMAT=Default
GLOBALEVENT_2_DUID=095BFF2F-5D10-495D-8C29-C4A56EBE31DA
GLOBALEVENT_2_NAME=Event2
GLOBALEVENT_2_DESCRIPTION=Event 2
GLOBALEVENT_2_ACTIVE=FALSE
GLOBALEVENT_2_LATITUDE=-23.630363
GLOBALEVENT_2_LONGITUDE=-46.554916
GLOBALEVENT_2_ACTIONS_CALLPRESET_ACTIVE=TRUE
GLOBALEVENT_2_ACTIONS_CALLPRESET_PRESETS_COUNT=1
GLOBALEVENT_2_ACTIONS_CALLPRESET_PRESETS_PRESET_1_CAMERA=29
GLOBALEVENT_2_ACTIONS_CALLPRESET_PRESETS_PRESET_1_PRESETNUMBER=4
```

Example of return in XML:

```
<Response>
  <Code>0</Code>
  <Message>OK</Message>
  <Data>
    <GlobalEvents>
      <Count>2</Count>
      <GlobalEvent>
        <DUID>3CBCF7E2-59FB-4C11-8635-60FC1C4F6104</DUID>
        <Name>Event1</Name>
        <Description>Event 1</Description>
        <Active>TRUE</Active>
        <Latitude>0.000000</Latitude>
        <Longitude>0.000000</Longitude>
        <Actions>
          <SendEmail>
            <Active>TRUE</Active>
            <EmailGroup>Administration</EmailGroup>
            <MessageText />
            <SMSFormat>Default</SMSFormat>
          </SendEmail>
        </Actions>
      </GlobalEvent>
      <GlobalEvent>
        <DUID>095BFF2F-5D10-495D-8C29-C4A56EBE31DA</DUID>
        <Name>Event2</Name>
        <Description>Event 2</Description>
        <Active>FALSE</Active>
        <Latitude>-23.630363</Latitude>
        <Longitude>-46.554916</Longitude>
        <Actions>
          <CallPreset>
            <Active>TRUE</Active>
            <Presets>
              <Count>1</Count>
              <Preset>
                <Camera>29</Camera>
                <PresetNumber>4</PresetNumber>
              </Preset>
            </Presets>
          </CallPreset>
        </Actions>
      </GlobalEvent>
    </GlobalEvents>
  </Data>
</Response>
```

Example of return in JSON:

```
{  
    "Response": {  
        "Code": 0,  
        "Message": "OK",  
        "Data": {  
            "GlobalEvents": [  
                {  
                    "DUID": "3CBCF7E2-59FB-4C11-8635-60FC1C4F6104",  
                    "Name": "Event1",  
                    "Description": "Event 1",  
                    "Active": true,  
                    "Latitude": "0.000000",  
                    "Longitude": "0.000000",  
                    "Actions": {  
                        "SendEmail": {  
                            "Active": true,  
                            "EmailGroup": "Administration",  
                            "MessageText": "",  
                            "SMSFormat": "Default"  
                        }  
                    }  
                },  
                {  
                    "DUID": "095BFF2F-5D10-495D-8C29-C4A56EBE31DA",  
                    "Name": "Event2",  
                    "Description": "Event 2",  
                    "Active": false,  
                    "Latitude": "-23.630363",  
                    "Longitude": "-46.554916",  
                    "Actions": {  
                        "CallPreset": {  
                            "Active": true,  
                            "Presets": [  
                                {  
                                    "Camera": "29",  
                                    "PresetNumber": 4  
                                }  
                            ]  
                        }  
                    }  
                }  
            ]  
        }  
    }  
}
```

4.10.2 Triggering a Global Event

By way of this command, you will be able to trigger a global event.

Compatibility: Standard, Professional, Enterprise

Security level: Requires user authentication with rights to trigger global events

Method: HTTP GET**Syntax:**

```
http://<server_address>/Interface/GlobalEvents/TriggerGlobalEvent?  
<argument=value>[&<argument=value>...][&general\_argument>...]
```

Arguments:

Argument	Valid values	Description						
Event*	String	DUID or Name of the global event						
Message	String	Message of the global event (to be registered in the server's log)						
OverrideEmailMessage	String	Overrides an e-mail message, in case the action of e-mail transmission for the event is selected						
OverrideEmailMessageFormat	text/plain text/base64	Select the format of the message being passed on OverrideEmailMessage parameter. The following options are available: <table border="1" style="margin-left: auto; margin-right: auto;"><thead><tr><th>Name</th><th>Description</th></tr></thead><tbody><tr><td>text/plain</td><td>Message will be passed as plain text</td></tr><tr><td>text/base64</td><td>Message will be passed using base64 encoding</td></tr></tbody></table> If this parameter is omitted, the value text/plain will be used as default	Name	Description	text/plain	Message will be passed as plain text	text/base64	Message will be passed using base64 encoding
Name	Description							
text/plain	Message will be passed as plain text							
text/base64	Message will be passed using base64 encoding							
OverrideOperatorMessage	String	Overrides an instant message sent to the operators, in case the action of message transmission to the operators is selected						
OverrideOperatorMessageFormat	text/plain text/base64	Select the format of the message being passed on OverrideOperatorMessage parameter. The following options are available: <table border="1" style="margin-left: auto; margin-right: auto;"><thead><tr><th>Name</th><th>Description</th></tr></thead><tbody><tr><td>text/plain</td><td>Message will be passed as plain text</td></tr><tr><td>text/base64</td><td>Message will be passed using base64 encoding</td></tr></tbody></table> If this parameter is omitted, the value text/plain will be used as default	Name	Description	text/plain	Message will be passed as plain text	text/base64	Message will be passed using base64 encoding
Name	Description							
text/plain	Message will be passed as plain text							
text/base64	Message will be passed using base64 encoding							
OverrideShowObjects	String	Overrides the list of objects to be sent to the operator inside an alarm popup. See below how the string must be generated.						
OverrideShowPlaybackLoop	String	Overrides the list of playback cameras to be sent to the operator inside an alarm popup. See below how the string must be generated.						
OverrideShowSnapshots	String	Overrides the list of snapshots to be sent to the operator inside an alarm popup. See						

		below how the string must be generated.
OverrideLatitude	APILatLang	Override the latitude from the event and provide a new latitude value.
OverrideLongitude	APILatLang	Override the longitude from the event and provide a new longitude value.
OverrideShowCameras	String	<p>Overrides the list of cameras to be sent to the operator inside an alarm popup. This parameter should contain the name of the cameras on a comma separated way.</p> <p>This parameter will only be effective if the parameter <code>OverrideShowObjects</code> is not specified.</p> <p>OBS: This parameter is only being maintained for compatibility. Use the new parameter <code>OverrideShowObjects</code> for new developments</p>

* Mandatory parameters

OverrideShowObjects

This parameter was introduced as a replacement to the old `OverrideShowCameras` parameter, being now more comprehensive, providing the possibility to choose the objects to be sent in the popup including maps, cameras, analytics configurations and LPR configurations.

The list of objects must be formated in the following way:

```
OBJECT_ID_1;OBJECT_NAME_1,OBJECT_ID_2;OBJECT_NAME_2
```

Being, `OBJECT_ID` the ID of the object and `OBJECT_NAME` the name of the objects. The below table displays the possible values for `OBJECT_ID`:

OBJECT_ID	Description
1	Camera
9	Map
12	Analytics configuration
13	LPR configuration

For example, to show cameras Camera1, Camera2 and the maps Map1 and Map2, the list must be defined as:

```
1;Camera1,1;Camera2,9;Map1,9;Map2
```

OverrideShowSnapshots

This parameter will override the list of objects for snapshots. It uses the same structure as `OverrideShowObjects` but only supports Camera Object ID at this moment.

The list of objects must be formated in the following way:

```
OBJECT_ID_1;OBJECT_NAME_1,OBJECT_ID_2;OBJECT_NAME_2
```

Being, `OBJECT_ID` the ID of the object and `OBJECT_NAME` the name of the objects. The below table displays the possible values for `OBJECT_ID`:

OBJECT_ID	Description
1	Camera

For example, to show snapshots from Camera1 and Camera2 the list must be defined as:

```
1;Camera1,1;Camera2
```

OverrideShowPlaybackLoop

This parameter will override the list of cameras for playback loop. It uses the same structure as `OverrideShowObjects` but only supports Camera Object ID at this moment.

The list of objects to must be formated in the following way:

```
OBJECT_ID_1;OBJECT_NAME_1,OBJECT_ID_2;OBJECT_NAME_2
```

Being, `OBJECT_ID` the ID of the object and `OBJECT_NAME` the name of the objects. The below table displays the possible values for `OBJECT_ID`:

OBJECT_ID	Description
1	Camera

For example, to show snapshots from Camera1 and Camera2 the list must be defined as:

```
1;Camera1,1;Camera2
```

Example 1: Trigger a global event with the message "Test message" and response in text

```
http://192.168.0.1:8601/Interface/GlobalEvents/TriggerGlobalEvent?
Event=Event1&Message=Test message&ResponseFormat=Text
```

Example 2: Trigger a global event, overriding the message to be sent to the operators with the message "External alarm" and response in XML

```
http://192.168.0.1:8601/Interface/GlobalEvents/TriggerGlobalEvent?
Event=3CBCF7E2-59FB-4C11-8635-60FC1C4F6104&
OverrideOperatorMessage=External alarm&ResponseFormat=XML
```

Example 3: Trigger a global event, with the message "Test message", overriding the message to be sent to the operators with the message "External alarm" and response in XML

```
http://192.168.0.1:8601/Interface/GlobalEvents/TriggerGlobalEvent?
Event=Event1&Message=Test message&OverrideOperatorMessage=External alarm&
ResponseFormat=XML
```

Example 4: Trigger a global event, with the message "Test message", overriding the message to be sent to the operators with the message "External alarm" and overriding the objects Camera1, Camera2, Map1, Map2 and response in XML

```
http://192.168.0.1:8601/Interface/GlobalEvents/TriggerGlobalEvent?
Event=Event1&Message=Test message&OverrideOperatorMessage=External alarm&
OverrideShowObjects=1;Camera1,1;Camera2,9;Map1,9;Map2&ResponseFormat=XML
```

Response:

Default response of API.

HTTP Return: 200 OK

Parameters of return: [Default return of API](#)

4.10.3 Requesting the Event Actions of a Global Event

Request the detailed data on every actions of a global event

Deprecated: This command is deprecated in API version 1.9.0 and superior. The list of actions of a global event is already added to the [list of global events](#)

Compatibility: Standard, Professional, Enterprise

Security level: Requires user authentication

Method: HTTP GET

Syntax:

```
http://<server_address>/Interface/GlobalEvents/GetEventActions?
<argument=value>[&<argument=value>...][&<general_argument>...]
```

Arguments:

Argument	Valid values	Description
GlobalEvent*	String	DUID or Name of the global event to read the actions
Actions	SendEmail ShowObjects ShowSnapshots PlaySound SendMessage RequestAck SendAudioClip CallPreset ActivateIOOutput ObjectActivation HTTPRequest CreateBookmark DownloadMedia CreateTimer	Specify which actions must be included in the result. This parameter accepts multiple actions as a comma-delimited list

* Mandatory parameters

Example 1: Requests the actions of event "Event1" with response in XML

```
http://192.168.0.1:8601/Interface/GlobalEvents/GetEventActions?
GlobalEvent=Event1&ResponseFormat=XML
```

Example 2: Requests the actions SendEmail and ShowObjects of event "Event1" with response in Text

```
http://192.168.0.1:8601/Interface/GlobalEvents/GetEventActions?
GlobalEvent=Event1&Actions=SendEmail,ShowObjects&ResponseFormat=XML
```

Response:

A list with all actions configured on the event is returned. Each action has its own parameters section.

HTTP Return: 200 OK

Parameters of return: [Event Actions](#)

4.10.4 Activating / Deactivating Global Events

Allows the activation or deactivation of multiple global events simultaneously.

Note: Activating or deactivating a global event will NOT trigger the event

Compatibility: Standard, Professional, Enterprise

Security level: Requires user authentication with rights to configure Global Events

Method: HTTP GET

Syntax:

```
http://<server_address>/Interface/GlobalEvents/Activation?<argument=value>
[&<argument=value>...] [&<general\_argument>...]
```

Arguments:

Argument	Valid values	Description										
GlobalEvents*	String	<p>List of global events to activate or deactivate.</p> <p>The global event DUIDs or Names must be separated by commas.</p> <p>This command is only mandatory when Action=Activate or Action=Deactivate</p>										
Action*	Activate Deactivate ActivateAll DeactivateAll	<p>Action to be executed</p> <table border="1" style="margin-left: auto; margin-right: auto;"> <thead> <tr> <th>Operation</th><th>Description</th></tr> </thead> <tbody> <tr> <td>Activate</td><td>Activate the events on GlobalEvents argument</td></tr> <tr> <td>Deactivate</td><td>Deactivate the events on GlobalEvents argument</td></tr> <tr> <td>ActivateAll</td><td>Activate all global events from the server</td></tr> <tr> <td>DeactivateAll</td><td>Deactivate all global events from the server</td></tr> </tbody> </table>	Operation	Description	Activate	Activate the events on GlobalEvents argument	Deactivate	Deactivate the events on GlobalEvents argument	ActivateAll	Activate all global events from the server	DeactivateAll	Deactivate all global events from the server
Operation	Description											
Activate	Activate the events on GlobalEvents argument											
Deactivate	Deactivate the events on GlobalEvents argument											
ActivateAll	Activate all global events from the server											
DeactivateAll	Deactivate all global events from the server											

* Mandatory parameters

Example 1: Activate all global events from the server

```
http://192.168.0.1:8601/Interface/GlobalEvents/Activation?
Action=ActivateAll
```

Example 2: Activate global events Event1 and Event2

```
http://192.168.0.1:8601/Interface/GlobalEvents/Activation?
Action=Activate&GlobalEvents=Event1,Event2
```

Example 3: Deactivate all global events from the server

```
http://192.168.0.1:8601/Interface/GlobalEvents/Activation?
Action=DeactivateAll
```

Response:

Default response of API.

HTTP Return HTTP: 200 OK

Parameters of return: [Default return of API](#)

4.11 Scheduled Events

4.11.1 Requesting the list of scheduled events

Requests the list of scheduled events from the server.

Compatibility: Standard, Professional, Enterprise

Security level: Requires user authentication with rights to configure Scheduled Events

Method: HTTP GET

Syntax:

```
http://<server_address>/Interface/ScheduledEvents/GetScheduledEvents  
[?<argument=value>[&<argument=value>...]] [&<general\_argument>...]
```

Arguments:

Argument	Valid values	Description														
ScheduledEvents	APIMasks	Mask to filter the results. Specify which scheduled events must be returned based on the provided masks.														
Fields	Name Description Active Latitude Longitude Actions	Specifies the list of desired fields. If this parameter is omitted, all of the fields will be sent. The fields must be separated by commas <table border="1"> <thead> <tr> <th>Name</th><th>Description</th></tr> </thead> <tbody> <tr> <td>Name</td><td>Name of the scheduled event</td></tr> <tr> <td>Description</td><td>Description of the scheduled event</td></tr> <tr> <td>Active</td><td>Scheduled event activated / deactivated</td></tr> <tr> <td>Latitude</td><td>Configured event latitude</td></tr> <tr> <td>Longitude</td><td>Configured event longitude</td></tr> <tr> <td>Actions</td><td>List of configured actions</td></tr> </tbody> </table>	Name	Description	Name	Name of the scheduled event	Description	Description of the scheduled event	Active	Scheduled event activated / deactivated	Latitude	Configured event latitude	Longitude	Configured event longitude	Actions	List of configured actions
Name	Description															
Name	Name of the scheduled event															
Description	Description of the scheduled event															
Active	Scheduled event activated / deactivated															
Latitude	Configured event latitude															
Longitude	Configured event longitude															
Actions	List of configured actions															

Example 1: Requests the list of scheduled events with all of the fields and response in XML

```
http://192.168.0.1:8601/Interface/ScheduledEvents/GetScheduledEvents?  
ResponseFormat=XML
```

Example 2: Requests the list of scheduled events with all of the fields and response in text

```
http://192.168.0.1:8601/Interface/ScheduledEvents/GetScheduledEvents?  
ResponseFormat=Text
```

Example 3: Requests the list of scheduled events with only name, response in XML and authentication of the user Admin

```
http://192.168.0.1:8601/Interface/ScheduledEvents/GetScheduledEvents?  
Fields=Name&ResponseFormat=XML&AuthUser=admin&AuthPass=pass
```

Example 4: Request the list of scheduled events starting with "A", with only name, response in XML and authentication with Admin user

```
http://192.168.0.1:8601/Interface/ScheduledEvents/GetScheduledEvents?  
ScheduledEvents=A*&Fields=Name&ResponseFormat=XML&AuthUser=admin&  
AuthPass=pass
```

Response:

A list with all of the scheduled events over which the user has rights of accessing is returned. The fields returned in the list will depend on the values informed in the argument **Fields**

HTTP Return: 200 OK

Parameters of return:**Fixed parameters:**

Parameter	Type	Description
COUNT	Integer	Total number of scheduled events (Not included in JSON response)

Parameters of the list of scheduled events:

Parameter	Type	Description
NAME	String	Name of the scheduled event
DESCRIPTION	String	Description of the scheduled event
ACTIVE	Boolean	Scheduled event activated / deactivated
LATITUDE	APILatLng	Event latitude with 6 digit precision
LONGITUDE	APILatLng	Event longitude with 6 digit precision
ACTIONS	Event Actions	List of configured Event Actions. Please refer to Event Actions section for the format of the list.

Example of return in text:

```

RESPONSE_CODE=0
RESPONSE_MESSAGE=OK
COUNT=2
SCHEDULEDEVENT_1_NAME=Event1
SCHEDULEDEVENT_1_DESCRIPTION=Event 1
SCHEDULEDEVENT_1_ACTIVE=TRUE
SCHEDULEDEVENT_1_LATITUDE=0.000000
SCHEDULEDEVENT_1_LONGITUDE=0.000000
SCHEDULEDEVENT_1_ACTIONS_SENDEMAIL_ACTIVE=TRUE
SCHEDULEDEVENT_1_ACTIONS_SENDEMAIL_EMAILGROUP=Administration
SCHEDULEDEVENT_1_ACTIONS_SENDEMAIL_MESSAGETEXT=
SCHEDULEDEVENT_1_ACTIONS_SENDEMAIL_SMSFORMAT=Default
SCHEDULEDEVENT_2_NAME=Event2
SCHEDULEDEVENT_2_DESCRIPTION=Event 2
SCHEDULEDEVENT_2_ACTIVE=FALSE
SCHEDULEDEVENT_2_LATITUDE=-23.630363
SCHEDULEDEVENT_2_LONGITUDE=-46.554916
SCHEDULEDEVENT_2_ACTIONS_CALLPRESET_ACTIVE=TRUE
SCHEDULEDEVENT_2_ACTIONS_CALLPRESET_PRESETS_COUNT=1
SCHEDULEDEVENT_2_ACTIONS_CALLPRESET_PRESETS_PRESET_1_CAMERA=29
SCHEDULEDEVENT_2_ACTIONS_CALLPRESET_PRESETS_PRESET_1_PRESETNUMBER=4

```

Example of return in XML:

```
<Response>
  <Code>0</Code>
  <Message>OK</Message>
  <Data>
    <ScheduledEvents>
      <Count>2</Count>
      <ScheduledEvent>
        <Name>Event1</Name>
        <Description>Event 1</Description>
        <Active>TRUE</Active>
        <Latitude>0.000000</Latitude>
        <Longitude>0.000000</Longitude>
        <Actions>
          <SendEmail>
            <Active>TRUE</Active>
            <EmailGroup>Administration</EmailGroup>
            <MessageText />
            <SMSFormat>Default</SMSFormat>
          </SendEmail>
        </Actions>
      </ScheduledEvent>
      <ScheduledEvent>
        <Name>Event2</Name>
        <Description>Event 2</Description>
        <Active>FALSE</Active>
        <Latitude>-23.630363</Latitude>
        <Longitude>-46.554916</Longitude>
        <Actions>
          <CallPreset>
            <Active>TRUE</Active>
            <Presets>
              <Count>1</Count>
              <Preset>
                <Camera>29</Camera>
                <PresetNumber>4</PresetNumber>
              </Preset>
            </Presets>
          </CallPreset>
        </Actions>
      </ScheduledEvent>
    </ScheduledEvents>
  </Data>
</Response>
```

Example of return in JSON:

```
{
  "Response": {
    "Code": 0,
    "Message": "OK",
    "Data": {
      "ScheduledEvents": [
        {
          "Name": "Event1",
          "Description": "Event 1",
          "Active": true,
          "Latitude": "0.000000",
          "Longitude": "0.000000",
          "Actions": {
            "SendEmail": {
              "Active": true,
              "EmailGroup": "Administration",
              "MessageText": "",
              "SMSFormat": "Default"
            }
          }
        },
        {
          "Name": "Event2",
          "Description": "Event 2",
          "Active": false,
          "Latitude": "-23.630363",
          "Longitude": "-46.554916",
          "Actions": {
            "CallPreset": {
              "Active": true,
              "Presets": [
                {
                  "Camera": "29",
                  "PresetNumber": 4
                }
              ]
            }
          }
        }
      ]
    }
  }
}
```

4.11.2 Activating / Deactivating scheduled events

Allows the activation or deactivation of multiple scheduled events simultaneously.

Compatibility: Standard, Professional, Enterprise

Security level: Requires user authentication with rights to configure Scheduled Events

Method: HTTP GET

Syntax:

```
http://<server_address>/Interface/ScheduledEvents/Activation?  
<argument=value> [&<argument=value>...] [&<general argument>...]
```

Arguments:

Argument	Valid values	Description										
ScheduledEvents*	String	<p>List of scheduled events to activate or deactivate.</p> <p>The scheduled event names must be separated by commas.</p> <p>This command is only mandatory when Action=Activate or Action=Deactivate</p>										
Action*	Activate Deactivate ActivateAll DeactivateAll	<p>Action to be executed</p> <table border="1" style="margin-left: 20px;"> <thead> <tr> <th>Operation</th><th>Description</th></tr> </thead> <tbody> <tr> <td>Activate</td><td>Activate the events on ScheduledEvents argument</td></tr> <tr> <td>Deactivate</td><td>Deactivate the events on ScheduledEvents argument</td></tr> <tr> <td>ActivateAll</td><td>Activate all scheduled events from the server</td></tr> <tr> <td>DeactivateAll</td><td>Deactivate all scheduled events from the server</td></tr> </tbody> </table>	Operation	Description	Activate	Activate the events on ScheduledEvents argument	Deactivate	Deactivate the events on ScheduledEvents argument	ActivateAll	Activate all scheduled events from the server	DeactivateAll	Deactivate all scheduled events from the server
Operation	Description											
Activate	Activate the events on ScheduledEvents argument											
Deactivate	Deactivate the events on ScheduledEvents argument											
ActivateAll	Activate all scheduled events from the server											
DeactivateAll	Deactivate all scheduled events from the server											

* Mandatory parameters

Example 1: Activate all scheduled events from the server

```
http://192.168.0.1:8601/Interface/ScheduledEvents/Activation?  
Action=ActivateAll
```

Example 2: Activate scheduled events Event1 and Event2

```
http://192.168.0.1:8601/Interface/ScheduledEvents/Activation?  
Action=Activate&ScheduledEvents=Event1,Event2
```

Example 3: Deactivate all scheduled events from the server

```
http://192.168.0.1:8601/Interface/ScheduledEvents/Activation?  
Action=DeactivateAll
```

Response:

Default response of API.

HTTP Return HTTP: 200 OK

Parameters of return: [Default return of API](#)

4.12 Virtual matrix

4.12.1 Requesting the list of active monitors

Requests the list of active monitors in the virtual matrix of the server.

Compatibility: Standard, Professional, Enterprise

Security level: Requires user authentication with rights to operate the virtual matrix

Method: HTTP GET

Syntax:

```
http://<server_address>/Interface/VirtualMatrix/GetActiveMonitors
[?<argument=value> [&<argument=value>...]] [&<general\_argument>...]
```

Arguments:

Argument	Valid values	Description																				
Monitors	API Masks	Mask to filter the results. Specify which monitors must be returned based on the provided masks.																				
Fields	MonitorID ClientID ClientAddress ClientUser CurrentObjectType CurrentObjectName CurrentObjectDescription CurrentScreenStyleData CurrentScreenViewObjects	<p>Specifies the list of desired fields. If this parameter is omitted, all of the fields will be sent.</p> <p>The fields must be separated by commas</p> <table border="1"> <thead> <tr> <th>Name</th><th>Description</th></tr> </thead> <tbody> <tr> <td>MonitorID</td><td>Identification of the monitor</td></tr> <tr> <td>ClientID</td><td>Identification of the client that owns the monitor</td></tr> <tr> <td>ClientAddress</td><td>IP Address of the client that owns the monitor</td></tr> <tr> <td>ClientUser</td><td>User of the client that owns the monitor</td></tr> <tr> <td>CurrentObjectType</td><td>Type of the object currently being displayed in the monitor</td></tr> <tr> <td>CurrentObjectName</td><td>Name of the object currently being displayed in the monitor</td></tr> <tr> <td>CurrentObjectDescription</td><td>Description of the object currently being displayed in the monitor</td></tr> <tr> <td>CurrentScreenStyleData</td><td>Internal structural data of current screen style</td></tr> <tr> <td>CurrentScreenViewObjects</td><td>List of all objects being currently displayed on the monitor</td></tr> </tbody> </table>	Name	Description	MonitorID	Identification of the monitor	ClientID	Identification of the client that owns the monitor	ClientAddress	IP Address of the client that owns the monitor	ClientUser	User of the client that owns the monitor	CurrentObjectType	Type of the object currently being displayed in the monitor	CurrentObjectName	Name of the object currently being displayed in the monitor	CurrentObjectDescription	Description of the object currently being displayed in the monitor	CurrentScreenStyleData	Internal structural data of current screen style	CurrentScreenViewObjects	List of all objects being currently displayed on the monitor
Name	Description																					
MonitorID	Identification of the monitor																					
ClientID	Identification of the client that owns the monitor																					
ClientAddress	IP Address of the client that owns the monitor																					
ClientUser	User of the client that owns the monitor																					
CurrentObjectType	Type of the object currently being displayed in the monitor																					
CurrentObjectName	Name of the object currently being displayed in the monitor																					
CurrentObjectDescription	Description of the object currently being displayed in the monitor																					
CurrentScreenStyleData	Internal structural data of current screen style																					
CurrentScreenViewObjects	List of all objects being currently displayed on the monitor																					

Example 1: Request the list of active monitors with all of the fields and response in XML

```
http://192.168.0.1:8601/Interface/VirtualMatrix/GetActiveMonitors?
ResponseFormat=XML
```

Example 2: Request the list of active monitors with all of the fields and response in text

```
http://192.168.0.1:8601/Interface/VirtualMatrix/GetActiveMonitors?
ResponseFormat=Text
```

Example 3: Request the list of active monitors with only ID of the monitor, response in XML and authentication of the user Admin

```
http://192.168.0.1:8601/Interface/VirtualMatrix/GetActiveMonitors?
Fields=MonitorID&ResponseFormat=XML&AuthUser=admin&AuthPass=pass
```

Example 4: Request the list of active monitors starting with "A", with only monitor ID, response in XML and authentication with Admin user

```
http://192.168.0.1:8601/Interface/VirtualMatrix/GetActiveMonitors?
Monitors=A*&Fields=MonitorID&ResponseFormat=XML&AuthUser=admin&
AuthPass=pass
```

Response:

A list with all of the active monitors of the virtual matrix of the server is returned. The fields returned in the list will depend on the values informed in the argument Fields

HTTP Return: 200 OK

Parameters of return:

Fixed parameters:

Parameter	Type	Description
COUNT	Integer	Total number of active monitors (Not included in JSON response)

Parameters of the list of active monitors:

Parameter	Type	Description
MONITORID	String	Identification of the monitor (Configured in the Surveillance Client)
CLIENTID	String	Identification of the client that owns the monitor
CLIENTADDRESS	String	IP Address of the client that owns the monitor
CLIENTUSER	String	User of the client that owns the monitor
CURRENTOBJECTTYPE	Integer	Type of the object. Consult the system objects type section to verify the object type.
CURRENTOBJECTNAME	String	Name of the object currently being displayed in the monitor
CURRENTOBJECTDESCRIPTION	String	Description of the object currently being displayed in the monitor
CURRENTSCREENSTYLEDATA	String	Internal structural data of current screen style
CURRENTSCREENVIEWOBJECTS	List	If CURRENTOBJECTTYPE is a view (Private or Public), this section will contain a list of objects from this view

Parameters of the list of screen view objects:

Fixed Parameters	Type	Description
COUNT	Integer	Total number of active monitors (Not included in JSON response)

Parameter	Type	Description
NAME	String	Object name
SERVER	String	Server ID
TYPE	Integer	Type of the object. Consult the system objects type section to verify the object type.

Example of return in text:

```
RESPONSE_CODE=0
RESPONSE_MESSAGE=OK
COUNT=2
MONITOR_1_MONITORID=MONITOR 1
MONITOR_1_CLIENTID=AF209016277C36811CFDC95AD0D921A7
MONITOR_1_CLIENTADDRESS=192.168.0.2
MONITOR_1_CLIENTUSER=admin
MONITOR_1_CURRENTOBJECTTYPE=1
MONITOR_1_CURRENTOBJECTNAME=Camera 1
MONITOR_1_CURRENTOBJECTDESCRIPTION=Entrance camera
MONITOR_1_CURRENTSCREENSTYLEDATA=1,1
MONITOR_2_MONITORID=MONITOR 2
MONITOR_2_CLIENTID=ABF3928474CFE7E7FCABC89DBCA98378
MONITOR_2_CLIENTADDRESS=192.168.0.3
MONITOR_2_CLIENTUSER=admin
MONITOR_2_CURRENTOBJECTTYPE=16
MONITOR_2_CURRENTOBJECTNAME=View1
MONITOR_2_CURRENTOBJECTDESCRIPTION=First floor
MONITOR_2_CURRENTSCREENSTYLEDATA=3,1,2,3,4,5,6,7,8,9
MONITOR_2_CURRENTSCREENVIEWOBJECTS_COUNT=2
MONITOR_2_CURRENTSCREENVIEWOBJECTS_OBJECT_1_NAME=03
MONITOR_2_CURRENTSCREENVIEWOBJECTS_OBJECT_1_SERVER=8E18A60F508A3428346B23D5C
MONITOR_2_CURRENTSCREENVIEWOBJECTS_OBJECT_1_TYPE=1
MONITOR_2_CURRENTSCREENVIEWOBJECTS_OBJECT_2_NAME=06
MONITOR_2_CURRENTSCREENVIEWOBJECTS_OBJECT_2_SERVER=8E18A60F508A3428346B23D5C
```

Example of return in XML:

```
<Response>
  <Code>0</Code>
  <Message>OK</Message>
  <Data>
    <Monitors>
      <Count>2</Count>
      <Monitor>
        <MonitorID>Monitor 1</MonitorID>
        <ClientID>AF209016277C36811CFDC95AD0D921A7</ClientID>
        <ClientAddress>192.168.0.2</ClientAddress>
        <ClientUser>admin</ClientUser>
        <CurrentObjectType>1</CurrentObjectType>
        <CurrentObjectName>Camera 1</CurrentObjectName>
        <CurrentObjectDescription>Entrance camera</CurrentObjectDescription>
        <CurrentScreenStyleData>1,1</CurrentScreenStyleData>
      </Monitor>
      <Monitor>
        <MonitorID>Monitor 2</MonitorID>
        <ClientID>ABF3928474CFE7E7FCABC89DBCA98378</ClientID>
        <ClientAddress>192.168.0.3</ClientAddress>
        <ClientUser>admin</ClientUser>
        <CurrentObjectType>16</CurrentObjectType>
        <CurrentObjectName>View1</CurrentObjectName>
        <CurrentObjectDescription>First floor</CurrentObjectDescription>
        <CurrentScreenStyleData>3,1,2,3,4,5,6,7,8,9</CurrentScreenStyleData>
        <CurrentScreenViewObjects>
          <Count>2</Count>
          <Object>
            <Name>03</Name>
            <Server>8E18A60F508A3428346B23D504B74484</Server>
            <Type>1</Type>
          </Object>
          <Object>
            <Name>06</Name>
            <Server>8E18A60F508A3428346B23D504B74484</Server>
            <Type>1</Type>
          </Object>
        </CurrentScreenViewObjects>
      </Monitor>
    </Monitors>
  </Data>
</Response>
```

Example of return in JSON:

```
{
  "Response": {
    "Code": 0,
    "Message": "OK",
    "Data": {
      "Monitors": [
        {
          "MonitorID": "Monitor 1",
          "ClientID": "AF209016277C36811CFDC95AD0D921A7",
          "ClientAddress": "192.168.0.2",
          "ClientUser": "admin",
          "CurrentObjectType": 1,
          "CurrentObjectName": "Camera 1",
          "CurrentObjectDescription": "Entrance camera",
          "CurrentScreenStyleData": "1,1"
        },
        {
          "MonitorID": "Monitor 2",
          "ClientID": "ABF3928474CFE7E7FCABC89DBCA98378",
          "ClientAddress": "192.168.0.3",
          "ClientUser": "admin",
          "CurrentObjectType": 16,
          "CurrentObjectName": "View1",
          "CurrentObjectDescription": "First floor",
          "CurrentScreenStyleData": "3,1,2,3,4,5,6,7,8,9",
          "CurrentScreenViewObjects": [
            {
              "Name": "03",
              "Server": "8E18A60F508A3428346B23D504B74484",
              "Type": 1
            },
            {
              "Name": "06",
              "Server": "8E18A60F508A3428346B23D504B74484",
              "Type": 1
            }
          ]
        }
      ]
    }
  }
}
```

4.12.2 Requesting the list of seen (Older) monitors

Requests the list of seen monitors in the virtual matrix of the server. This list delivers the registers of all of the monitors of the virtual matrix of the clients that connected to the server in the last 24 hours. This list does not include the records of currently active monitors.

Compatibility: Standard, Professional, Enterprise

Security level: Requires user authentication with rights to operate the virtual matrix

Method: HTTP GET

Syntax:

```
http://<server_address>/Interface/VirtualMatrix/GetSeenMonitors
[?<argument=value> [&<argument=value>...]] [&<general_argument>...]
```

Arguments:

Argument	Valid values	Description										
Monitors	APIMasks	Mask to filter the results. Specify which monitors must be returned based on the provided masks.										
Fields	MonitorID ClientID ClientAddress ClientUser	<p>Specifies the list of desired fields. If this parameter is omitted, all of the fields will be sent.</p> <p>The fields must be separated by commas</p> <table border="1" style="margin-left: auto; margin-right: auto;"> <thead> <tr> <th>Name</th><th>Description</th></tr> </thead> <tbody> <tr> <td>MonitorID</td><td>Identification of the monitor</td></tr> <tr> <td>ClientID</td><td>Identification of the client that owns the monitor</td></tr> <tr> <td>ClientAddress</td><td>IP Address of the client that owns the monitor</td></tr> <tr> <td>ClientUser</td><td>User of the client that owns the monitor</td></tr> </tbody> </table>	Name	Description	MonitorID	Identification of the monitor	ClientID	Identification of the client that owns the monitor	ClientAddress	IP Address of the client that owns the monitor	ClientUser	User of the client that owns the monitor
Name	Description											
MonitorID	Identification of the monitor											
ClientID	Identification of the client that owns the monitor											
ClientAddress	IP Address of the client that owns the monitor											
ClientUser	User of the client that owns the monitor											

Example 1: Request the list of seen monitors with all of the fields and response in XML

```
http://192.168.0.1:8601/Interface/VirtualMatrix/GetSeenMonitors?
ResponseFormat=XML
```

Example 2: Request the list of seen monitors with all of the fields and response in text

```
http://192.168.0.1:8601/Interface/VirtualMatrix/GetSeenMonitors?
ResponseFormat=Text
```

Example 3: Request the list of seen monitors with only ID of the monitor, response in XML and authentication of the user Admin

```
http://192.168.0.1:8601/Interface/VirtualMatrix/GetSeenMonitors?
Fields=MonitorID&ResponseFormat=XML&AuthUser=admin&AuthPass=pass
```

Example 4: Request the list of seen monitors starting with "A", with only monitor ID, response in XML and authentication with Admin user

```
http://192.168.0.1:8601/Interface/VirtualMatrix/GetSeenMonitors?
Monitors=A*&Fields=MonitorID&ResponseFormat=XML&AuthUser=admin&
AuthPass=pass
```

Response:

A list with all of the viewed monitors in the virtual matrix of the server is returned. The fields returned in the list will depend on the values informed in the argument **Fields**

HTTP Return: 200 OK

Parameters of return:**Fixed parameters:**

Parameter	Type	Description
COUNT	Integer	Total number of viewed monitors (Not included in JSON response)

Parameters of the list of viewed monitors:

Parameter	Type	Description
MONITORID	String	Identification of the monitor (Configured in the Surveillance Client)
CLIENTID	String	Identification of the client that owns the monitor
CLIENTADDRESS	String	IP Address of the client that owns the monitor
CLIENTUSER	String	User of the client that owns the monitor

Example of return in text:

```
RESPONSE_CODE=0
RESPONSE_MESSAGE=OK
COUNT=2
MONITOR_1_MONITORID=MONITOR_1
MONITOR_1_CLIENTID=AF209016277C36811CFDC95AD0D921A7
MONITOR_1_CLIENTADDRESS=192.168.0.2
MONITOR_1_CLIENTUSER=admin
MONITOR_2_MONITORID=MONITOR_2
MONITOR_2_CLIENTID=ABF3928474CFE7E7FCABC89DBCA98378
MONITOR_2_CLIENTADDRESS=192.168.0.3
MONITOR_2_CLIENTUSER=admin
```

Example of return in XML:

```
<Response>
  <Code>0</Code>
  <Message>OK</Message>
  <Data>
    <Monitors>
      <Count>2</Count>
      <Monitor>
        <MonitorID>Monitor_1</MonitorID>
        <ClientID>AF209016277C36811CFDC95AD0D921A7</ClientID>
        <ClientAddress>192.168.0.2</ClientAddress>
        <ClientUser>admin</ClientUser>
      </Monitor>
      <Monitor>
        <MonitorID>Monitor_2</MonitorID>
        <ClientID>ABF3928474CFE7E7FCABC89DBCA98378</ClientID>
        <ClientAddress>192.168.0.3</ClientAddress>
        <ClientUser>admin</ClientUser>
      </Monitor>
    </Monitors>
  </Data>
</Response>
```

Example of return in JSON:

```
{
  "Response": {
    "Code": 0,
    "Message": "OK",
    "Data": {
      "Monitors": [
        {
          "MonitorID": "Monitor 1",
          "ClientID": "AF209016277C36811CFDC95AD0D921A7",
          "ClientAddress": "192.168.0.2",
          "ClientUser": "admin"
        },
        {
          "MonitorID": "Monitor 2",
          "ClientID": "ABF3928474CFE7E7FCABC89DBCA98378",
          "ClientAddress": "192.168.0.3",
          "ClientUser": "admin"
        }
      ]
    }
  }
}
```

4.12.3 Loading an empty screen style in a monitor

By way of this command, you will be able to load an empty screen style (Layout) in any monitor of the virtual matrix.

Compatibility: Standard, Professional, Enterprise

Security level: Requires user authentication with rights to operate the virtual matrix

Method: HTTP GET

Syntax

```
http://<server_address>/Interface/VirtualMatrix/LoadScreenStyle?  
<argument=value>[&<argument=value>...][&<general argument>...]
```

Arguments:

Argument	Valid values	Description
MonitorID*	String	Identification of the monitor. Multiple monitors can be specified by using comma as separator.
ScreenStyleID*	Integer	ID of the screenstyle
DoNotReloadSameStyle	TRUE FALSE	If this flag is FALSE, the screen style will be reloaded regardless of current style. If this flag is TRUE and the current style is the same as the one being loaded, the system will not reload it If this parameter is omitted, the default value FALSE will be used.

* Mandatory parameters

Example 1: Display the screenstyle 6278 in monitor 1 of the virtual matrix with response in text

```
http://192.168.0.1:8601/Interface/VirtualMatrix/LoadScreenStyle?
MonitorID=Monitor 1&ScreenStyleID=6278&ResponseFormat=Text
```

Example 2: Display the screenstyle 9983 in monitor 1 of the virtual matrix with response in XML

```
http://192.168.0.1:8601/Interface/VirtualMatrix/LoadScreenStyle?
MonitorID=Monitor 1&ScreenStyleID=9983&ResponseFormat=XML
```

Response:

Default response of API.

HTTP Return: 200 OK

Parameter of return: [Default return of API](#)

4.12.4 Displaying an object in a monitor

By way of this command, you will be able to display an object in any monitor of the virtual matrix.

Tip: By way of subsequent calls of this command, you will be able to display the same object in several monitors of the virtual matrix simultaneously.

Compatibility: Standard, Professional, Enterprise

Security level: Requires user authentication with rights to operate the virtual matrix

Method: HTTP GET

Syntax:

```
http://<server_address>/Interface/VirtualMatrix>ShowObject?
<argument=value>[&<argument=value>...][&<general\_argument>...]
```

Arguments:

Argument	Valid values	Description																		
MonitorID*	String	Identification of the monitor. Multiple monitors can be specified by using comma as separator.																		
SpotNumber*	Integer. X=-1 Integer. X>= 1	Spot number <table border="1" data-bbox="840 1431 1379 1537"> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>-1</td><td>Object in fullscreen</td></tr> <tr> <td>>= 1</td><td>Spot number</td></tr> </tbody> </table>	Value	Description	-1	Object in fullscreen	>= 1	Spot number												
Value	Description																			
-1	Object in fullscreen																			
>= 1	Spot number																			
ObjectType*	Integer	Type of object <table border="1" data-bbox="840 1636 1379 1921"> <thead> <tr> <th>Type</th><th>Description</th></tr> </thead> <tbody> <tr> <td>0</td><td>Live camera</td></tr> <tr> <td>2</td><td>Map</td></tr> <tr> <td>3</td><td>Analytics Configuration</td></tr> <tr> <td>4</td><td>LPR Configuration</td></tr> <tr> <td>5</td><td>Web Pages</td></tr> <tr> <td>6</td><td>Operational Maps</td></tr> <tr> <td>7</td><td>LPR Zone</td></tr> <tr> <td>8</td><td>Group of LPR Zones</td></tr> </tbody> </table>	Type	Description	0	Live camera	2	Map	3	Analytics Configuration	4	LPR Configuration	5	Web Pages	6	Operational Maps	7	LPR Zone	8	Group of LPR Zones
Type	Description																			
0	Live camera																			
2	Map																			
3	Analytics Configuration																			
4	LPR Configuration																			
5	Web Pages																			
6	Operational Maps																			
7	LPR Zone																			
8	Group of LPR Zones																			

ObjectName**	String	Name of the object to be displayed
ObjectDUID**	String	DUID of the object to be displayed
DoNotReplaceSameObj	TRUE FALSE	When this flag is FALSE, the object being sent to the virtual matrix spot will be updated regardless if it is already on the screen. When this flag is TRUE and the object is already present on the spot, the system will not reload the object. If this parameter is omitted, the default value FALSE will be used.

* Mandatory parameters

** Mandatory mutually exclusive parameters (Use one or another)

Example 1: Displays a camera in monitor 1 of the virtual matrix with response in text

```
http://192.168.0.1:8601/Interface/VirtualMatrix>ShowObject?  
MonitorID=Monitor 1&SpotNumber=-1&ObjectType=0&ObjectName=Camera1&  
ResponseFormat=Text
```

Example 2: Displays a map in monitor 25 of the virtual matrix with response in XML

```
http://192.168.0.1:8601/Interface/VirtualMatrix>ShowObject?  
MonitorID=Monitor 25&SpotNumber=-1&ObjectType=2&ObjectName=Map1&  
ResponseFormat=XML
```

Example 3: Displays a map in spot 2 of monitor 25 of the virtual matrix with response in XML

```
http://192.168.0.1:8601/Interface/VirtualMatrix>ShowObject?  
MonitorID=Monitor 25&SpotNumber=2&ObjectType=2&ObjectName=Map1&  
ResponseFormat=XML
```

Response:

Default response of API.

HTTP Return: 200 OK

Parameter of return: [Default return of API](#)

4.12.5 Displaying an user screen view in a monitor

By way of this command, you will be able to display a user screen view in any monitor of the virtual matrix.

Tip: By way of subsequent calls of this command, you will be able to display the same screen view in several monitors of the virtual matrix simultaneously.

Compatibility: Standard, Professional, Enterprise

Security level: Requires user authentication with rights to operate the virtual matrix

Method: HTTP GET

Syntax

```
http://<server_address>/Interface/VirtualMatrix>ShowUserScreenView?
<argument=value>[&<argument=value>...][&<general argument>...]
```

Arguments:

Argument	Valid values	Description
MonitorID*	String	Identification of the monitor. Multiple monitors can be specified by using comma as separator.
ScreenStyleID*	Integer	ID of the screenstyle
ScreenViewName*	String	Name of the user view

* Mandatory parameters

Example 1: Display the view "View 1" of screenstyle 6278 in monitor 1 of the virtual matrix with response in text

```
http://192.168.0.1:8601/Interface/VirtualMatrix>ShowUserScreenView?
MonitorID=Monitor 1&ScreenStyleID=6278&ScreenViewName=View 1&
ResponseFormat=Text
```

Example 2: Display the view "View 2" of user guest1 of screenstyle 1 in monitor 1 of the virtual matrix with response in XML

```
http://192.168.0.1:8601/Interface/VirtualMatrix>ShowUserScreenView?
MonitorID=Monitor 1&ScreenStyleID=1&ScreenViewName=View 2&AuthUser=guest1&
AuthPass=guestpass&ResponseFormat=XML
```

Response:

Default response of API.

HTTP Return: 200 OK

Parameter of return: [Default return of API](#)

4.12.6 Displaying a public screen view in a monitor

By way of this command, you will be able to display a public screen view in any monitor of the virtual matrix.

Tip: By way of subsequent calls of this command, you will be able to display the same screen view in several monitors of the virtual matrix simultaneously.

Compatibility: Standard, Professional, Enterprise

Security level: Requires user authentication with rights to operate the virtual matrix

Method: HTTP GET

Syntax

```
http://<server_address>/Interface/VirtualMatrix>ShowPublicScreenView?
<argument=value>[&<argument=value>...][&<general argument>...]
```

Arguments:

Argument	Valid values	Description
MonitorID*	String	Identification of the monitor. Multiple monitors can be specified by using comma as separator.

ScreenStyleID*	Integer	ID of the screenstyle
ScreenViewName*	String	Name of the public screen view

* Mandatory parameters

Example 1: Display the view "View 1" of screenstyle 6278 in monitor 1 of the virtual matrix with response in text

```
http://192.168.0.1:8601/Interface/VirtualMatrix>ShowPublicScreenView?
MonitorID=Monitor 1&ScreenStyleID=6278&ScreenViewName=View 1&
ResponseFormat=Text
```

Example 2: Display the view "View 2" of screenstyle 1 in monitor 2 of the virtual matrix with response in XML

```
http://192.168.0.1:8601/Interface/VirtualMatrix>ShowPublicScreenView?
MonitorID=Monitor 2&ScreenStyleID=1&ScreenViewName=View 2&
ResponseFormat=XML
```

Response:

Default response of API.

HTTP Return: 200 OK

Parameter of return: [Default return of API](#)

4.12.7 Removing objects from a monitor

By way of this command, you will be able to remove objects from a monitor. You have the option of clearing the whole monitor or just a single spot.

Compatibility: Standard, Professional, Enterprise

Security level: Requires user authentication with rights to operate the virtual matrix

Method: HTTP GET

Syntax

```
http://<server_address>/Interface/VirtualMatrix/ClearMonitor?
<argument=value>[&<argument=value>...][&<general\_argument>...]
```

Arguments:

Argument	Valid values	Description
MonitorID*	String	Identification of the monitor. Multiple monitors can be specified by using comma as separator.
SpotNumber*	Integer	Number of the spot to clear (0 for whole screen). If this parameter is omitted, the default value of 0 will be used and all objects will be removed from the specified monitor.

* Mandatory parameters

Example 1: Clear all objects in monitor 1 of the virtual matrix with response in text

```
http://192.168.0.1:8601/Interface/VirtualMatrix/ClearMonitor?
MonitorID=Monitor 1&ResponseFormat=Text
```

Example 2: Remove object from spot number 1 in monitor 2 of the virtual matrix with response in XML

```
http://192.168.0.1:8601/Interface/VirtualMatrix/ClearMonitor?
MonitorID=Monitor 2&SpotNumber=1&ResponseFormat=XML
```

Response:

Default response of API.

HTTP Return: 200 OK

Parameter of return: [Default return of API](#)

4.12.8 Starting media playback in a monitor

By using this command you can start a playback session of any camera in any monitor of virtual matrix.

Tip: By way of subsequent calls of this command, you will be able to display the same object in several monitors of the virtual matrix simultaneously.

Compatibility: Standard, Professional, Enterprise

Security level: Requires user authentication with rights to operate the virtual matrix

Method: HTTP GET

Syntax:

```
http://<server_address>/Interface/VirtualMatrix/StartMediaPlayback?
<argument=value>[&<argument=value>... ] [&<general\_argument>... ]
```

Arguments:

Argument	Valid values	Description
MonitorID*	String	Monitor identification
Cameras*	String	Name of cameras for media playback (List must be comma-separated)
SecondsAgo*	Integer	Open media playback of X seconds ago Specify -1 or omit this value for using <code>StartDate</code> , <code>StartTime</code> , <code>EndDate</code> and <code>EndTime</code> parameters instead.
StartDate	APIDate	Start date of media session
StartTime	APITime	Start time of media session
EndDate	APIDate	End date of media session. If this parameter is omitted, the current date will be used
EndTime	APITime	End time of media session. If this parameter is omitted, the current time will be used

* Mandatory parameters

Example 1: Start media playback of cameras "Camera1" and "Camera2" in monitor "Monitor1" to view recordings of last 5 minutes

```
http://192.168.0.1:8601/Interface/VirtualMatrix/StartMediaPlayback?
MonitorID=Monitor1&Cameras=Camera1, Camera2&SecondsAgo=300
```

Example 2: Start media playback of camera "Camera1" in monitor "Monitor1" to view recordings from March 07, 2014 17:15:00.000 to March 08, 2014 17:00:00.000

```
http://192.168.0.1:8601/Interface/VirtualMatrix/StartMediaPlayback?
MonitorID=Monitor1&Cameras=Camera1&SecondsAgo=-1&StartDate=2014.03.07&
StartTime=17.15.00.000&EndDate=2014.03.08&EndTime=17.00.00.000
```

Example 3: Start media playback of camera "Camera1" in monitor "Monitor1" to view recordings from March 07, 2014 17:15:00.000 to date.

```
http://192.168.0.1:8601/Interface/VirtualMatrix/StartMediaPlayback?
MonitorID=Monitor1&Cameras=Camera1&SecondsAgo=-1&StartDate=2014.03.07&
StartTime=17.15.00.000
```

Response:

Default response of API.

HTTP Return: 200 OK

Parameter of return: [Default return of API](#)

4.13 Events

4.13.1 Searching for events records

Perform a search for event records in the database

Compatibility: All editions

Security level: Requires user authentication with rights to search for event records

Method: HTTP GET

Syntax:

```
http://<server_address>/Interface/Events/Search
[?<argument=value> [&<argument=value>...]] [&<general_argument>...]]
```

Arguments:

Argument	Valid values	Description				
StartDate	APIDate	Search for records starting with the specified date				
StartTime	APITime	Search for records starting with the specified time				
EndDate	APIDate	End date of search. Search for all records inside the range of start and end. If this parameter is omitted, the value specified in <code>StartDate</code> parameter will be used				
EndTime	APITime	End time of search. Search for all records inside the range of start and end. If this parameter is omitted, the value specified in <code>StartTime</code> parameter will be used				
EventTypes	String	List of types of events to search. The list must be comma-separated. <div style="margin-left: 20px;"> <table border="1"> <thead> <tr> <th>Event Types</th> </tr> </thead> <tbody> <tr><td>AlarmInput</td></tr> <tr><td>DeviceCommunication</td></tr> <tr><td>RecordingError</td></tr> </tbody> </table> </div>	Event Types	AlarmInput	DeviceCommunication	RecordingError
Event Types						
AlarmInput						
DeviceCommunication						
RecordingError						

MotionDetection
AudioLevelDetection
ManualEvent
ScheduledEvent
GlobalEvent
Analytics
LPR
LPRZone
ServerFailover
DeviceEvent
ServerEvent

Example 1: Search for all records between January 01, 2014 and February 01, 2014

```
http://192.168.0.1:8601/Interface/Events/Search?StartDate=2014.01.01&
StartTime=00.00.00.000&EndDate=2014.02.01&EndTime=23.59.59.999
```

Example 2: Search for all records of AlarmInput and LPR

```
http://192.168.0.1:8601/Interface/Events/Search?Events=AlarmInput,LPR
```

Response:

A list with all found event records is returned.

HTTP Return: 200 OK

Parameters of return:

Fixed Parameter:

Parameter	Type	Description
COUNT	Integer	Total of records from all event types

Parameters of records list:

Parameter	Type	Description
ALARMINPUTCOUNT	Integer	Number of records of alarm input events
DEVICECOMMUNICATIONCOUNT	Integer	Number of records of device communication events
RECORDINGERRORCOUNT	Integer	Number of records of recording error events
MOTIONDETECTIONCOUNT	Integer	Number of records of motion detection events
AUDIOLEVELDETECTIONCOUNT	Integer	Number of records of audio level detection events
DEVICEEVENTCOUNT	Integer	Number of records of device events
MANUALEVENTCOUNT	Integer	Number of records of manual events
SCHEDULEDVENTCOUNT	Integer	Number of records of scheduled events
GLOBALEVENTCOUNT	Integer	Number of records of global events
ANALYTICSCOUNT	Integer	Number of records of analytics events
LPRCOUNT	Integer	Number of records of LPR events
SERVERFAILOVERCOUNT	Integer	Number of records of server failover events
SERVEVENTCOUNT	Integer	Number of records of server events
RECORDNUMBER	Integer	Number of records on database
DATETIME	APITimestamp	Date and time of the event
EVENTNAME*	String	Name of the event

DEVICENAME*	String	Name of the device
ANALYTICSTYPE*	String	Type of analytics
ZONE*	String	Analytics zone name
LICENSEPLATE*	String	License plate string
USERNAME*	String	Name of user
IP*	String	Network address
AUDIOLEVEL*	String	Type of audio level
SERVERTAILOVER*	String	Type of failover event
DEVICECOMMUNICATIONEVENT*	String	Type of device communication event
DEVICECOMMUNICATIONFAILURETIME*	String	Total failure time of device
RECORDINGEVENT*	String	Type of recording event
SERVEVENT*	String	Type of server event
RECORDINGFAILURETIME*	String	Total recording failure time
SERVEVENTTIME*	String	Total server event time
DEVICEEVENT*	String	Type of device event
LPRZONEEVENTTYPE*	String	Type of LPR Zone event
LPRZONEDWELLTIME*	String	Total dwell time of an LPR Zone
LPRZONEENTRANCETIMES*	String	LPR Zone Entrance Times

* These parameters are event-dependent and may only be available in certain types of events

OBS: Count fields are not included in JSON response format

List of records:

The parameters of the list of event records will depend on the type of response (Text or XML).

List of event records with response in text:

The parameters of response in text will obey the following syntax:

<eventtype>RECORD_<num>_<field>=<value>

Parameter	Description															
eventtype	Type of event <table border="1" style="margin-left: 20px;"> <tr> <th>Event Types</th></tr> <tr><td>ALARMINPUT</td></tr> <tr><td>DEVICECOMMUNICATION</td></tr> <tr><td>RECORDINGERROR</td></tr> <tr><td>MOTIONDETECTION</td></tr> <tr><td>AUDIOLEVELDETECTION</td></tr> <tr><td>DEVICEEVENT</td></tr> <tr><td>MANUALEVENT</td></tr> <tr><td>SCHEDULEDEVENT</td></tr> <tr><td>GLOBALEVENT</td></tr> <tr><td>ANALYTICS</td></tr> <tr><td>LPR</td></tr> <tr><td>LPRZONE</td></tr> <tr><td>SERVERTAILOVER</td></tr> <tr><td>SERVEVENT</td></tr> </table>	Event Types	ALARMINPUT	DEVICECOMMUNICATION	RECORDINGERROR	MOTIONDETECTION	AUDIOLEVELDETECTION	DEVICEEVENT	MANUALEVENT	SCHEDULEDEVENT	GLOBALEVENT	ANALYTICS	LPR	LPRZONE	SERVERTAILOVER	SERVEVENT
Event Types																
ALARMINPUT																
DEVICECOMMUNICATION																
RECORDINGERROR																
MOTIONDETECTION																
AUDIOLEVELDETECTION																
DEVICEEVENT																
MANUALEVENT																
SCHEDULEDEVENT																
GLOBALEVENT																
ANALYTICS																
LPR																
LPRZONE																
SERVERTAILOVER																
SERVEVENT																
num	Number of the record															
field	Name of the field															
value	Value of the field															

Example of return in text:

```

RESPONSE_CODE=0
RESPONSE_MESSAGE=OK
COUNT=2
DEVICECOMMUNICATIONCOUNT=1
ANALYTICSCOUNT=1
DEVICECOMMUNICATIONRECORD_1_RECORDNUMBER=7324163
DEVICECOMMUNICATIONRECORD_1_DATETIME=2015-09-27 17:00:17.308
DEVICECOMMUNICATIONRECORD_1_DEVICENAME=Internal I/O Device
DEVICECOMMUNICATIONRECORD_1_DEVICECOMMUNICATIONEVENT=Communication failure
ANALYTICSRECORD_1_RECORDNUMBER=7324209
ANALYTICSRECORD_1_DATETIME=2015-09-27 17:09:11.295
ANALYTICSRECORD_1_DEVICENAME=04
ANALYTICSRECORD_1_ANALYTICSTYPE=Face detection

```

List of events records with response in XML:

The parameters of response in XML will obey the following syntax:

```

<EventLogRecords>
    <Count>COUNT</Count>
    <ServerFailoverCount>FAILOVER_EVENT_COUNT</ServerFailoverCount>
    <LPRCount>LPR_EVENT_COUNT</LPRCount>
    <AnalyticsCount>ANALYTICS_EVENT_COUNT</AnalyticsCount>
    <GlobalEventCount>GLOBAL_EVENT_COUNT</GlobalEventCount>
    <ScheduledEventCount>SCHEDULED_EVENT_COUNT</ScheduledEventCount>
    <ManualEventCount>MANUAL_EVENT_COUNT</ManualEventCount>
    <AudioLevelDetectionCount>AUDIO_LEVEL_EVENT_COUNT</AudioLevelDetectionCount>
    <MotionDetectionCount>MOTION_DETECTION_EVENT_COUNT</MotionDetectionCount>
    <RecordingErrorCount>RECORDING_ERROR_EVENT_COUNT</RecordingErrorCount>
    <DeviceCommunicationCount>DEVICE_COMM_EVENT_COUNT</DeviceCommunicationCount>
    <AlarmInputCount>ALARM_INPUT_EVENT_COUNT</AlarmInputCount>

    <!--
        Each event type will have a subsection containing its events
        list and properties, below is an example of LPR events
    -->
    <LPRRecords>
        <LPRRecord>
            <RecordNumber>RECORD_NUMBER</RecordNumber>
            <DateTime>DATE_TIME</DateTime>
            <DeviceName>DEVICE_NAME</DeviceName>
            <LicensePlate>LICENSE_PLATE</LicensePlate>
        </LPRRecord>
    </LPRRecords>

</EventLogRecords>

```

Example of return in XML:

```

<Response>
    <Code>0</Code>
    <Message>OK</Message>
    <Data>
        <EventLogRecords>
            <Count>2</Count>

```

```

<DeviceCommunicationCount>1</DeviceCommunicationCount>
<AnalyticsCount>1</AnalyticsCount>
<DeviceCommunicationRecords>
    <DeviceCommunicationRecord>
        <RecordNumber>7324163</RecordNumber>
        <DateTime>2015-09-27 17:00:17.308</DateTime>
        <DeviceName>Internal I/O Device</DeviceName>
        <DeviceCommunicationEvent>Communication failure</DeviceCommunicationEvent>
    </DeviceCommunicationRecord>
</DeviceCommunicationRecords>
<AnalyticsRecords>
    <AnalyticsRecord>
        <RecordNumber>7324209</RecordNumber>
        <DateTime>2015-09-27 17:09:11.295</DateTime>
        <DeviceName>04</DeviceName>
        <AnalyticsType>Face detection</AnalyticsType>
    </AnalyticsRecord>
</AnalyticsRecords>
</EventLogRecords>
</Data>
</Response>

```

Example of return in JSON (JSON response does not include count fields):

```
{
    "Response": {
        "Code": 0,
        "Message": "OK",
        "Data": {
            "EventLogRecords": [
                "DeviceCommunicationRecords": [
                    {
                        "RecordNumber": 529578,
                        "DateTime": "2020-06-11T00:00:00.536Z",
                        "DeviceName": "LPR",
                        "DeviceCommunicationEvent": "Communication failure",
                        "DeviceCommunicationFailureTime": "317:37:18"
                    }
                ],
                "ScheduledEventRecords": [
                    {
                        "RecordNumber": 529579,
                        "DateTime": "2020-06-11T00:00:00.747Z",
                        "EventName": "Dome Preset"
                    }
                ]
            }
        }
    }
}
```

4.13.2 Monitoring server events

Start the server events monitoring.

Compatibility: All editions

Security level: Requires user authentication with rights to view the event log

Method: HTTP GET

Syntax:

```
http://<server_address>/Interface/Events/Monitor  
[?<argument=value>[&<argument=value>...]] [ &<general\_argument>... ]
```

Arguments:

Argument	Valid values	Description
KeepAliveInterval	Integer. 1 >= X <= 120	Specify the time (in seconds) of Keep-Alive packet sending. The Keep-Alive packet is used to control the connection. If this parameter is omitted, the default value of 5 will be used

Example 1: Start the server events monitoring with response in XML

```
http://192.168.0.1:8601/Interface/Events/Monitor?ResponseFormat=XML
```

Example 2: Start the server events monitoring with response in text and keep-alive interval of 60 seconds

```
http://192.168.0.1:8601/Interface/Events/Monitor?ResponseFormat=Text&  
KeepAliveInterval=60
```

Example 3: Start the server events monitoring with response in XML and authentication of the user Admin

```
http://192.168.0.1:8601/Interface/Events/Monitor?  
ResponseFormat=XML&AuthUser=admin&AuthPass=pass
```

Response:

A stream of events data will be sent using HTTP Multipart x-mixed-replace transmission. The response data stream is continuous, so an event of Keep-Alive will be sent every X seconds (Configurable by KeepAliveInterval parameter), with this event you can verify if the TCP connection is still open. Each event is separated by the multipart boundary --DigifortBoundary.

HTTP Return: 200 OK

Parameter of return:

The return parameters are divided into two subcategories: ObjectData and EventData.

ObjectData parameters:

Parameter	Type	Description
DUID	DUID	DUID of the object that triggered the event
NAME	String	Name of the object that triggered the event
TYPE	String	System Object ID

EventData parameters:

Parameter	Type	Description
NAME	String	Event name. Only applicable to events: ALARM_INPUT, MANUAL, TIMER.

Parameter	Type	Description
TYPE	String	Event type
DATA	String	Internal event data (Not to be used by third party)
METADATA	String	Event variables containing extra information on the event. Format is a list of Parameter/Value using the selected response format (Text, XML, JSON) encoded in Base64. The parameters will vary according to the type of event. For a list of all possible variables and values please consult the Event Variables document.
TIMESTAMP	APITimestamp	Date and time of the event
UTCTIMESTAMP	APITimestamp	UTC date and time of the event

Example of return in text:

```
--DigifortBoundary
Content-Type: text/plain; charset=UTF-8
Content-Length: 217

RESPONSE_CODE=0
RESPONSE_MESSAGE=OK
OBJECTDATA_NAME=Event1
OBJECTDATA_TYPE=GLOBAL_EVENT
EVENTDATA_NAME=
EVENTDATA_TYPE=GLOBAL
EVENTDATA_DATA=
EVENTDATA_METADATA=
EVENTDATA_TIMESTAMP=2009-11-15 10:55:30.347
EVENTDATA_UTCTIMESTAMP=2009-11-15 13:55:30.347
--DigifortBoundary
Content-Type: text/plain; charset=UTF-8
Content-Length: 203

RESPONSE_CODE=0
RESPONSE_MESSAGE=OK
OBJECTDATA_NAME=
OBJECTDATA_TYPE=
EVENTDATA_NAME=
EVENTDATA_TYPE=KEEP_ALIVE
EVENTDATA_DATA=
EVENTDATA_METADATA=
EVENTDATA_TIMESTAMP=2009-11-15 10:55:32.550
EVENTDATA_UTCTIMESTAMP=2009-11-15 13:55:32.550
--DigifortBoundary
..
..
```

Example of return in XML:

```
--DigifortBoundary
Content-Type: text/xml; charset=UTF-8
Content-Length: 323

<?xml version="1.0" encoding="UTF-8" ?>
<Response>
  <Code>0</Code>
  <Message>OK</Message>
  <Data>
    <Event>
      <EventData>
        <ObjectData>
          <Name>Event1</Name>
          <Type>GLOBAL_EVENT</Type>
        </ObjectData>
        <EventData>
          <Name></Name>
          <Type>GLOBAL</Type>
          <Data></Data>
          <Metadata></Metadata>
          <Timestamp>2009-11-15 10:55:30.347</Timestamp>
          <UTCTimestamp>2009-11-15 13:55:30.347</UTCTimestamp>
        </EventData>
      </Event>
    </Data>
  </Response>
--DigifortBoundary
Content-Type: text/xml; charset=UTF-8
Content-Length: 337

<?xml version="1.0" encoding="UTF-8" ?>
<Response>
  <Code>0</Code>
  <Message>OK</Message>
  <Data>
    <Event>
      <EventData>
        <ObjectData>
          <Name></Name>
          <Type></Type>
        </ObjectData>
        <EventData>
          <Name></Name>
          <Type>KEEP_ALIVE</Type>
          <Data></Data>
          <Metadata></Metadata>
          <Timestamp>2009-11-15 10:55:32.550</Timestamp>
          <UTCTimestamp>2009-11-15 13:55:32.550</UTCTimestamp>
        </EventData>
      </Event>
    </Data>
  </Response>
--DigifortBoundary
..
..
```

Example of return in JSON:

--DigifortBoundary
Content-Type: text/xml; charset=UTF-8
Content-Length: 323

```
{
  "Code": 0,
  "Message": "OK",
  "Data": {
    "Event": {
      "EventData": {
        "Name": "Event1",
        "Type": "GLOBAL_EVENT"
      },
      "Event": {
        "Name": "",
        "Type": "GLOBAL",
        "Data": "",
        "Metadata": "",
        "Timestamp": "2009-11-15T10:55:30.347Z",
        "UTCTimestamp": "2009-11-15T13:55:30.347Z"
      }
    }
  }
}
```

--DigifortBoundary
Content-Type: text/xml; charset=UTF-8
Content-Length: 337

```
{
  "Code": 0,
  "Message": "OK",
  "Data": {
    "Event": {
      "EventData": {
        "Name": "",
        "Type": ""
      }
    },
    "Event": {
      "Name": "",
      "Type": "KEEP_ALIVE",
      "Data": "",
      "Metadata": "",
      "Timestamp": "2009-11-15T10:55:32.550Z",
      "UTCTimestamp": "2009-11-15T13:55:32.550Z"
    }
  }
}
```

--DigifortBoundary

..

4.14 LPR

4.14.1 Requesting the list of LPR Configurations

Request the list of LPR Configurations over which the user has rights.

Compatibility: Standard, Professional, Enterprise

Security level: Requires user authentication

Method: HTTP GET

Syntax:

```
http://<server_address>/Interface/LPR/GetLPRConfigurations[?<argument=value>
[&<argument=value>...][&<general_argument>...]]
```

Arguments:

Argument	Valid values	Description										
LPRConfigurations	APIMasks	Mask to filter the results. Specify which LPR Configurations must be returned based on the provided masks.										
Fields	Name Description Active Camera	<p>Specifies the list of desired fields. In case this parameter is omitted, all of the fields will be sent.</p> <p>The fields must be separated by commas</p> <table border="1"> <thead> <tr> <th>Name</th><th>Description</th></tr> </thead> <tbody> <tr> <td>Name</td><td>Name of the LPR Configuration</td></tr> <tr> <td>Description</td><td>Description of the LPR Configuration</td></tr> <tr> <td>Active</td><td>LPR Configuration activated or deactivated</td></tr> <tr> <td>Camera</td><td>Associated camera with the LPR Configuration</td></tr> </tbody> </table>	Name	Description	Name	Name of the LPR Configuration	Description	Description of the LPR Configuration	Active	LPR Configuration activated or deactivated	Camera	Associated camera with the LPR Configuration
Name	Description											
Name	Name of the LPR Configuration											
Description	Description of the LPR Configuration											
Active	LPR Configuration activated or deactivated											
Camera	Associated camera with the LPR Configuration											

Example 1: Request the list of LPR Configurations with all fields and response in XML

```
http://192.168.0.1:8601/Interface/LPR/GetLPRConfigurations?
ResponseFormat=XML
```

Example 2: Request the list of LPR Configurations with all fields and response in text

```
http://192.168.0.1:8601/Interface/LPR/GetLPRConfigurations?
ResponseFormat=Text
```

Example 3: Request the list of LPR Configurations with only name and description, response in XML and authentication of the user Admin

```
http://192.168.0.1:8601/Interface/LPR/GetLPRConfigurations?Fields=Name,
Description&ResponseFormat=XML&AuthUser=admin
```

Example 4: Request the list of LPR Configurations starting with "A", with only name and description, response in XML and authentication with Admin user

```
http://192.168.0.1:8601/Interface/LPR/GetLPRConfigurations?
LPRConfigurations=A*&Fields=Name,Description&ResponseFormat=XML&
AuthUser=admin
```

Response:

A list of all of the LPR Configurations registered in the system is returned. The fields returned in the will depend on the values informed in the argument Fields

HTTP Return: 200 OK

Parameters of return:**Fixed Parameter:**

Parameter	Type	Description
COUNT	Integer	Total number of LPR Configurations (Not included in JSON response)

Parameters in the list of LPR Configurations:

Parameter	Type	Description
NAME	String	Name of the LPR Configuration
DESCRIPTION	String	Description of the LPR Configuration
ACTIVE	Boolean	LPR Configuration is activated or deactivated
CAMERA	String	Name of the camera associated to the LPR Configuration

Example of return in text:

```
RESPONSE_CODE=0
RESPONSE_MESSAGE=OK
COUNT=2
LPRCONFIGURATION_1_NAME=Configuration1
LPRCONFIGURATION_1_DESCRIPTION=My Configuration
LPRCONFIGURATION_1_ACTIVE=TRUE
LPRCONFIGURATION_1_CAMERA=Camera1
LPRCONFIGURATION_2_NAME=Configuration2
LPRCONFIGURATION_2_DESCRIPTION=My Configuration
LPRCONFIGURATION_2_CAMERA=Camera2
LPRCONFIGURATION_2_ACTIVE=FALSE
```

Example of return in XML:

```
<Response>
  <Code>0</Code>
  <Message>OK</Message>
  <Data>
    <LPRConfigurations>
      <Count>2</Count>
      <LPRConfiguration>
        <Name>Configuration1</Name>
        <Description>My Configuration</Description>
        <Active>TRUE</Active>
        <Camera>Camera1</Camera>
      </LPRConfiguration>
      <LPRConfiguration>
        <Name>Configuration2</Name>
        <Description>My Configuration</Description>
        <Active>FALSE</Active>
        <Camera>Camera2</Camera>
      </LPRConfiguration>
    </LPRConfigurations>
  </Data>
</Response>
```

```

</LPRConfigurations>
</Data>
</Response>
```

Example of return in JSON:

```
{
  "Response": {
    "Code": 0,
    "Message": "OK",
    "Data": {
      "LPRConfigurations": [
        {
          "Name": "Configuration1",
          "Description": "My Configuration",
          "Active": true,
          "Camera": "Camera1"
        },
        {
          "Name": "Configuration1",
          "Description": "My Configuration",
          "Active": false,
          "Camera": "Camera2"
        }
      ]
    }
  }
}
```

4.14.2 Requesting the status of LPR Configurations

Request the status of the LPR Configurations over which the user has rights.

Compatibility: Standard, Professional, Enterprise

Security level: Requires user authentication

Method: HTTP GET

Syntax:

```
http://<server_address>/Interface/LPR/GetStatus [?<argument=value>
[&<argument=value>...]] [&<general_argument>...]]
```

Arguments:

Argument	Valid values	Description
LPRConfigurations	APIMasks	Mask to filter the results. Specify which configurations must be returned based on the provided masks.
Fields	Active Working Status StatusMessage	Specifies the list of desired fields. In case this parameter is omitted, all of the fields will be sent. The fields must be separated by commas
Name		Description
Active		Identify if the configuration is active

		<table border="1"> <tr><td>Working</td><td>Identify if the configuration is working</td></tr> <tr><td>Status</td><td>Current status of the configuration</td></tr> <tr><td>StatusMessage</td><td>Current status message of the configuration</td></tr> </table>	Working	Identify if the configuration is working	Status	Current status of the configuration	StatusMessage	Current status message of the configuration
Working	Identify if the configuration is working							
Status	Current status of the configuration							
StatusMessage	Current status message of the configuration							
Active	TRUE FALSE	In case this parameter is specified, a filter will be applied and only the configurations that matches this filter will be returned, thus, in case the value Active=TRUE is specified, the result will only include the active configurations, while if the value Active=FALSE is specified, the result will only include the deactivated configurations.						
Working	TRUE FALSE	In case this parameter is specified, a filter will be applied and only the configurations that matches this filter will be returned, thus, in case the value Working=TRUE is specified, the result will only include the working configurations, while if the value Working=FALSE is specified, the result will only include the configurations that are not working.						

Example 1: Request the status of all configurations with all fields and response in XML

```
http://192.168.0.1:8601/Interface/LPR/GetStatus?ResponseFormat=XML
```

Example 2: Request the status of all active configurations with response in text

```
http://192.168.0.1:8601/Interface/LPR/GetStatus?Active=TRUE&ResponseFormat=Text
```

Example 3: Request the status of all active configurations starting with A, with response in text

```
http://192.168.0.1:8601/Interface/LPR/GetStatus?LPRConfigurations=A*&Active=TRUE&ResponseFormat=Text
```

Example 4: Request the status of all active configurations that are not working, with response in XML and authentication with admin user (No password)

```
http://192.168.0.1:8601/Interface/LPR/GetStatus?Active=TRUE&Working=FALSE&ResponseFormat=XML&AuthUser=admin
```

Response:

A list with the status of all of the configurations over which the user has rights is returned. The fields returned in the will depend on the values informed in the argument Fields

HTTP Return: 200 OK

Parameters of return:

Fixed Parameter:

Parameter	Type	Description
COUNT	Integer	Total number of configurations (Not included in JSON response)

Parameters in the list of status of configurations:

Parameter	Type	Description
NAME	String	Name of the configuration
ACTIVE	Boolean	Identify if the configuration is active
WORKING	Boolean	Identify if the configuration is working

Parameter	Type	Description										
STATUS	String	<p>The following table lists the possible values for status:</p> <table border="1"> <thead> <tr> <th>Values</th></tr> </thead> <tbody> <tr><td>CAMERA_NOT_FOUND</td></tr> <tr><td>CAMERA_DEACTIVATED</td></tr> <tr><td>CAMERA_NOT_WORKING</td></tr> <tr><td>INVALID_MEDIA_PROFILE</td></tr> <tr><td>INVALID_ENGINE_CONFIGURATION_DATA</td></tr> <tr><td>INVALID_LPR_DRIVER</td></tr> <tr><td>VIDEO_NODE_LOST</td></tr> <tr><td>DISABLED_BY_SCHEDULE</td></tr> <tr><td>DRIVER</td></tr> </tbody> </table>	Values	CAMERA_NOT_FOUND	CAMERA_DEACTIVATED	CAMERA_NOT_WORKING	INVALID_MEDIA_PROFILE	INVALID_ENGINE_CONFIGURATION_DATA	INVALID_LPR_DRIVER	VIDEO_NODE_LOST	DISABLED_BY_SCHEDULE	DRIVER
Values												
CAMERA_NOT_FOUND												
CAMERA_DEACTIVATED												
CAMERA_NOT_WORKING												
INVALID_MEDIA_PROFILE												
INVALID_ENGINE_CONFIGURATION_DATA												
INVALID_LPR_DRIVER												
VIDEO_NODE_LOST												
DISABLED_BY_SCHEDULE												
DRIVER												
STATUSMESSAGE	String	This parameter will contain a human readable message describing the current status of the configuration										

Example of return in text:

```

RESPONSE_CODE=0
RESPONSE_MESSAGE=OK
COUNT=2
LPRCONFIGURATION_1_NAME=Entrance
LPRCONFIGURATION_1_ACTIVE=TRUE
LPRCONFIGURATION_1_WORKING=TRUE
LPRCONFIGURATION_1_STATUS=DRIVER
LPRCONFIGURATION_1_STATUSMESSAGE=Data: Processing...
LPRCONFIGURATION_2_NAME=Exit
LPRCONFIGURATION_2_ACTIVE=TRUE
LPRCONFIGURATION_2_WORKING=TRUE
LPRCONFIGURATION_2_STATUS=DRIVER
LPRCONFIGURATION_2_STATUSMESSAGE=Data: Processing...

```

Example of return in XML:

```

<Response>
  <Code>0</Code>
  <Message>OK</Message>
  <Data>
    <LPRConfigurations>
      <Count>2</Count>
      <LPRConfiguration>
        <Name>Entrance</Name>
        <Active>TRUE</Active>
        <Working>TRUE</Working>
        <Status>DRIVER</Status>
        <StatusMessage>Data: Processing...</StatusMessage>
      </LPRConfiguration>
      <LPRConfiguration>
        <Name>Exit</Name>
        <Active>TRUE</Active>
        <Working>TRUE</Working>
        <Status>DRIVER</Status>
        <StatusMessage>Data: Processing...</StatusMessage>
      </LPRConfiguration>
    </LPRConfigurations>
  </Data>
</Response>

```

Example of return in JSON:

```
{
  "Response": {
    "Code": 0,
    "Message": "OK",
    "Data": {
      "LPRConfigurations": [
        {
          "Name": "Entrance",
          "Active": true,
          "Working": true,
          "Status": "DRIVER",
          "StatusMessage": "Data: Processing..."
        },
        {
          "Name": "Exit",
          "Active": true,
          "Working": true,
          "Status": "DRIVER",
          "StatusMessage": "Data: Processing..."
        }
      ]
    }
  }
}
```

4.14.3 Requesting the list of surrounding cameras of an LPR Configuration

Request the list of surrounding cameras of an LPR Configuration.

Compatibility: Standard, Professional, Enterprise

Security level: Requires user authentication

Method: HTTP GET

Syntax:

```
http://<server_address>/Interface/LPR/GetSurroundingCameras?
<argument=value>[&<argument=value>...][&<general\_argument>...]
```

Arguments:

Argument	Valid values	Description
LPRConfiguration*	String	LPR Configuration name
SurroundingCameras	API Masks	Mask to filter the results. Specify which surrounding cameras must be returned based on the provided masks.

* Mandatory parameters

Example 1: Request the list of surrounding cameras of LPR Configuration "LPR1", with response in XML

```
http://192.168.0.1:8601/Interface/LPR/GetSurroundingCameras?
LPRConfiguration=LPR1&ResponseFormat=XML
```

Example 2: Request the list of surrounding cameras starting with "A" of LPR Configuration "LPR1", with response in text

```
http://192.168.0.1:8601/Interface/LPR/GetSurroundingCameras?
LPRConfiguration=LPR1&SurroundingCameras=A*&ResponseFormat=Text
```

Response:

A list with all surrounding cameras of the specified LPR Configuration is returned

HTTP Return: 200 OK

Parameters of return:

Fixed Parameter:

Parameter	Type	Description
COUNT	Integer	Total of surrounding cameras (Not included in JSON response)

Parameters of surrounding cameras list:

Parameter	Type	Description
NAME	String	Name of surrounding camera

Example of return in text:

```
RESPONSE_CODE=0
RESPONSE_MESSAGE=OK
COUNT=4
SURROUNDINGCAMERA_1_NAME=Camera1
SURROUNDINGCAMERA_2_NAME=Camera2
SURROUNDINGCAMERA_3_NAME=Camera3
SURROUNDINGCAMERA_4_NAME=Camera4
```

Example of return in XML:

```

<Response>
  <Code>0</Code>
  <Message>OK</Message>
  <Data>
    <SurroundingCameras>
      <Count>4</Count>
      <SurroundingCamera>
        <Name>Camera1</Name>
      </SurroundingCamera>
      <SurroundingCamera>
        <Name>Camera2</Name>
      </SurroundingCamera>
      <SurroundingCamera>
        <Name>Camera3</Name>
      </SurroundingCamera>
      <SurroundingCamera>
        <Name>Camera4</Name>
      </SurroundingCamera>
    </SurroundingCameras>
  </Data>
</Response>

```

Example of return in JSON:

```
{
  "Response": {
    "Code": 0,
    "Message": "OK",
    "Data": {
      "SurroundingCameras": [
        {
          "Name": "Camera1"
        },
        {
          "Name": "Camera2"
        },
        {
          "Name": "Camera3"
        },
        {
          "Name": "Camera4"
        }
      ]
    }
  }
}
```

4.14.4 Activating / Deactivating LPR configurations

Allows the activation or deactivation of multiple LPR Configurations simultaneously

Compatibility: Standard, Professional, Enterprise

Security level: Requires user authentication with rights to LPR Configuration Registration

Method: HTTP GET

Syntax:

```
http://<server_address>/Interface/LPR/Activation?<argument=value>
[&<argument=value>...] [&<general argument>...]
```

Arguments:

Argument	Valid values	Description										
LPRConfigurations*	String	<p>List of LPR configurations to activate or deactivate.</p> <p>The configuration names must be separated by commas.</p> <p>This command is only mandatory when Action=Activate or Action=Deactivate</p>										
Action*	Activate Deactivate ActivateAll DeactivateAll	<p>Action to be executed</p> <table border="1"> <thead> <tr> <th>Operation</th><th>Description</th></tr> </thead> <tbody> <tr> <td>Activate</td><td>Activate the configurations on LPRConfigurations argument</td></tr> <tr> <td>Deactivate</td><td>Deactivate the configurations on LPRConfigurations argument</td></tr> <tr> <td>ActivateAll</td><td>Activate all LPR Configurations from the server</td></tr> <tr> <td>DeactivateAll</td><td>Deactivate all LPR Configurations from the server</td></tr> </tbody> </table>	Operation	Description	Activate	Activate the configurations on LPRConfigurations argument	Deactivate	Deactivate the configurations on LPRConfigurations argument	ActivateAll	Activate all LPR Configurations from the server	DeactivateAll	Deactivate all LPR Configurations from the server
Operation	Description											
Activate	Activate the configurations on LPRConfigurations argument											
Deactivate	Deactivate the configurations on LPRConfigurations argument											
ActivateAll	Activate all LPR Configurations from the server											
DeactivateAll	Deactivate all LPR Configurations from the server											

* Mandatory parameters

Example 1: Activate all LPR Configurations from the server

```
http://192.168.0.1:8601/Interface/LPR/Activation?Action=ActivateAll
```

Example 2: Activate LPR Configurations LPR1 and LPR2

```
http://192.168.0.1:8601/Interface/LPR/Activation?Action=Activate&
LPRConfigurations=LPR1,LPR2
```

Example 3: Deactivate all LPR Configurations from the server

```
http://192.168.0.1:8601/Interface/LPR/Activation?Action=DeactivateAll
```

Response:

Default response of API.

HTTP Return HTTP: 200 OK

Parameters of return: [Default return of API](#)

4.14.5 Searching for LPR records

Perform a search for LPR records in the database

Compatibility: Standard, Professional, Enterprise

Security level: Requires user authentication with rights to search for LPR records

Method: HTTP GET**Syntax:**

```
http://<server_address>/Interface/LPR/Search
[?<argument=value> [&<argument=value>... ] [&<general\_argument>... ] ]
```

Arguments:

Argument	Valid values	Description								
StartDate	APIDate	Search for records starting with the specified date								
StartTime	APITime	Search for records starting with the specified time								
EndDate	APIDate	End date of filter. Search for all records inside the range of start and end. If this parameter is omitted, the value specified in <code>StartDate</code> parameter will be used								
EndTime	APITime	End time of filter. Search for all records inside the range of start and end. If this parameter is omitted, the value specified in <code>StartTime</code> parameter will be used								
Cameras	String	List of cameras to search. The list with camera names must be comma-separated								
LicensePlates	String	List of license plates to filter. The list with license plates must be comma-separated. The plates could include the special character asterisk ("*") that can be used as mask for search								
LPRConfigurations	String	List of LPR Configurations to filter. The list with LPR Configuration names must be comma-separated								
SuppressRepeatedPlates	String	This parameter will determine what to do when a plate is found more than once in the results. This is useful if you want to have a search containing unique plates only. If this parameter is omitted, the default value of <code>Deactivated</code> will be used. <table border="1" data-bbox="816 1389 1436 1594"> <thead> <tr> <th>Operation</th><th>Description</th></tr> </thead> <tbody> <tr> <td>Deactivated</td><td>Do not supress repeated plates</td></tr> <tr> <td>ShowFirst</td><td>Show only the first record containing the repeated plate</td></tr> <tr> <td>ShowLast</td><td>Show only the last record containing the repeated plate</td></tr> </tbody> </table>	Operation	Description	Deactivated	Do not supress repeated plates	ShowFirst	Show only the first record containing the repeated plate	ShowLast	Show only the last record containing the repeated plate
Operation	Description									
Deactivated	Do not supress repeated plates									
ShowFirst	Show only the first record containing the repeated plate									
ShowLast	Show only the last record containing the repeated plate									
Countries	String	List of recognized countries to filter. The list must be comma-separated.								
PlateColors	String	List of recognized plate color to filter (RGB values). The list must be comma-separated.								
BackgroundColors	String	List of recognized plate background color to filter (RGB Values). The list must be comma-separated.								
Categories	String	List of plate categories to filter. The list must be comma-separated.								
Lists	String	List of plate lists to filter. The list must be comma-								

		separated.																
Owners	String	List of owner names to filter. The list must be comma-separated.																
SpeedFrom	Integer	Starting speed value to filter.																
SpeedTo	Integer	Ending speed value to filter.																
Classifications	String	List of vehicle classifications to filter. The list must be comma-separated.																
Manufacturers	String	List of vehicle manufacturers to filter. The list must be comma-separated.																
VehicleColors	String	List of vehicle colors to search (RGB Values). The list must be comma-separated.																
VehicleModels	String	List of vehicle models to search. The list must be comma-separated.																
WithLPRBridgeData	TRUE FALSE	Filter and return only records that contain LPR Bridge Result Data																
OrderBy	String	<p>Sort the records by the specified column.</p> <table border="1" style="margin-left: 20px; border-collapse: collapse;"> <thead> <tr style="background-color: #ADD8E6;"> <th>Columns</th> </tr> </thead> <tbody> <tr><td>RecordNumber</td></tr> <tr><td>DateTime</td></tr> <tr><td>LicensePlate</td></tr> <tr><td>Camera</td></tr> <tr><td>LPRConfiguration</td></tr> <tr><td>Country</td></tr> <tr><td>PlateColor</td></tr> <tr><td>BackgroundColor</td></tr> <tr><td>Category</td></tr> <tr><td>List</td></tr> <tr><td>Speed</td></tr> <tr><td>VehicleType</td></tr> <tr><td>VehicleModel</td></tr> <tr><td>Manufacturer</td></tr> <tr><td>VehicleColor</td></tr> </tbody> </table> <p>If this parameter is omitted, the records will be displayed in retrieval order (Faster method with less server memory usage)</p>	Columns	RecordNumber	DateTime	LicensePlate	Camera	LPRConfiguration	Country	PlateColor	BackgroundColor	Category	List	Speed	VehicleType	VehicleModel	Manufacturer	VehicleColor
Columns																		
RecordNumber																		
DateTime																		
LicensePlate																		
Camera																		
LPRConfiguration																		
Country																		
PlateColor																		
BackgroundColor																		
Category																		
List																		
Speed																		
VehicleType																		
VehicleModel																		
Manufacturer																		
VehicleColor																		

Example 1: Search for all records between January 01, 2014 and February 01, 2014

```
http://192.168.0.1:8601/Interface/LPR/Search?StartDate=2014.01.01&
StartTime=00.00.00.000&EndDate=2014.02.01&EndTime=23.59.59.999
```

Example 2: Search for all records of plates ABC0001 and DEF1234

```
http://192.168.0.1:8601/Interface/LPR/Search?LicensePlates=ABC0001,DEF1234
```

Example 3: Search for all plates starting with "A", sorting the result by license plate

```
http://192.168.0.1:8601/Interface/LPR/Search?LicensePlates=A*&
OrderBy=LicensePlate
```

Response:

A list with all found LPR records is returned.

HTTP Return: 200 OK

Parameters of return:

Fixed Parameter:

Parameter	Type	Description
COUNT	Integer	Total of records (Not included in JSON response)

Parameters of records list: Refer to [LPR Record Details](#)

4.14.6 Requesting the data of an LPR record

Request the data of an LPR record.

Compatibility: Standard, Professional, Enterprise

Security level: Requires user authentication with rights to search for LPR records

Method: HTTP GET

Syntax:

```
http://<server_address>/Interface/LPR/GetRecordData?<argument=value>
[&<argument=value>...] [&<general argument>...]
```

Arguments:

Argument	Valid values	Description
RecordNumber *	Integer. X >= 0	LPR record number

* Mandatory parameters

Example 1: Request the record 40 with response in XML

```
http://192.168.0.1:8601/Interface/LPR/GetRecordData?RecordNumber=40&
ResponseFormat=XML
```

Example 2: Request the record 237 with response in text

```
http://192.168.0.1:8601/Interface/LPR/GetRecordData?RecordNumber=237&
ResponseFormat=Text
```

Example 3: Request the record 500 with response in XML and authentication with Admin user

```
http://192.168.0.1:8601/Interface/LPR/GetRecordData?RecordNumber=500&
ResponseFormat=XML&AuthUser=admin
```

Response:

If the record is found, a list of parameter-value pairs is returned, otherwise, the error 4000 Record not found is returned

HTTP Return: 200 OK

Parameters of return: Refer to [LPR Record Details](#)

4.14.7 Requesting the image of an LPR record

Request the image of an LPR record.

Compatibility: Standard, Professional, Enterprise

Security level: Requires user authentication with rights to search for LPR records

Method: HTTP GET

Syntax:

```
http://<server_address>/Interface/LPR/GetRecordImage?<argument=value>
[&<argument=value>...] [&<general\_argument>...]
```

Arguments:

Argument	Valid values	Description
RecordNumber *	Integer. X >= 0	LPR record number
PlateCrop	TRUE FALSE	Request the crop of the plate only If this parameter is omitted, the default value FALSE will be used

* Mandatory parameters

Example 1: Request the image of record 40 with error response in XML

```
http://192.168.0.1:8601/Interface/LPR/GetRecordImage?RecordNumber=40&
ResponseFormat=XML
```

Example 2: Request the image of record 237 with error response in text

```
http://192.168.0.1:8601/Interface/LPR/GetRecordImage?RecordNumber=237&
ResponseFormat=Text
```

Example 3: Request the image of record 500 with error response in XML and authentication with Admin user

```
http://192.168.0.1:8601/Interface/LPR/GetRecordImage?RecordNumber=500&
ResponseFormat=XML&AuthUser=admin
```

Response:

If the record is found, its JPEG image is returned, otherwise, the error 4000 Record not found is returned.

HTTP Return: 200 OK

Parameters of return: [Default return of API](#)

4.14.8 Requesting the data of the last LPR record

Request the data of the last LPR record of a camera or database.

Compatibility: Standard, Professional, Enterprise

Security level: Requires user authentication with rights to search for LPR records

Method: HTTP GET

Syntax:

```
http://<server_address>/Interface/LPR/GetLastRecordData[?<argument=value>
[&<argument=value>...] [&<general\_argument>...]]
```

Arguments:

Argument	Valid values	Description
LPRConfiguration	String	LPR Configuration name. If this parameter is omitted, the command will return the last database record, regardless of LPR configuration. Use either this parameter or Camera parameter
Camera	String	Camera name. If this parameter is omitted, the command will return the last database record, regardless of camera.

Example 1: Request the last record of a camera with response in XML

```
http://192.168.0.1:8601/Interface/LPR/GetLastRecordData?Camera=Camera1&ResponseFormat=XML
```

Example 2: Request the last record of an LPR configuration with response in text

```
http://192.168.0.1:8601/Interface/LPR/GetLastRecordData?  
LPRConfiguration=Config1&ResponseFormat=Text
```

Example 3: Request the last record of the database with response in XML

```
http://192.168.0.1:8601/Interface/LPR/GetLastRecordData?ResponseFormat=XML
```

Response:

If the record is found, a list of parameter-value pairs is returned, otherwise, the error 4000 Record not found is returned

HTTP Return: 200 OK

Parameters of return: Refer to [LPR Record Details](#)

4.14.9 Requesting the image of the last LPR record

Request the image of the last LPR record of a camera or database.

Compatibility: Standard, Professional, Enterprise

Security level: Requires user authentication with rights to search for LPR records

Method: HTTP GET

Syntax:

```
http://<server_address>/Interface/LPR/GetLastRecordImage[?<argument=value>  
[&<argument=value>...][&<general\_argument>...]]
```

Arguments:

Argument	Valid values	Description
LPRConfiguration	String	LPR Configuration name. If this parameter is omitted, the command will return the last database record, regardless of LPR configuration. Use either this parameter or Camera parameter
Camera	String	Camera name. If this parameter is omitted, the command will return the last database record, regardless of camera.

PlateCrop	TRUE FALSE	Request the crop of the plate only If this parameter is omitted, the default value FALSE will be used
-----------	---------------	--

Example 1: Request the image of the last record of a camera with error response in XML

```
http://192.168.0.1:8601/Interface/LPR/GetLastRecordImage?Camera=Camera1&ResponseFormat=XML
```

Example 2: Request the image of the last record of an LPR Configuration with error response in text

```
http://192.168.0.1:8601/Interface/LPR/GetLastRecordImage?  
LPRConfiguration=Config1&ResponseFormat=Text
```

Example 3: Request the image of the last record of database with error response in text

```
http://192.168.0.1:8601/Interface/LPR/GetLastRecordImage?ResponseFormat=XML
```

Response:

If the record is found, its JPEG image is returned, otherwise, the error 4000 Record not found is returned.

HTTP Return: 200 OK

Parameters of return: [Default return of API](#)

4.14.10 Triggering the recognition

This command should be used as a kind of external sensor to start or stop the license plate recognition from a LPR Configuration.

Note: This command will only work in LPR Configurations set to use the external sensor

Compatibility: Standard, Professional, Enterprise

Security level: Requires user authentication with rights to LPR Configuration Registration

Method: HTTP GET

Syntax:

```
http://<server_address>/Interface/LPR/ExternalSensorControl?  
<argument=value>[&<argument=value>...][&<general argument>...]
```

Arguments:

Argument	Valid values	Description								
LPRConfiguration*	String	Name of the LPR Configuration								
Action	Start Stop Instant	Action to be executed <table border="1"> <thead> <tr> <th>Operation</th><th>Description</th></tr> </thead> <tbody> <tr> <td>Start</td><td>Start the recognition</td></tr> <tr> <td>Stop</td><td>Stop the recognition</td></tr> <tr> <td>Instant</td><td>Instant recognition</td></tr> </tbody> </table> If this parameter is omitted, the default value of "Instant" will be used	Operation	Description	Start	Start the recognition	Stop	Stop the recognition	Instant	Instant recognition
Operation	Description									
Start	Start the recognition									
Stop	Stop the recognition									
Instant	Instant recognition									

* Mandatory parameters

Usage:

When the LPR Configuration is set to use an external sensor, the system will not start the license plate recognition until this API command is called, so this command should be used to start or stop the license plate recognition according to the integration requirements of an external system.

There are two ways to work with an external sensor: Start/Stop and Instant.

Start/Stop: In this type of operation, this trigger command should be called a time to start the recognition and again to stop the recognition. Once started, the system will begin to analyse all the received images in search for license plates and will stop only when this trigger command is called with the parameter Action=Stop.

Instant: In this type of operation, after calling this trigger command with parameter Action=Instant, the system will analyse the next three received images in search of license plates and will automatically stop, waiting for the next call for analysis.

Example 1: Start the license plate recognition from a configuration with response in text

```
http://192.168.0.1:8601/Interface/LPR/ExternalSensorControl?
LPRConfiguration=LPR1&Action=Start&ResponseFormat=Text
```

Example 2: Stop the license plate recognition from a configuration with response in XML

```
http://192.168.0.1:8601/Interface/LPR/ExternalSensorControl?
LPRConfiguration=LPR1&Action=Stop&ResponseFormat=XML
```

Example 3: Start the instant plate recognition from a configuration with response in XML

```
http://192.168.0.1:8601/Interface/LPR/ExternalSensorControl?
LPRConfiguration=LPR1&Action=Instant&ResponseFormat=XML
```

Response:

Default response of API.

HTTP Return: 200 OK

Parameters of return: [Default return of API](#)

4.14.11 Monitoring the LPR events

Starts the monitoring of the LPR events from the server. By using this command, you will receive in real time, the characters of recognized license plates from all LPR Configurations.

Compatibility: Standard, Professional, Enterprise

Security level: Requires user authentication

Method: HTTP GET

Syntax:

```
http://<server_address>/Interface/LPR/MonitorEvents
[?<argument=value> [&<argument=value>...]] [&<general\_argument>...]
```

Arguments:

Argument	Valid values	Description
----------	--------------	-------------

KeepAliveInterval	Integer. 1 >= X <= 120	Specify the time (in seconds) of Keep-Alive packet sending. The Keep-Alive packet is used to control the connection. If this parameter is omitted, the default value of 5 will be used
LPRConfigurations	API Masks	Mask to filter the results. Specify which LPR Configurations must be returned based on the provided masks.
IncludePlateDetails	TRUE FALSE	Include the details of the recognized plate. If this parameter is omitted, the default value FALSE will be used.
IncludeImage	TRUE FALSE	Include the full image of the recognition in Base64 format. If this parameter is omitted, the default value FALSE will be used.
IncludePlateCropImage	TRUE FALSE	Include plate crop image of the recognition in Base64 format. If this parameter is omitted, the default value FALSE will be used.

Example 1: Start the LPR event monitoring with response in XML

```
http://192.168.0.1:8601/Interface/LPR/MonitorEvents?ResponseFormat=XML
```

Example 2: Start the LPR event monitoring with response in text and keep-alive interval of 60 seconds

```
http://192.168.0.1:8601/Interface/LPR/MonitorEvents?ResponseFormat=Text&KeepAliveInterval=60
```

Example 3: Start the LPR event monitoring with response in XML and authentication with Admin user

```
http://192.168.0.1:8601/Interface/LPR/MonitorEvents?  
ResponseFormat=XML&AuthUser=admin&AuthPass=pass
```

Response:

A stream of events data will be sent using HTTP Multipart `x-mixed-replace` transmission. The response data stream is continuous, so an event of Keep-Alive will be sent every X seconds (Configurable by `KeepAliveInterval` parameter), with this event you can verify if the TCP connection is still open.

Each event is separated by the multipart boundary `--DigifortBoundary`.

HTTP Return: 200 OK

Parameter of return:

The return parameters are divided into two subcategories: `EventData` and `ObjectData`.

EventData parameters:

Parameter	Type	Description						
TYPE	String	Event type <table border="1" data-bbox="714 1805 1416 1926"> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>PLATE_RECOGNIZED</td><td>Recognized plate event</td></tr> <tr> <td>KEEP_ALIVE</td><td>HTTP Connection Keep-Alive</td></tr> </tbody> </table>	Value	Description	PLATE_RECOGNIZED	Recognized plate event	KEEP_ALIVE	HTTP Connection Keep-Alive
Value	Description							
PLATE_RECOGNIZED	Recognized plate event							
KEEP_ALIVE	HTTP Connection Keep-Alive							

Parameter	Type	Description
		message. Sent every 5 seconds to keep the TCP / HTTP connection
RECORDNUMBER	Integer	LPR record number
TIMESTAMP	APITimestamp	Date and time of the event
UTCTIMESTAMP	APITimestamp	UTC date and time of the event
PLATE	String	Characters of the recognized license plate
PLATEDETAILS	PlateDetails	If IncludePlateDetails=TRUE this field will contain the details of the record. Please refer to LPR Record Details
IMAGE	String	If IncludeImage=TRUE this field will provide a JPEG Image in Base64 format
PLATECROPIMAGE	String	If IncludePlateCropImage=TRUE this field will provide a JPEG Image in Base64 format

ObjectData parameters:

Parameter	Type	Description
NAME	String	Name of the LPR Configuration that triggered the event
CAMERA	String	Name of the camera associated with the LPR Configuration that triggered the event

Example of return in text:

```
--DigifortBoundary
Content-Type: text/plain; charset=UTF-8
Content-Length: 2480

RESPONSE_CODE=0
RESPONSE_MESSAGE=OK
EVENTDATA_TYPE=PLATE_RECOGNIZED
EVENTDATA_RECORDNUMBER=26931
EVENTDATA_TIMESTAMP=2020-06-11 11:33:32.484
EVENTDATA_UTCTIMESTAMP=2020-06-11 14:33:32.484
EVENTDATA_PLATE=CZQ2457
OBJECTDATA_NAME=LPR
OBJECTDATA_CAMERA=51
PLATEDETAILS_RECORDNUMBER=26931
PLATEDETAILS_DATETIME=2020-06-11 11:33:30.414
PLATEDETAILS_LPRCONFIGURATION=LPR
PLATEDETAILS_CAMERA=51
PLATEDETAILS_LICENSEPLATE=CZQ2457
PLATEDETAILS_LICENSEPLATECODE=
PLATEDETAILS_LATITUDE=0.000000
PLATEDETAILS_LONGITUDE=0.000000
PLATEDETAILS_LISTS=Plates ending in 7 and 8
PLATEDETAILS_COUNTRY=
PLATEDETAILS_CATEGORY=
PLATEDETAILS_OWNER=
PLATEDETAILS_REMARKS=
PLATEDETAILS_PLATECOLOR=0
PLATEDETAILS_BACKGROUNDCOLOR=0
PLATEDETAILS_MANUFACTURER=
PLATEDETAILS_VEHICLETYPE=
PLATEDETAILS_VEHICLEMODEL=
PLATEDETAILS_VEHICLECOLOR=-1
PLATEDETAILS_SPEED=0
PLATEDETAILS_SPEEDMETRIC=km/h
PLATEDETAILS_RELIABILITY=Medium
PLATEDETAILS_HIT=Hit
PLATEDETAILS_CONFIDENCE=74
PLATEDETAILS_PLATERECT_LEFT=1028
PLATEDETAILS_PLATERECT_TOP=856
PLATEDETAILS_PLATERECT_RIGHT=1279
PLATEDETAILS_PLATERECT_BOTTOM=928
PLATEDETAILS_TIPS_COUNT=7
TIP_1_CONFIDENCE=74
TIP_1_COLOR=0
TIP_1_BACKGROUNDCOLOR=0
TIP_1_CODE=0
TIP_1_COORDINATES_LEFT=1035
TIP_1_COORDINATES_TOP=877
TIP_1_COORDINATES_RIGHT=1064
TIP_1_COORDINATES_BOTTOM=920
TIP_2_CONFIDENCE=74
TIP_2_COLOR=0
TIP_2_BACKGROUNDCOLOR=0
TIP_2_CODE=0
TIP_2_COORDINATES_LEFT=1069
TIP_2_COORDINATES_TOP=875
TIP_2_COORDINATES_RIGHT=1097
TIP_2_COORDINATES_BOTTOM=914
TIP_3_CONFIDENCE=74
```

Example of return in XML:

```
--DigifortBoundary
Content-Type: text/xml; charset=UTF-8
Content-Length: 2631

<?xml version="1.0" encoding="UTF-8" ?>
<Response>
    <Code>0</Code>
    <Message>OK</Message>
    <Data>
        <Event>
            <EventData>
                <Type>PLATE_RECOGNIZED</Type>
                <RecordNumber>26923</RecordNumber>
                <Timestamp>2020-06-11 11:32:59.595</Timestamp>
                <UTCTimestamp>2020-06-11 14:32:59.595</UTCTimestamp>
                <Plate>CZD2457</Plate>
            </EventData>
            <ObjectData>
                <Name>LPR</Name>
                <Camera>51</Camera>
            </ObjectData>
            <PlateDetails>
                <RecordNumber>26923</RecordNumber>
                <DateTime>2020-06-11 11:32:58.357</DateTime>
                <LPRConfiguration>LPR</LPRConfiguration>
                <Camera>51</Camera>
                <LicensePlate>CZD2457</LicensePlate>
                <LicensePlateCode></LicensePlateCode>
                <Latitude>0.000000</Latitude>
                <Longitude>0.000000</Longitude>
                <Lists>Plates ending in 7 and 8</Lists>
                <Country></Country>
                <Category></Category>
                <Owner></Owner>
                <Remarks></Remarks>
                <PlateColor>0</PlateColor>
                <BackgroundColor>0</BackgroundColor>
                <Manufacturer></Manufacturer>
                <VehicleType></VehicleType>
                <VehicleModel></VehicleModel>
                <VehicleColor>-1</VehicleColor>
                <Speed>0</Speed>
                <SpeedMetric>km/h</SpeedMetric>
                <Reliability>Medium</Reliability>
                <Hit>Hit</Hit>
                <Confidence>77</Confidence>
                <PlateRect>
                    <Left>1037</Left>
                    <Top>847</Top>
                    <Right>1288</Right>
                    <Bottom>920</Bottom>
                </PlateRect>
                <Tips>
                    <Count>7</Count>
                    <Tip>
                        <Confidence>77</Confidence>
                        <Color>0</Color>
                        <BackgroundColor>0</BackgroundColor>
                        <Code>0</Code>
                    </Tip>
                </Tips>
            </PlateDetails>
        </Event>
    </Data>
</Response>
```

Example of return in JSON:

```
--DigifortBoundary
Content-Type: application/json; charset=UTF-8
Content-Length: 2631

{
    "Response": {
        "Code": 0,
        "Message": "OK",
        "Data": {
            "Event": {
                "EventData": {
                    "Type": "PLATE_RECOGNIZED",
                    "RecordNumber": 26938,
                    "Timestamp": "2020-06-11T11:34:22.424Z",
                    "UTCTimestamp": "2020-06-11T14:34:22.424Z",
                    "Plate": "CZU2457"
                },
                "ObjectData": {
                    "Name": "LPR",
                    "Camera": "51"
                },
                "PlateDetails": {
                    "RecordNumber": 26938,
                    "DateTime": "2020-06-11T11:34:20.361Z",
                    "LPRConfiguration": "LPR",
                    "Camera": "51",
                    "LicensePlate": "CZU2457",
                    "LicensePlateCode": "",
                    "Latitude": "0.000000",
                    "Longitude": "0.000000",
                    "Lists": "Plates ending in 7 and 8",
                    "Country": "",
                    "Category": "",
                    "Owner": "",
                    "Remarks": "",
                    "PlateColor": 0,
                    "BackgroundColor": 0,
                    "Manufacturer": "",
                    "VehicleType": "",
                    "VehicleModel": "",
                    "VehicleColor": -1,
                    "Speed": 0,
                    "SpeedMetric": "km/h",
                    "Reliability": "Medium",
                    "Hit": "Hit",
                    "Confidence": 83,
                    "PlateRect": {
                        "Left": 1029,
                        "Top": 850,
                        "Right": 1280,
                        "Bottom": 923
                    },
                    "Tips": [
                        {
                            "Confidence": 83,
                            "Color": 0,
                            "BackgroundColor": 0,
                            "Code": 0,
                            "Coordinates": {
                                "X": 1029,
                                "Y": 850
                            }
                        }
                    ]
                }
            }
        }
    }
}
```

4.14.12 Processing an external image

Input an external JPEG image for processing and retrieve the result of the LPR reading.

Compatibility: Standard, Professional, Enterprise

Security level: Requires user authentication

Method: HTTP POST

Syntax:

```
http://<server_address>/Interface/LPR/ProcessImage?<argument=value>
[&<argument=value>...] [&<general_argument>...]
```

Arguments:

Argument	Valid values	Description
JPEGImage*	String	Base64 encoded JPEG image
LPRConfiguration*	String	Name of the LPR Configuration that should process the image

* Mandatory parameters

Response:

If the record is found, a list of parameter-value pairs is returned, otherwise, the error 4000 Record not found is returned

HTTP Return: 200 OK

Parameters of return: Refer to [LPR Record Details](#)

4.14.13 Querying data from a license plate

Query data from the specified license plate. The system will return data from license plate registration or LPR Bridge integration.

Compatibility: Standard, Professional, Enterprise

Security level: Requires user authentication

Method: HTTP GET

Syntax:

```
http://<server_address>/Interface/LPR/QueryPlate?<argument=value>
[&<argument=value>...] [&<general_argument>...]
```

Arguments:

Argument	Valid values	Description
LPRConfiguration*	String	Name of the LPR Configuration that should query the plate
LicensePlate*	String	License plate number

* Mandatory parameters

Example 1: Request the data of plate DIQ6939 using LPR Configuration LPR1 with response in text

```
http://192.168.0.1:8601/Interface/LPR/QueryPlate?LPRConfiguration=LPR1&
LicensePlate=DIQ6939&ResponseFormat=Text
```

Example 2: Request the data of plate DIQ6939 using LPR Configuration LPR1 with response in JSON
 http://192.168.0.1:8601/Interface/LPR/QueryPlate?LPRConfiguration=LPR1&
 LicensePlate=DIQ6939&ResponseFormat=JSON

Response:

The available information for the plate will be returned

HTTP Return: 200 OK

Parameters of return: Refer to [LPR Record Details](#)

4.14.14 LPR Record Details

Parameter	Type	Description										
RECORDNUMBER	Integer	Record number										
DATETIME	APITimestamp	Date and time of the record in timestamp format										
LPRCONFIGURATION	String	Name of the LPR Configuration										
CAMERA	String	Camera name										
LICENSEPLATE	String	Recognized plate characters										
LICENSEPLATECODE	String	Special code used for Arabic plates										
LATITUDE	APILatLnG	Latitude with 6 digit precision										
LONGITUDE	APILatLnG	Longitude with 6 digit precision										
LISTS	String	Lists where the plate was recognized (Comma separated)										
COUNTRY	String	Recognized country										
CATEGORY	String	Plate category										
OWNER	String	Recognized owner (From internal plate list)										
REMARKS	String	Plate remarks (From internal plate list)										
PLATECOLOR	Integer	Recognized RGB Color of the plate										
BACKGROUNDCOLOR	Integer	Recognized RGB Color of the plate background										
MANUFACTURER	String	Identified vehicle manufacturer										
VEHICLETYPEn	String	Identified vehicle type										
VEHICLEMODEL	String	Identified vehicle model										
VEHICLECOLOR	Integer	Identified RGB Color of the vehicle										
SPEED	Integer	Vehicle speed										
SPEEDMETRIC	String	Metric of vehicle speed <table border="1" style="margin-left: 20px;"> <tr> <th>Values</th> </tr> <tr> <td>km/h</td> </tr> <tr> <td>mph</td> </tr> </table>	Values	km/h	mph							
Values												
km/h												
mph												
RELIABILITY	String	Reliability of reading <table border="1" style="margin-left: 20px;"> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>No Reliability</td> <td>No reliability value. The used LPR engine does not provide reliability values</td> </tr> <tr> <td>Low</td> <td>Low reliability</td> </tr> <tr> <td>Medium</td> <td>Medium reliability</td> </tr> <tr> <td>High</td> <td>High reliability</td> </tr> </tbody> </table>	Value	Description	No Reliability	No reliability value. The used LPR engine does not provide reliability values	Low	Low reliability	Medium	Medium reliability	High	High reliability
Value	Description											
No Reliability	No reliability value. The used LPR engine does not provide reliability values											
Low	Low reliability											
Medium	Medium reliability											
High	High reliability											
HIT	String	This value identifies if the engine believes having This value identifies whether the engine believed to have hit or										

Parameter	Type	Description								
		wrong the recognized value <table border="1" data-bbox="780 325 1416 530"> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>Error</td><td>Error on plate recognition</td></tr> <tr> <td>Hit</td><td>The engine believes to have hit the value</td></tr> <tr> <td>Uncertainty</td><td>The engine is not sure to have hit the value</td></tr> </tbody> </table>	Value	Description	Error	Error on plate recognition	Hit	The engine believes to have hit the value	Uncertainty	The engine is not sure to have hit the value
Value	Description									
Error	Error on plate recognition									
Hit	The engine believes to have hit the value									
Uncertainty	The engine is not sure to have hit the value									
CONFIDENCE	Integer	Confidence value Range from 0 to 100								
PLATERECT LEFT	Integer	Position of the plate rect on the image (Left)								
PLATERECT TOP	Integer	Position of the plate rect on the image (Top)								
PLATERECT RIGHT	Integer	Position of the plate rect on the image (Right)								
PLATERECT BOTTOM	Integer	Position of the plate rect on the image (Bottom)								
TIPS	TipsList	List with the information on all character tips on the plate								
LPR BRIDGE DATA	LPRBridgeList	List with data captured from LPR Bridge								

Fixed parameters of Tips list:

Parameter	Type	Description
COUNT	Integer	Total number of tips (Not included in JSON response)

Parameters in the list of tips:

Parameter	Type	Description
CONFIDENCE	Integer	Confidence value Range from 0 to 100
COLOR	Integer	RGB Color of character tip
BACKGROUNDCOLOR	Integer	RGB Color of character tip background
CODE	Integer	Unicode tip code
COORDINATES LEFT	String	Position of the tip rect on the plate (Left)
COORDINATES TOP	Integer	Position of the tip rect on the plate (Top)
COORDINATES RIGHT	Integer	Position of the tip rect on the plate (Right)
COORDINATES BOTTOM	Integer	Position of the tip rect on the plate (Bottom)

Fixed parameters of LPRBridge List:

Parameter	Type	Description
COUNT	Integer	Total number of records (Not included in JSON response)

Parameters in the list of records:

Parameter	Type	Description
NAME	String	Field name
VALUE	String	Field value

Example of return in text:

RECORDNUMBER=25280
DATETIME=2020-06-10 14:53:47.257
LPRCONFIGURATION=LPR_01
CAMERA=01
LICENSEPLATE=FCS5070
LICENSEPLATECODE=
LATITUDE=-23.630363
LONGITUDE=-46.554916
LISTS=Plates ending in 9 and 0
COUNTRY=
CATEGORY=
OWNER=
REMARKS=
PLATECOLOR=0
BACKGROUNDCOLOR=0
MANUFACTURER=
VEHICLETYPEn=
VEHICLEMODEL=
VEHICLECOLOR=-1
SPEED=0
SPEEDMETRIC=km/h
RELIABILITY=High
HIT=Hit
CONFIDENCE=93
PLATERECT_LEFT=1064
PLATERECT_TOP=559
PLATERECT_RIGHT=1143
PLATERECT_BOTTOM=618
TIPS_COUNT=7
TIP_1_CONFIDENCE=93
TIP_1_COLOR=0
TIP_1_BACKGROUNDCOLOR=0
TIP_1_CODE=0
TIP_1_COORDINATES_LEFT=1067
TIP_1_COORDINATES_TOP=566
TIP_1_COORDINATES_RIGHT=1083
TIP_1_COORDINATES_BOTTOM=588
TIP_2_CONFIDENCE=93
TIP_2_COLOR=0
TIP_2_BACKGROUNDCOLOR=0
TIP_2_CODE=0
TIP_2_COORDINATES_LEFT=1092
TIP_2_COORDINATES_TOP=565
TIP_2_COORDINATES_RIGHT=1109
TIP_2_COORDINATES_BOTTOM=587
TIP_3_CONFIDENCE=93
TIP_3_COLOR=0
TIP_3_BACKGROUNDCOLOR=0
TIP_3_CODE=0
TIP_3_COORDINATES_LEFT=1116
TIP_3_COORDINATES_TOP=564
TIP_3_COORDINATES_RIGHT=1133
TIP_3_COORDINATES_BOTTOM=586
TIP_4_CONFIDENCE=93
TIP_4_COLOR=0
TIP_4_BACKGROUNDCOLOR=0
TIP_4_CODE=0
TIP_4_COORDINATES_LEFT=1071
TIP_4_COORDINATES_TOP=591

Example of return in XML:

```
<RecordNumber>25280</RecordNumber>
<DateTime>2020-06-10 14:53:47.257</DateTime>
<LPRConfiguration>LPR_01</LPRConfiguration>
<Camera>01</Camera>
<LicensePlate>FCS5070</LicensePlate>
<LicensePlateCode />
<Latitude>-23.630363</Latitude>
<Longitude>-46.554916</Longitude>
<Lists>Plates ending in 9 and 0</Lists>
<Country />
<Category />
<Owner />
<Remarks />
<PlateColor>0</PlateColor>
<BackgroundColor>0</BackgroundColor>
<Manufacturer />
<VehicleType />
<VehicleModel />
<VehicleColor>-1</VehicleColor>
<Speed>0</Speed>
<SpeedMetric>km/h</SpeedMetric>
<Reliability>High</Reliability>
<Hit>Hit</Hit>
<Confidence>93</Confidence>
<PlateRect>
    <Left>1064</Left>
    <Top>559</Top>
    <Right>1143</Right>
    <Bottom>618</Bottom>
</PlateRect>
<Tips>
    <Count>7</Count>
    <Tip>
        <Confidence>93</Confidence>
        <Color>0</Color>
        <BackgroundColor>0</BackgroundColor>
        <Code>0</Code>
        <Coordinates>
            <Left>1067</Left>
            <Top>566</Top>
            <Right>1083</Right>
            <Bottom>588</Bottom>
        </Coordinates>
    </Tip>
    <Tip>
        <Confidence>93</Confidence>
        <Color>0</Color>
        <BackgroundColor>0</BackgroundColor>
        <Code>0</Code>
        <Coordinates>
            <Left>1092</Left>
            <Top>565</Top>
            <Right>1109</Right>
            <Bottom>587</Bottom>
        </Coordinates>
    </Tip>
    <Tip>
        <Confidence>93</Confidence>
        <Color>0</Color>
```

Example of return in JSON:

```
{  
    "RecordNumber": 25280,  
    "DateTime": "2020-06-10T14:53:47.257Z",  
    "LPRConfiguration": "LPR_01",  
    "Camera": "01",  
    "LicensePlate": "FCS5070",  
    "LicensePlateCode": "",  
    "Latitude": "-23.630363",  
    "Longitude": "-46.554916",  
    "Lists": "Plates ending in 9 and 0",  
    "Country": "",  
    "Category": "",  
    "Owner": "",  
    "Remarks": "",  
    "PlateColor": 0,  
    "BackgroundColor": 0,  
    "Manufacturer": "",  
    "VehicleType": "",  
    "VehicleModel": "",  
    "VehicleColor": -1,  
    "Speed": 0,  
    "SpeedMetric": "km/h",  
    "Reliability": "High",  
    "Hit": "Hit",  
    "Confidence": 93,  
    "PlateRect": {  
        "Left": 1064,  
        "Top": 559,  
        "Right": 1143,  
        "Bottom": 618  
    },  
    "Tips": [  
        {  
            "Confidence": 93,  
            "Color": 0,  
            "BackgroundColor": 0,  
            "Code": 0,  
            "Coordinates": {  
                "Left": 1067,  
                "Top": 566,  
                "Right": 1083,  
                "Bottom": 588  
            }  
        },  
        {  
            "Confidence": 93,  
            "Color": 0,  
            "BackgroundColor": 0,  
            "Code": 0,  
            "Coordinates": {  
                "Left": 1092,  
                "Top": 565,  
                "Right": 1109,  
                "Bottom": 587  
            }  
        },  
        {  
            "Confidence": 93,  
            "Color": 0,  
            "BackgroundColor": 0,  
            "Code": 0,  
            "Coordinates": {  
                "Left": 1092,  
                "Top": 565,  
                "Right": 1109,  
                "Bottom": 587  
            }  
        }  
    ]  
}
```

4.14.15 Zones

4.14.15.1 Requesting the list of LPR Zones

Request the list of LPR Zones over which the user has rights.

Compatibility: Professional, Enterprise

Security level: Requires user authentication

Method: HTTP GET

Syntax:

```
http://<server_address>/Interface/LPR/Zones/GetLPRZones[?<argument=value>
[&<argument=value>...][&<general argument>...]]
```

Arguments:

Argument	Valid values	Description																								
LPRZones	APIMasks	Mask to filter the results. Specify which LPR Zones must be returned based on the provided masks.																								
Fields	DUID Name Description Active LPRConfigsIn LPRConfigsOut ProcessingLogic ProcessingLPRLists WarningLevels RemovePlateAfterTime Events	<p>Specifies the list of desired fields. In case this parameter is omitted, all of the fields will be sent.</p> <p>The fields must be separated by commas</p> <table border="1"> <thead> <tr> <th>Name</th><th>Description</th></tr> </thead> <tbody> <tr> <td>DUID</td><td>DUID of the LPR Zone</td></tr> <tr> <td>Name</td><td>Name of the LPR Zone</td></tr> <tr> <td>Description</td><td>Description of the LPR Zone</td></tr> <tr> <td>Active</td><td>LPR Zone is activated or deactivated</td></tr> <tr> <td>LPRConfigsIn</td><td>Comma-separated list of LPR Configurations for Entrance</td></tr> <tr> <td>LPRConfigsOut</td><td>Comma-separated list of LPR Configurations for Exit</td></tr> <tr> <td>ProcessingLogic</td><td>Logic of plate processing</td></tr> <tr> <td>ProcessingLPRLists</td><td>Comma-separated list of LPR Lists for processing</td></tr> <tr> <td>WarningLevels</td><td>Warning levels</td></tr> <tr> <td>RemovePlateAfterTime</td><td>Remove a plate from the list after specified time</td></tr> <tr> <td>Events</td><td>Configured events and actions</td></tr> </tbody> </table>	Name	Description	DUID	DUID of the LPR Zone	Name	Name of the LPR Zone	Description	Description of the LPR Zone	Active	LPR Zone is activated or deactivated	LPRConfigsIn	Comma-separated list of LPR Configurations for Entrance	LPRConfigsOut	Comma-separated list of LPR Configurations for Exit	ProcessingLogic	Logic of plate processing	ProcessingLPRLists	Comma-separated list of LPR Lists for processing	WarningLevels	Warning levels	RemovePlateAfterTime	Remove a plate from the list after specified time	Events	Configured events and actions
Name	Description																									
DUID	DUID of the LPR Zone																									
Name	Name of the LPR Zone																									
Description	Description of the LPR Zone																									
Active	LPR Zone is activated or deactivated																									
LPRConfigsIn	Comma-separated list of LPR Configurations for Entrance																									
LPRConfigsOut	Comma-separated list of LPR Configurations for Exit																									
ProcessingLogic	Logic of plate processing																									
ProcessingLPRLists	Comma-separated list of LPR Lists for processing																									
WarningLevels	Warning levels																									
RemovePlateAfterTime	Remove a plate from the list after specified time																									
Events	Configured events and actions																									

Example 1: Request the list of LPR Zones with all fields and response in XML

```
http://192.168.0.1:8601/Interface/LPR/Zones/GetLPRZones?
ResponseFormat=XML
```

Example 2: Request the list of LPR Zones with all fields and response in text

```
http://192.168.0.1:8601/Interface/LPR/Zones/GetLPRZones?
ResponseFormat=Text
```

Example 3: Request the list of LPR Zones with only name and description, response in XML and

authentication of the user Admin

```
http://192.168.0.1:8601/Interface/LPR/Zones/GetLPRZones?Fields=Name,
Description&ResponseFormat=XML&AuthUser=admin
```

Example 4: Request the list of LPR Zones starting with "A", with only name and description, response in XML and authentication with Admin user

```
http://192.168.0.1:8601/Interface/LPR/Zones/GetLPRZones?
LPRZones=A*&Fields=Name,Description&ResponseFormat=XML&
AuthUser=admin
```

Response:

A list of all of the LPR Zones registered in the system is returned. The fields returned in the will depend on the values informed in the argument Fields

HTTP Return: 200 OK

Parameters of return:

Fixed Parameter:

Parameter	Type	Description
COUNT	Integer	Total number of LPR Zones (Not included in JSON response)

Parameters in the list of LPR Zones:

Parameter	Type	Description								
DUID	DUID	DUID of the LPR Zone								
NAME	String	Name of the LPR Zone								
DESCRIPTION	String	Description of the LPR Zone								
ACTIVE	Boolean	Zone activated / deactivated								
LPRCONFIGSIN	String	Comma-separated list of LPR Configurations for Entrance								
LPRCONFIGSOUT	String	Comma-separated list of LPR Configurations for Exit								
PROCESSINGLOGIC	ALL_PLATES PLATES_IN_LIST PLATES_NOT_IN_LIST	Type of plate processing logic <table border="1" data-bbox="889 1320 1455 1573"> <thead> <tr> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>ALL_PLATES</td> <td>Process all plates</td> </tr> <tr> <td>PLATES_IN_LIST</td> <td>Process only plates found in the assigned Plate Lists</td> </tr> <tr> <td>PLATES_NOT_IN_LIST</td> <td>Process only plates that are not found in any assigned Plate Lists</td> </tr> </tbody> </table>	Type	Description	ALL_PLATES	Process all plates	PLATES_IN_LIST	Process only plates found in the assigned Plate Lists	PLATES_NOT_IN_LIST	Process only plates that are not found in any assigned Plate Lists
Type	Description									
ALL_PLATES	Process all plates									
PLATES_IN_LIST	Process only plates found in the assigned Plate Lists									
PLATES_NOT_IN_LIST	Process only plates that are not found in any assigned Plate Lists									
PROCESSINGLPRLISTS	String	Comma-separated list of assigned LPR Lists								
WarningLevels	Type	Description								
ACTIVE	Boolean	Warning levels active								
AVAILABLESPACES	Integer	Number of available spaces for warning levels								
MEDIUMLEVEL	Integer	Percentage of occupied spaces for Medium warning								
HIGHLEVEL	Integer	Percentage of occupied spaces for High warning								

Parameter	Type	Description
RemovePlateAfterTime	Type	Description
ACTIVE	Boolean	Remove plate from zone after specified time
TIME	Integer	Time to remove plate (Minutes)
Events		
DwellTime	Type	Description
ACTIVE	Boolean	Event active
TIME	Integer	Configured time (In minutes)
ACTIONS	Event Actions	List of configured Event Actions. Please refer to Event Actions section for the format of the list.
LeaveBeforeTime		
ACTIVE	Boolean	Event active
TIME	Integer	Configured time (In minutes)
ACTIONS	Event Actions	List of configured Event Actions. Please refer to Event Actions section for the format of the list.
LeaveAfterTime		
ACTIVE	Boolean	Event active
TIME	Integer	Configured time (In minutes)
ACTIONS	Event Actions	List of configured Event Actions. Please refer to Event Actions section for the format of the list.
EnterXTimes		
ACTIVE	Boolean	Event active
TIMES	Integer	Number of times to enter the zone
INTERVAL	Integer	Maximum interval (In minutes)
ACTIONS	Event Actions	List of configured Event Actions. Please refer to Event Actions section for the format of the list.
Occupancy		
COUNT	Integer	Number of occupancy events (Not included in JSON response)
Parameters of each occupancy event		
Event	Type	Description
NAME	String	Event name
DESCRIPTION	String	Event description
OCCUPANCY	Integer	Occupancy threshold
OPERATOR	EQUAL_TO NOT_EQUAL_TO LESS_THAN LESS_THAN_OR_EQUAL_TO GREATER_THAN	Event operator type

Parameter	Type	Description
ACTIONS	GREATER THAN OR EQUAL TO Event Actions	List of configured Event Actions. Please refer to Event Actions section for the format of the list.
NORMALIZATIONEVENT	Boolean	Indicate if Normalization Event is active
NORMALIZATIONEVENTACTION	GREATER THAN OR EQUAL TO Event Actions	List of configured Event Actions. Please refer to Event Actions section for the format of the list.

Example of return in text:

```

RESPONSE_CODE=0
RESPONSE_MESSAGE=OK
COUNT=1
LPRZONE_1_DUID=EF85B01F-6237-4A32-B7DF-CC2DF6322DE9
LPRZONE_1_NAME=Zone 1
LPRZONE_1_DESCRIPTION=LPR Zone 1
LPRZONE_1_LPRCONFIGSIN=Entrance
LPRZONE_1_LPRCONFIGSOUT=Exit
LPRZONE_1_PROCESSINGLOGIC=ALL_PLATES
LPRZONE_1_PROCESSINGLPRLISTS=
LPRZONE_1_WARNINGLEVELS_ACTIVE=TRUE
LPRZONE_1_WARNINGLEVELS_AVAILABLESPACES=250
LPRZONE_1_WARNINGLEVELS_MEDIUMLEVEL=70
LPRZONE_1_WARNINGLEVELS_HIGHLVEL=90
LPRZONE_1_REMOVEPLATEAFTERTIME_ACTIVE=TRUE
LPRZONE_1_REMOVEPLATEAFTERTIME_TIME=60
LPRZONE_1_EVENTS_DWELLTIME_ACTIVE=TRUE
LPRZONE_1_EVENTS_DWELLTIME_TIME=60
LPRZONE_1_EVENTS_LEAVEBEFORETIME_ACTIVE=TRUE
LPRZONE_1_EVENTS_LEAVEBEFORETIME_TIME=60
LPRZONE_1_EVENTS_LEAVEAFTERTIME_ACTIVE=TRUE
LPRZONE_1_EVENTS_LEAVEAFTERTIME_TIME=60
LPRZONE_1_EVENTS_ENTERXTIMES_ACTIVE=TRUE
LPRZONE_1_EVENTS_ENTERXTIMES_TIMES=2
LPRZONE_1_EVENTS_ENTERXTIMES_INTERVAL=60
LPRZONE_1_EVENTS_OCCUPANCY_COUNT=1
LPRZONE_1_EVENTS_OCCUPANCY_EVENT_1_NAME=High
LPRZONE_1_EVENTS_OCCUPANCY_EVENT_1_DESCRIPTION=High occupancy
LPRZONE_1_EVENTS_OCCUPANCY_EVENT_1_OCCUPANCY=200
LPRZONE_1_EVENTS_OCCUPANCY_EVENT_1_OPERATOR=GREATER_THAN_OR_EQUAL_TO
LPRZONE_1_EVENTS_OCCUPANCY_EVENT_1_NORMALIZATIONEVENT=TRUE
LPRZONE_1_ACTIVE=TRUE

```

Example of return in XML:

<Response>

```
<Code>0</Code>
<Message>OK</Message>
<Data>
  <LPRZones>
    <Count>1</Count>
    <LPRZone>
      <DUID>EF85B01F-6237-4A32-B7DF-CC2DF6322DE9</DUID>
      <Name>Zone 1</Name>
      <Description>LPR Zone 1</Description>
      <LPRConfigsIn>Entrance</LPRConfigsIn>
      <LPRConfigsOut>Exit</LPRConfigsOut>
      <ProcessingLogic>ALL_PLATES</ProcessingLogic>
      <ProcessingLPRLists />
      <WarningLevels>
        <Active>TRUE</Active>
        <AvailableSpaces>250</AvailableSpaces>
        <MediumLevel>70</MediumLevel>
        <HighLevel>90</HighLevel>
      </WarningLevels>
      <RemovePlateAfterTime>
        <Active>TRUE</Active>
        <Time>60</Time>
      </RemovePlateAfterTime>
      <Events>
        <DwellTime>
          <Active>TRUE</Active>
          <Time>60</Time>
          <Actions />
        </DwellTime>
        <LeaveBeforeTime>
          <Active>TRUE</Active>
          <Time>60</Time>
          <Actions />
        </LeaveBeforeTime>
        <LeaveAfterTime>
          <Active>TRUE</Active>
          <Time>60</Time>
          <Actions />
        </LeaveAfterTime>
        <EnterXTimes>
          <Active>TRUE</Active>
          <Times>2</Times>
          <Interval>60</Interval>
          <Actions />
        </EnterXTimes>
        <Occupancy>
          <Count>1</Count>
          <Event>
            <Name>High</Name>
            <Description>High occupancy</Description>
            <Occupancy>200</Occupancy>
            <Operator>GREATER_THAN_OR_EQUAL_TO</Operator>
            <Actions />
          </Event>
        </Occupancy>
      </Events>
    </LPRZone>
  </LPRZones>
</Data>
```

```

<NormalizationEvent>TRUE</NormalizationEvent>
<NormalizationEventActions />
</Event>
</Occupancy>
</Events>
<Active>TRUE</Active>
</LPRZone>
</LPRZones>
</Data>
</Response>
```

Example of return in JSON:

```
{
  "Response": {
    "Code": 0,
    "Message": "OK",
    "Data": {
      "LPRZones": [
        {
          "DUID": "EF85B01F-6237-4A32-B7DF-CC2DF6322DE9",
          "Name": "Zone 1",
          "Description": "LPR Zone 1",
          "LPRConfigsIn": "Entrance",
          "LPRConfigsOut": "Exit",
          "ProcessingLogic": "ALL_PLATES",
          "ProcessingLPRLists": "",
          "WarningLevels": {
            "Active": true,
            "AvailableSpaces": 250,
            "MediumLevel": 70,
            "HighLevel": 90
          },
          "RemovePlateAfterTime": {
            "Active": true,
            "Time": 60
          },
          "Events": {
            "DwellTime": {
              "Active": true,
              "Time": 60,
              "Actions": {}
            },
            "LeaveBeforeTime": {
              "Active": true,
              "Time": 60,
              "Actions": {}
            },
            "LeaveAfterTime": {
              "Active": true,
              "Time": 60,
              "Actions": {}
            },
            "EnterXTimes": {

```

```
        "Active": true,
        "Times": 2,
        "Interval": 60,
        "Actions": {}
    },
    "Occupancy": [
        {
            "Name": "High",
            "Description": "High occupancy",
            "Occupancy": 200,
            "Operator": "GREATER_THAN_OR_EQUAL_TO",
            "Actions": {},
            "NormalizationEvent": true,
            "NormalizationEventActions": {}
        }
    ]
},
"Active": true
}
]
}
```

4.14.15.2 Requesting the statistics of LPR Zones

Request the statistics of LPR Zones

Compatibility: Professional, Enterprise

Security level: Requires user authentication with rights to access the desired LPR Zones

Method: HTTP GET

Syntax:

```
http://<server_address>/Interface/LPR/Zones/GetStatistics  
[?<argument=value> [&<argument=value>...]] [&<general_argument>...]]
```

Arguments:

Argument	Valid values	Description
LPRZones	APIMasks	Mask to filter the results. Filter the results to include only the data from the specified zones

Example 1: Request the statistics of all LPR Zones with response in XML

`http://192.168.0.1:8601/Interface/LPR/Zones/GetStatistics?ResponseFormat=XML`

Example 2: Request the statistics of all LPR Zones with response in text

`http://192.168.0.1:8601/Interface/LPR/Zones/GetStatistics?ResponseFormat=Text`

Example 3: Request the statistics of LPR Zones starting with A with response in JSON

```
http://192.168.0.1:8601/Interface/LPR/Zones/GetStatistics?
LPRZones=A*&ResponseFormat=JSON
```

Response:

A list with the statistics data of the LPR Zones is returned

HTTP Return: 200 OK

Parameters of return:**Fixed Parameter:**

Parameter	Type	Description
COUNT	Integer	Total number of LPR Zones (Not included in JSON response)

Parameters in the list of LPR Zones:

Parameter	Type	Description
DUID	DUID	DUID of the LPR Zone
NAME	String	Name of the LPR Zone
VEHICLECOUNT	Integer	Number of vehicles inside the LPR Zone
TODAYENTRIES	Integer	Number of zone entries today
TODAYEXITS	Integer	Number of zone exits today
AVAILABLESPACES	Integer	Number of available parking spaces in the zone
FREESPACES	Integer	Number of free parking spaces in the zone (Available - Count)
OCCUPATIONAVERAGESECS	Integer	Average occupation time (In seconds)

Example of return in text:

```
RESPONSE_CODE=0
RESPONSE_MESSAGE=OK
COUNT=1
LPRZONE_1_DUID=1FF92E39-2FD2-4BCE-9047-BC250D969E37
LPRZONE_1_NAME=LPR Zone 1
LPRZONE_1_VEHICLECOUNT=3
LPRZONE_1_TODAYENTRIES=20
LPRZONE_1_TODAYEXITS=17
LPRZONE_1_AVAILABLESPACES=180
LPRZONE_1_FREESPACES=177
LPRZONE_1_OCCUPATIONAVERAGESECS=3600
```

Example of return in XML:

```

<Response>
  <Code>0</Code>
  <Message>OK</Message>
  <Data>
    <LPRZones>
      <Count>1</Count>
      <LPRZone>
        <DUID>1FF92E39-2FD2-4BCE-9047-BC250D969E37</DUID>
        <Name>LPR Zone 1</Name>
        <VehicleCount>3</VehicleCount>
        <TodayEntries>20</TodayEntries>
        <TodayExits>17</TodayExits>
        <AvailableSpaces>180</AvailableSpaces>
        <FreeSpaces>177</FreeSpaces>
        <OccupationAverageSecs>3600</OccupationAverageSecs>
      </LPRZone>
    </LPRZones>
  </Data>
</Response>

```

Example of return in JSON:

```
{
  "Response": {
    "Code": 0,
    "Message": "OK",
    "Data": {
      "LPRZones": [
        {
          "DUID": "1FF92E39-2FD2-4BCE-9047-BC250D969E37",
          "Name": "LPR Zone 1",
          "VehicleCount": 3,
          "TodayEntries": 20,
          "TodayExits": 17,
          "AvailableSpaces": 180,
          "FreeSpaces": 177,
          "OccupationAverageSecs": 3600
        }
      ]
    }
  }
}
```

4.14.15.3 Seaching LPR Zone Records

Perform a search for LPR Zone records in the database

Compatibility: Professional, Enterprise

Security level: Requires user authentication with rights to search for LPR records

Method: HTTP GET

Syntax:

```
http://<server_address>/Interface/LPR/Zones/Search
[?<argument=value> [&<argument=value>... ] [&<general argument>... ] ]
```

Arguments:

Argument	Valid values	Description					
InDateFrom	APIDate	Search for records starting with the specified entrance date					
InTimeFrom	APITime	Search for records starting with the specified entrance time					
InDateTo	APIDate	End date of filter. Search for all records inside the range of start and end entrance date. If this parameter is omitted, the value specified in <code>InDateFrom</code> parameter will be used					
InTimeTo	APITime	End time of filter. Search for all records inside the range of start and end entrance time If this parameter is omitted, the value specified in <code>InTimeFrom</code> parameter will be used					
OutDateFrom	APIDate	Search for records starting with the specified exit date					
OutTimeFrom	APITime	Search for records starting with the specified exit time					
OutDateTo	APIDate	End date of filter. Search for all records inside the range of start and end exit date. If this parameter is omitted, the value specified in <code>OutDateFrom</code> parameter will be used					
OutTimeTo	APITime	End time of filter. Search for all records inside the range of start and end exit time If this parameter is omitted, the value specified in <code>OutTimeFrom</code> parameter will be used					
ShowOnlyPlatesInZones	Boolean	Search only for plates that are still inside the zone					
Zones	String	List of Zones to filter. The list with license plates must be comma-separated. Zones can be either Name or DUID					
ZoneGroups	String	List of Zone Groups to filter. The list with license plates must be comma-separated. Zones Groups can be either Name or DUID					
LicensePlates	String	List of license plates filter. The list with license plates must be comma-separated.					
OrderBy	String	Sort the records by the specified column. <table border="1" style="margin-left: auto; margin-right: auto;"> <thead> <tr> <th>Columns</th> </tr> </thead> <tbody> <tr> <td>Dateln</td> </tr> <tr> <td>DateOut</td> </tr> <tr> <td>LicensePlate</td> </tr> <tr> <td>LengthOfStay</td> </tr> </tbody> </table> If this parameter is omitted, the records will be displayed in retrieval order (Faster method with less server memory)	Columns	Dateln	DateOut	LicensePlate	LengthOfStay
Columns							
Dateln							
DateOut							
LicensePlate							
LengthOfStay							

		usage)
--	--	--------

Example 1: Search for all records with entrance date between January 01, 2022 and February 01, 2022
 http://192.168.0.1:8601/Interface/LPR/ZonesSearch?InDateFrom=2022.01.01&InTimeFrom=00.00.00.000&InDateTo=2022.02.01&InTimeTo=23.59.59.999

Example 2: Search for all records of plates ABC0001 and DEF1234

http://192.168.0.1:8601/Interface/LPR/Zones/Search?LicensePlates=ABC0001
 DEF1234

Response:

A list with all found LPR Zone records is returned.

HTTP Return: 200 OK

Parameters of return:

Fixed Parameter:

Parameter	Type	Description
COUNT	Integer	Total of records (Not included in JSON response)

Parameters of records list:

LPRZoneRecord	Type	Description
DATETIMEIN	APITimestamp	Date and time of the entrance in timestamp format
DATETIMEOUT	APITimestamp	Date and time of the exit in timestamp format. * This field will only be included in the result if there is an exit
ZONEUID	DUID	DUID of the LPR Zone
ZONE	String	Zone name * This field will only be included in the result if the zone is still registered in the server
LICENSEPLATE	String	License plate
LENGTHOFSTAYDAYS	Integer	Length of stay in days * This field will only be included in the result if there is an exit
LENGTHOFSTAYTIME	APITime	Length of stay in hours * This field will only be included in the result if there is an exit
LPRRECORDNUMBERIN	Integer	LPR Record number of the entrance
LPRRECORDNUMBEROUT	Integer	LPR Record number of the exit * This field will only be included in the result if there is an exit
LPRCONFIGURATIONIN	String	LPR Configuration of the entrance
LPRCONFIGURATIONOUT	String	LPR Configuration of the exit * This field will only be included in the result if there is an exit

LPRZoneRecord	Type	Description
CAMERAINDUID	String	DUID of the entrance camera
CAMERAIN	String	Name of the entrance camera * This field will only be included in the result if the camera is still registered in the server
CAMERAOUTDUID	String	DUID of the exit camera
CAMERAOUT	String	Name of the exit camera * This field will only be included in the result if the camera is still registered in the server

Example of return in text:

```

RESPONSE_CODE=0
RESPONSE_MESSAGE=OK
COUNT=1
LPRZONERECORD_1_DATETIMEIN=2022-12-04 00:13:22.499
LPRZONERECORD_1_DATETIMEOUT=2022-12-04 00:13:22.499
LPRZONERECORD_1_ZONEDUID=1FF92E39-2FD2-4BCE-9047-BC250D969E37
LPRZONERECORD_1_ZONE=Controle de Zonas Digifort
LPRZONERECORD_1_LICENSEPLATE=FMZ0E49
LPRZONERECORD_1_LENGTHOFSTAYDAYS=0
LPRZONERECORD_1_LENGTHOFSTAYTIME=30:00:00
LPRZONERECORD_1_LPRRECORDNUMBERIN=58597
LPRZONERECORD_1_LPRRECORDNUMBEROUT=58597
LPRZONERECORD_1_LPRCONFIGURATIONIN=LPR_Rua2
LPRZONERECORD_1_LPRCONFIGURATIONOUT=LPR_Rua2
LPRZONERECORD_1_CAMERAINDUID=51804AFE-ADA0-4691-BC77-754EDB1A940F
LPRZONERECORD_1_CAMERAIN=40-2
LPRZONERECORD_1_CAMERAOUTDUID=51804AFE-ADA0-4691-BC77-754EDB1A940F
LPRZONERECORD_1_CAMERAOUT=40-2

```

Example of return in XML:

```
<Response>
  <Code>0</Code>
  <Message>OK</Message>
  <Data>
    <LPRZoneRecords>
      <Count>1</Count>
      <LPRZoneRecord>
        <DateTimeIn>2022-12-04 00:13:22.499</DateTimeIn>
        <DateTimeOut>2022-12-04 00:13:22.499</DateTimeOut>
        <ZoneDUID>1FF92E39-2FD2-4BCE-9047-BC250D969E37</ZoneDUID>
        <Zone>Controle de Zonas Digifort</Zone>
        <LicensePlate>FMZ0E49</LicensePlate>
        <LengthOfStayDays>0</LengthOfStayDays>
        <LengthOfStayTime>30:00:00</LengthOfStayTime>
        <LPRRecordNumberIn>58597</LPRRecordNumberIn>
        <LPRRecordNumberOut>58597</LPRRecordNumberOut>
        <LPRConfigurationIn>LPR_Rua2</LPRConfigurationIn>
        <LPRConfigurationOut>LPR_Rua2</LPRConfigurationOut>
        <CameraInDUID>51804AFE-ADA0-4691-BC77-754EDB1A940F</CameraInDUID>
        <CameraIn>40-2</CameraIn>
        <CameraOUTDUID>51804AFE-ADA0-4691-BC77-754EDB1A940F</CameraOUTDUID>
        <CameraOut>40-2</CameraOut>
      </LPRZoneRecord>
    </LPRZoneRecords>
  </Data>
</Response>
```

Example of return in JSON:

```
{
  "Response": {
    "Code": 0,
    "Message": "OK",
    "Data": {
      "LPRZoneRecords": [
        {
          "DateTimeIn": "2022-12-04T00:13:22.499Z",
          "DateTimeOut": "2022-12-04T00:13:22.499Z",
          "ZoneDUID": "1FF92E39-2FD2-4BCE-9047-BC250D969E37",
          "Zone": "Controle de Zonas Digifort",
          "LicensePlate": "FMZ0E49",
          "LengthOfStayDays": 0,
          "LengthOfStayTime": "30:00:00",
          "LPRRecordNumberIn": 58597,
          "LPRRecordNumberOut": 58597,
          "LPRConfigurationIn": "LPR_Rua2",
          "LPRConfigurationOut": "LPR_Rua2",
          "CameraInDUID": "51804AFE-ADA0-4691-BC77-754EDB1A940F",
          "CameraIn": "40-2",
          "CameraOUTDUID": "51804AFE-ADA0-4691-BC77-754EDB1A940F",
          "CameraOut": "40-2"
        }
      ]
    }
  }
}
```

4.14.15.4 Activating / Deactivating LPR Zones

Allows the activation or deactivation of multiple LPR Zones simultaneously

Compatibility: Professional, Enterprise

Security level: Requires user authentication with rights to LPR Configuration Registration

Method: HTTP GET

Syntax:

```
http://<server_address>/Interface/LPR/Zones/Activation?<argument=value>
[&<argument=value>...] [&<general argument>...]
```

Arguments:

Argument	Valid values	Description
LPRZones*	String	<p>List of LPR Zones to activate or deactivate.</p> <p>The zone names must be separated by commas.</p> <p>This command is only mandatory when Action=Activate or Action=Deactivate</p>
Action*	Activate Deactivate ActivateAll	Action to be executed

Operation	Description
-----------	-------------

	DeactivateAll	Activate	Activate the configurations on LPRZones argument	
		Deactivate	Deactivate the configurations on LPRZones argument	
		ActivateAll	Activate all LPR Zones from the server	
		DeactivateAll	Deactivate all LPR Zones from the server	

* Mandatory parameters

Example 1: Activate all LPR Zones from the server

```
http://192.168.0.1:8601/Interface/LPR/Zones/Activation?Action=ActivateAll
```

Example 2: Activate LPR Zones Zone1 and Zone2

```
http://192.168.0.1:8601/Interface/LPR/Zones/Activation?Action=Activate&LPRZones=Zone1,Zone2
```

Example 3: Deactivate all LPR Zones from the server

```
http://192.168.0.1:8601/Interface/LPR/Zones/Activation?Action=DeactivateAll
```

Response:

Default response of API.

HTTP Return HTTP: 200 OK

Parameters of return: [Default return of API](#)

4.14.15.5 Zone Groups

4.14.15.5.1 Requesting the list of LPR Zones Groups

Request the list of LPR Zone Groups over which the user has rights.

Compatibility: Professional, Enterprise

Security level: Requires user authentication

Method: HTTP GET

Syntax:

```
http://<server_address>/Interface/LPR/ZoneGroups/GetLPRZoneGroups  
[?<argument=value> [&<argument=value>...]] [&<general\_argument>...]]
```

Arguments:

Argument	Valid values	Description
LPRZoneGroups	APIMasks	Mask to filter the results. Specify which LPR Zone Groups must be returned based on the provided masks.
Fields	DUID Name Description Active Zones	Specifies the list of desired fields. In case this parameter is omitted, all of the fields will be sent. The fields must be separated by commas

	WarningLevels Events	Name	Description
DUID	DUID of the LPR Zone Group		
Name	Name of the LPR Zone Group		
Description	Description of the LPR Zone Group		
Active	LPR Zone Group is activated or deactivated		
Zones	Comma-separated list of Zone DUIDs that are part of this group		
WarningLevels	Warning levels		
Events	Configured events and actions		

Example 1: Request the list of LPR Zone Groups with all fields and response in XML

```
http://192.168.0.1:8601/Interface/LPR/ZoneGroups/GetLPRZoneGroups?  
ResponseFormat=XML
```

Example 2: Request the list of LPR Zone Groups with all fields and response in text

```
http://192.168.0.1:8601/Interface/LPR/ZoneGroups/GetLPRZoneGroups?  
ResponseFormat=Text
```

Example 3: Request the list of LPR Zone Groups with only name and description, response in XML and authentication of the user Admin

```
http://192.168.0.1:8601/Interface/LPR/ZoneGroups/GetLPRZoneGroups?  
Fields=Name,Description&ResponseFormat=XML&AuthUser=admin
```

Example 4: Request the list of LPR Zone Groups starting with "A", with only name and description, response in XML and authentication with Admin user

```
http://192.168.0.1:8601/Interface/LPR/ZoneGroups/GetLPRZoneGroups?  
LPRZoneGroups=A*&Fields=Name,Description&ResponseFormat=XML&  
AuthUser=admin
```

Response:

A list of all of the LPR Zone Groups registered in the system is returned. The fields returned in the will depend on the values informed in the argument Fields

HTTP Return: 200 OK

Parameters of return:

Fixed Parameter:

Parameter	Type	Description
COUNT	Integer	Total number of LPR Zone Groups (Not included in JSON response)

Parameters in the list of LPR Zone Group:

Parameter	Type	Description
DUID	DUID	DUID of the LPR Zone Group
NAME	String	Name of the LPR Zone Group
DESCRIPTION	String	Description of the LPR Zone Group
ACTIVE	Boolean	Zone Group activated / deactivated
ZONES	String	Comma-separated list of LPR Zone DUIDs that are part of the group

WarningLevels	Type	Description
---------------	------	-------------

Parameter	Type	Description
ACTIVE	Boolean	Warning levels active
AVAILABLESPACES	Integer	Number of available spaces for warning levels
MEDIUMLEVEL	Integer	Percentage of occupied spaces for Medium warning
HIGHLEVEL	Integer	Percentage of occupied spaces for High warning
Events		
DwellTime	Type	Description
ACTIVE	Boolean	Event active
TIME	Integer	Configured time (In minutes)
ACTIONS	Event Actions	List of configured Event Actions. Please refer to Event Actions section for the format of the list.
LeaveBeforeTime		
ACTIVE	Boolean	Event active
TIME	Integer	Configured time (In minutes)
ACTIONS	Event Actions	List of configured Event Actions. Please refer to Event Actions section for the format of the list.
LeaveAfterTime		
ACTIVE	Boolean	Event active
TIME	Integer	Configured time (In minutes)
ACTIONS	Event Actions	List of configured Event Actions. Please refer to Event Actions section for the format of the list.
EnterXTimes		
ACTIVE	Boolean	Event active
TIMES	Integer	Number of times to enter the zone
INTERVAL	Integer	Maximum interval (In minutes)
ACTIONS	Event Actions	List of configured Event Actions. Please refer to Event Actions section for the format of the list.
Occupancy		
COUNT	Integer	Number of occupancy events (Not included in JSON response)
Parameters of each occupancy event		
Event	Type	Description
NAME	String	Event name
DESCRIPTION	String	Event description
OCCUPANCY	Integer	Occupancy threshold
OPERATOR	EQUAL_TO NOT_EQUAL_TO LESS_THAN LESS_THAN_OR_EQUAL_TO	Event operator type

Parameter	Type	Description
	GREATER_THAN GREATER THAN OR EQUAL TO	
ACTIONS	Event Actions	List of configured Event Actions. Please refer to Event Actions section for the format of the list.
NORMALIZATIONEVENT	Boolean	Indicate if Normalization Event is active
NORMALIZATIONEVENTACTION	Event Actions	List of configured Event Actions. Please refer to Event Actions section for the format of the list.

Example of return in text:

```

RESPONSE_CODE=0
RESPONSE_MESSAGE=OK
COUNT=1
LPRZONEGROUP_1_DUID=ED5646FD-37FF-4A26-B135-A77ED22AF287
LPRZONEGROUP_1_NAME=Group 1
LPRZONEGROUP_1_DESCRIPTION=
LPRZONEGROUP_1_ACTIVE=TRUE
LPRZONEGROUP_1_ZONES=EF85B01F-6237-4A32-B7DF-CC2DF6322DE9
LPRZONEGROUP_1_WARNTIMELEVELS_ACTIVE=TRUE
LPRZONEGROUP_1_WARNTIMELEVELS_AVAILABLESPACES=200
LPRZONEGROUP_1_WARNTIMELEVELS_MEDIUMLEVEL=70
LPRZONEGROUP_1_WARNTIMELEVELS_HIGHLIMIT=90
LPRZONEGROUP_1_EVENTS_DWELLTIME_ACTIVE=FALSE
LPRZONEGROUP_1_EVENTS_DWELLTIME_TIME=60
LPRZONEGROUP_1_EVENTS_LEAVEBEFORETIME_ACTIVE=FALSE
LPRZONEGROUP_1_EVENTS_LEAVEBEFORETIME_TIME=60
LPRZONEGROUP_1_EVENTS_LEAVEAFTERTIME_ACTIVE=FALSE
LPRZONEGROUP_1_EVENTS_LEAVEAFTERTIME_TIME=60
LPRZONEGROUP_1_EVENTS_ENTERXTIMES_ACTIVE=FALSE
LPRZONEGROUP_1_EVENTS_ENTERXTIMES_TIMES=2
LPRZONEGROUP_1_EVENTS_ENTERXTIMES_INTERVAL=60
LPRZONEGROUP_1_EVENTS_OCCUPANCY_COUNT=1
LPRZONEGROUP_1_EVENTS_OCCUPANCY_EVENT_1_NAME=High Occupancy
LPRZONEGROUP_1_EVENTS_OCCUPANCY_EVENT_1_DESCRIPTION=
LPRZONEGROUP_1_EVENTS_OCCUPANCY_EVENT_1_OCCUPANCY=150
LPRZONEGROUP_1_EVENTS_OCCUPANCY_EVENT_1_OPERATOR=GREATER_THAN_OR_EQUAL_TO
LPRZONEGROUP_1_EVENTS_OCCUPANCY_EVENT_1_NORMALIZATIONEVENT=TRUE

```

Example of return in XML:

```

<Response>
  <Code>0</Code>
  <Message>OK</Message>
  <Data>
    <LPRZoneGroups>
```

```
<Count>1</Count>
<LPRZoneGroup>
  <DUID>ED5646FD-37FF-4A26-B135-A77ED22AF287</DUID>
  <Name>Group 1</Name>
  <Description />
  <Active>TRUE</Active>
  <Zones>EF85B01F-6237-4A32-B7DF-CC2DF6322DE9</Zones>
  <WarningLevels>
    <Active>TRUE</Active>
    <AvailableSpaces>200</AvailableSpaces>
    <MediumLevel>70</MediumLevel>
    <HighLevel>90</HighLevel>
  </WarningLevels>
  <Events>
    <DwellTime>
      <Active>FALSE</Active>
      <Time>60</Time>
      <Actions />
    </DwellTime>
    <LeaveBeforeTime>
      <Active>FALSE</Active>
      <Time>60</Time>
      <Actions />
    </LeaveBeforeTime>
    <LeaveAfterTime>
      <Active>FALSE</Active>
      <Time>60</Time>
      <Actions />
    </LeaveAfterTime>
    <EnterXTimes>
      <Active>FALSE</Active>
      <Times>2</Times>
      <Interval>60</Interval>
      <Actions />
    </EnterXTimes>
    <Occupancy>
      <Count>1</Count>
      <Event>
        <Name>High Occupancy</Name>
        <Description />
        <Occupancy>150</Occupancy>
        <Operator>GREATER_THAN_OR_EQUAL_TO</Operator>
        <Actions />
        <NormalizationEvent>TRUE</NormalizationEvent>
        <NormalizationEventActions />
      </Event>
    </Occupancy>
  </Events>
</LPRZoneGroup>
</LPRZoneGroups>
</Data>
</Response>
```

Example of return in JSON:

```
{
  "Response": {
    "Code": 0,
    "Message": "OK",
    "Data": {
      "LPRZoneGroups": [
        {
          "DUID": "ED5646FD-37FF-4A26-B135-A77ED22AF287",
          "Name": "Group 1",
          "Description": "",
          "Active": true,
          "Zones": "EF85B01F-6237-4A32-B7DF-CC2DF6322DE9",
          "WarningLevels": {
            "Active": true,
            "AvailableSpaces": 200,
            "MediumLevel": 70,
            "HighLevel": 90
          },
          "Events": {
            "DwellTime": {
              "Active": false,
              "Time": 60,
              "Actions": {}
            },
            "LeaveBeforeTime": {
              "Active": false,
              "Time": 60,
              "Actions": {}
            },
            "LeaveAfterTime": {
              "Active": false,
              "Time": 60,
              "Actions": {}
            },
            "EnterXTimes": {
              "Active": false,
              "Times": 2,
              "Interval": 60,
              "Actions": {}
            },
            "Occupancy": [
              {
                "Name": "High Occupancy",
                "Description": "",
                "Occupancy": 150,
                "Operator": "GREATER_THAN_OR_EQUAL_TO",
                "Actions": {},
                "NormalizationEvent": true,
                "NormalizationEventActions": {}
              }
            ]
          }
        }
      ]
    }
  }
}
```

```

        }
    ]
}
}
}
```

4.14.15.5.2 Requesting the statistics of LPR Zone Groups

Request the statistics of LPR Zone Groups

Compatibility: Professional, Enterprise

Security level: Requires user authentication with rights to access the desired LPR Zone Groups

Method: HTTP GET

Syntax:

```
http://<server_address>/Interface/LPR/ZoneGroups/GetStatistics
[?<argument=value>[&<argument=value>...][&<general\_argument>...]]
```

Arguments:

Argument	Valid values	Description
LPRZoneGroups	APIMasks	Mask to filter the results. Filter the results to include only the data from the specified zone groups

Example 1: Request the statistics of all LPR Zone Groups with response in XML

```
http://192.168.0.1:8601/Interface/LPR/ZoneGroups/GetStatistics?
ResponseFormat=XML
```

Example 2: Request the statistics of all LPR Zone Groups with response in text

```
http://192.168.0.1:8601/Interface/LPR/ZoneGroups/GetStatistics?
ResponseFormat=Text
```

Example 3: Request the statistics of LPR Zone Groups starting with A with response in JSON

```
http://192.168.0.1:8601/Interface/LPR/ZoneGroups/GetStatistics?
LPRZones=A*&ResponseFormat=JSON
```

Response:

A list with the statistics data of the LPR Zone Groups is returned

HTTP Return: 200 OK

Parameters of return:

Fixed Parameter:

Parameter	Type	Description
COUNT	Integer	Total number of LPR Zone Groups (Not included in JSON response)

Parameters in the list of LPR Zone Groups:

Parameter	Type	Description
DUID	DUID	DUID of the LPR Zone Group
NAME	String	Name of the LPR Zone Group

Parameter	Type	Description
VEHICLECOUNT	Integer	Number of vehicles inside the LPR Zone Group
TODAYENTRIES	Integer	Number of zone group entries today
TODAYEXITS	Integer	Number of zone group exits today
AVAILABLESPACES	Integer	Number of available parking spaces in the zone group
FREESPACES	Integer	Number of free parking spaces in the zone group (Available - Count)
OCCUPATIONAVERAGESECS	Integer	Average occupation time (In seconds)

Example of return in text:

```
RESPONSE_CODE=0
RESPONSE_MESSAGE=OK
COUNT=1
LPRZONEGROUP_1_DUID=1FF92E39-2FD2-4BCE-9047-BC250D969E37
LPRZONEGROUP_1_NAME=LPR Zone Group 1
LPRZONEGROUP_1_VEHICLECOUNT=3
LPRZONEGROUP_1_TODAYENTRIES=20
LPRZONEGROUP_1_TODAYEXITS=17
LPRZONEGROUP_1_AVAILABLESPACES=180
LPRZONEGROUP_1_FREESPACES=177
LPRZONEGROUP_1_OCCUPATIONAVERAGESECS=3600
```

Example of return in XML:

```
<Response>
  <Code>0</Code>
  <Message>OK</Message>
  <Data>
    <LPRZoneGroups>
      <Count>1</Count>
      <LPRZoneGroup>
        <DUID>1FF92E39-2FD2-4BCE-9047-BC250D969E37</DUID>
        <Name>LPR Zone Group 1</Name>
        <VehicleCount>3</VehicleCount>
        <TodayEntries>20</TodayEntries>
        <TodayExits>17</TodayExits>
        <AvailableSpaces>180</AvailableSpaces>
        <FreeSpaces>177</FreeSpaces>
        <OccupationAverageSecs>3600</OccupationAverageSecs>
      </LPRZoneGroup>
    </LPRZoneGroups>
  </Data>
</Response>
```

Example of return in JSON:

```
{
  "Response": {
    "Code": 0,
    "Message": "OK",
    "Data": {
      "LPRZoneGroups": [
        {
          "DUID": "1FF92E39-2FD2-4BCE-9047-BC250D969E37",
          "Name": "LPR Zone Group 1",
          "VehicleCount": 3,
          "TodayEntries": 20,
          "TodayExits": 17,
          "AvailableSpaces": 180,
          "FreeSpaces": 177,
          "OccupationAverageSecs": 3600
        }
      ]
    }
  }
}
```

4.14.15.5.3 Activating / Deactivating LPR Zone Groups

Allows the activation or deactivation of multiple LPR Zone Groups simultaneously

Compatibility: Professional, Enterprise

Security level: Requires user authentication with rights to LPR Configuration Registration

Method: HTTP GET

Syntax:

```
http://<server_address>/Interface/LPR/ZoneGroups/Activation?  
<argument=value>[&<argument=value>...][&<general argument>...]
```

Arguments:

Argument	Valid values	Description										
LPRZoneGroups*	String	<p>List of LPR Zone Groups to activate or deactivate.</p> <p>The zone group names must be separated by commas.</p> <p>This command is only mandatory when Action=Activate or Action=Deactivate</p>										
Action*	Activate Deactivate ActivateAll DeactivateAll	<p>Action to be executed</p> <table border="1"> <thead> <tr> <th>Operation</th><th>Description</th></tr> </thead> <tbody> <tr> <td>Activate</td><td>Activate the configurations on LPRZoneGroups argument</td></tr> <tr> <td>Deactivate</td><td>Deactivate the configurations on LPRZoneGroups argument</td></tr> <tr> <td>ActivateAll</td><td>Activate all LPR Zone Groups from the server</td></tr> <tr> <td>DeactivateAll</td><td>Deactivate all LPR Zone Groups</td></tr> </tbody> </table>	Operation	Description	Activate	Activate the configurations on LPRZoneGroups argument	Deactivate	Deactivate the configurations on LPRZoneGroups argument	ActivateAll	Activate all LPR Zone Groups from the server	DeactivateAll	Deactivate all LPR Zone Groups
Operation	Description											
Activate	Activate the configurations on LPRZoneGroups argument											
Deactivate	Deactivate the configurations on LPRZoneGroups argument											
ActivateAll	Activate all LPR Zone Groups from the server											
DeactivateAll	Deactivate all LPR Zone Groups											

		from the server
--	--	-----------------

* Mandatory parameters

Example 1: Activate all LPR Zone Groups from the server

```
http://192.168.0.1:8601/Interface/LPR/ZoneGroups/Activation?
Action=ActivateAll
```

Example 2: Activate LPR Zone Groups ZoneGroup1 and ZoneGroup2

```
http://192.168.0.1:8601/Interface/LPR/ZoneGroups/Activation?
Action=Activate&LPRZoneGroups=ZoneGroup1,ZoneGroup2
```

Example 3: Deactivate all LPR Zone Groups from the server

```
http://192.168.0.1:8601/Interface/LPR/ZoneGroups/Activation?
Action=DeactivateAll
```

Response:

Default response of API.

HTTP Return HTTP: 200 OK

Parameters of return: [Default return of API](#)

4.15 Analytics

4.15.1 Requesting the list of Analytics Configurations

Request the list of Analytics Configurations over which the user has rights.

Compatibility: Standard, Professional, Enterprise

Security level: Requires user authentication

Method: HTTP GET

Syntax:

```
http://<server_address>/Interface/Analytics/GetAnalyticsConfigurations
[?<argument=value>[&<argument=value>...][&<general\_argument>...]]
```

Arguments:

Argument	Valid values	Description						
AnalyticsConfigurations	APIMasks	Mask to filter the results. Specify which Analytics Configurations must be returned based on the provided masks.						
Fields	Name Description Active Camera Events	Specifies the list of desired fields. In case this parameter is omitted, all of the fields will be sent. The fields must be separated by commas <table border="1" style="margin-left: 20px;"> <thead> <tr> <th>Name</th><th>Description</th></tr> </thead> <tbody> <tr> <td>Name</td><td>Name of the Analytics Configuration</td></tr> <tr> <td>Description</td><td>Description of the Analytics</td></tr> </tbody> </table>	Name	Description	Name	Name of the Analytics Configuration	Description	Description of the Analytics
Name	Description							
Name	Name of the Analytics Configuration							
Description	Description of the Analytics							

		Configuration
Active		Analytics Configuration activate or deactivated
Camera		Associated camera with the Analytics Configuration
Events		List of configured analytics events

Example 1: Request the list of Analytics Configurations with all fields and response in XML

```
http://192.168.0.1:8601/Interface/Analytics/GetAnalyticsConfigurations?
ResponseFormat=XML
```

Example 2: Request the list of Analytics Configurations with all fields and response in text

```
http://192.168.0.1:8601/Interface/Analytics/GetAnalyticsConfigurations?
ResponseFormat=Text
```

Example 3: Request the list of Analytics Configurations with only name and description, response in XML and authentication of the user Admin

```
http://192.168.0.1:8601/Interface/Analytics/GetAnalyticsConfigurations?
Fields=Name,Description&ResponseFormat=XML&AuthUser=admin
```

Example 4: Request the list of Analytics Configurations starting with "A", with only name and description, response in XML and authentication with Admin user

```
http://192.168.0.1:8601/Interface/Analytics/GetAnalyticsConfigurations?
AnalyticsConfigurations=A*&Fields=Name,Description&ResponseFormat=XML&
AuthUser=admin
```

Response:

A list of all of the Analytics Configurations registered in the system is returned. The fields returned in the will depend on the values informed in the argument **Fields**

HTTP Return: 200 OK

Parameters of return:

Fixed Parameter:

Parameter	Type	Description
COUNT	Integer	Total number of Analytics Configurations (Not included in JSON response)

Parameters in the list of Analytics Configurations:

Parameter	Type	Description
NAME	String	Name of the Analytics Configuration
DESCRIPTION	String	Description of the Analytics Configuration
ACTIVE	Boolean	Analytics Configuration is activated or deactivated
CAMERA	String	Name of the camera associated to the Analytics Configuration
ANALYTICSEVENTS	List	List of analytics events as defined below

Parameters of return for AnalyticsEvents list:

Parameter	Type	Description
COUNT	Integer	Number of events
AnalyticsEvent		
DUID	DUID	DUID of the event
NAME	String	Name of the event
TYPE	Event Type	Type of event
ID	String	Internal event ID

Example of return in text:

```

RESPONSE_CODE=0
RESPONSE_MESSAGE=OK
COUNT=2
ANALYTICSConfiguration_1_NAME=Configuration1
ANALYTICSConfiguration_1_DESCRIPTION=My Configuration
ANALYTICSConfiguration_1_ACTIVE=TRUE
ANALYTICSConfiguration_1_CAMERA=Camera1
ANALYTICSConfiguration_1_ANALYTICSEVENTS_COUNT=2
ANALYTICSConfiguration_1_ANALYTICSEVENTS_ANALYTICSEVENT_1_DUID=00000000-0000
ANALYTICSConfiguration_1_ANALYTICSEVENTS_ANALYTICSEVENT_1_NAME=WrongWay
ANALYTICSConfiguration_1_ANALYTICSEVENTS_ANALYTICSEVENT_1_TYPE=ANALYTICS_DIRECTION
ANALYTICSConfiguration_1_ANALYTICSEVENTS_ANALYTICSEVENT_1_ID=4-EVENT_ID
ANALYTICSConfiguration_1_ANALYTICSEVENTS_ANALYTICSEVENT_2_DUID=00000000-0000
ANALYTICSConfiguration_1_ANALYTICSEVENTS_ANALYTICSEVENT_2_NAME=ProhibitedArea
ANALYTICSConfiguration_1_ANALYTICSEVENTS_ANALYTICSEVENT_2_TYPE=ANALYTICS_ENTITY
ANALYTICSConfiguration_1_ANALYTICSEVENTS_ANALYTICSEVENT_2_ID=4-EVENT_ID
ANALYTICSConfiguration_2_NAME=Configuration2
ANALYTICSConfiguration_2_DESCRIPTION=My Configuration
ANALYTICSConfiguration_2_ACTIVE=FALSE
ANALYTICSConfiguration_2_CAMERA=Camera2
ANALYTICSConfiguration_2_ANALYTICSEVENTS_COUNT=0

```

Example of return in XML:

```

<Response>
  <Code>0</Code>
  <Message>OK</Message>
  <Data>
    <AnalyticsConfigurations>
      <Count>2</Count>
      <AnalyticsConfiguration>
        <Name>Configuration1</Name>
        <Description>My Configuration</Description>
        <Active>TRUE</Active>
        <Camera>Camera1</Camera>
        <AnalyticsEvents>
          <Count>2</Count>
          <AnalyticsEvent>
            <DUID>00000000-0000-0000-000000000000</DUID>
            <Name>WrongWay</Name>
            <Type>ANALYTICS_DIRECTION</Type>
            <ID>EVENT_ID</ID>
          </AnalyticsEvent>
        </AnalyticsEvents>
      </AnalyticsConfiguration>
    </AnalyticsConfigurations>
  </Data>
</Response>

```

```

</AnalyticsEvent>
<AnalyticsEvent>
<DUID>00000000-0000-0000-0000-000000000000</DUID>
<Name>ProhibitedArea</Name>
<Type>ANALYTICS_ENTER</Type>
<ID>EVENT_ID</ID>
</AnalyticsEvent>
</AnalyticsEvents>
</AnalyticsConfiguration>
<AnalyticsConfiguration>
<Name>Configuration2</Name>
<Description>My Configuration</Description>
<Active>FALSE</Active>
<Camera>Camera2</Camera>
<AnalyticsEvents>
<Count>0</Count>
</AnalyticsEvents>
</AnalyticsConfiguration>
</AnalyticsConfigurations>
</Data>
</Response>

```

Example of return in JSON:

```

{
  "Response": {
    "Code": 0,
    "Message": "OK",
    "Data": {
      "AnalyticsConfigurations": [
        {
          "Name": "Configuration1",
          "Description": "My Configuration",
          "Active": true,
          "Camera": "Cameral"
          "AnalyticsEvents": [
            {
              "DUID": "00000000-0000-0000-0000-000000000000",
              "Name": "WrongWay",
              "Type": "ANALYTICS_DIRECTION",
              "ID": "EVENT_ID"
            },
            {
              "DUID": "00000000-0000-0000-0000-000000000000",
              "Name": "ProhibitedArea",
              "Type": "ANALYTICS_ENTER",
              "ID": "EVENT_ID"
            }
          ]
        },
        {
          "Name": "Configuration1",
          "Description": "My Configuration",
          "Active": false,
        }
      ]
    }
  }
}

```

```

        "Camera": "Camera2"
        "AnalyticsEvents": []
    }
]
}
}
}

```

4.15.2 Requesting the status of Analytics Configurations

Request the status of the Analytics Configurations over which the user has rights.

Compatibility: Standard, Professional, Enterprise

Security level: Requires user authentication

Method: HTTP GET

Syntax:

```
http://<server_address>/Interface/Analytics/GetStatus?<argument=value>
[&<argument=value>...] [&<general\_argument>...]]
```

Arguments:

Argument	Valid values	Description										
AnalyticsConfigurations	APIMasks	Mask to filter the results. Specify which configurations must be returned based on the provided masks.										
Fields	Active Working Status StatusMessage	Specifies the list of desired fields. In case this parameter is omitted, all of the fields will be sent. The fields must be separated by commas <table border="1" data-bbox="861 1241 1437 1537"> <thead> <tr> <th>Name</th><th>Description</th></tr> </thead> <tbody> <tr> <td>Active</td><td>Identify if the configuration is active</td></tr> <tr> <td>Working</td><td>Identify if the configuration is working</td></tr> <tr> <td>Status</td><td>Current status of the configuration</td></tr> <tr> <td>StatusMessage</td><td>Current status message of the configuration</td></tr> </tbody> </table>	Name	Description	Active	Identify if the configuration is active	Working	Identify if the configuration is working	Status	Current status of the configuration	StatusMessage	Current status message of the configuration
Name	Description											
Active	Identify if the configuration is active											
Working	Identify if the configuration is working											
Status	Current status of the configuration											
StatusMessage	Current status message of the configuration											
Active	TRUE FALSE	In case this parameter is specified, a filter will be applied and only the configurations that matches this filter will be returned, thus, in case the value Active=TRUE is specified, the result will only include the active configurations, while if the value Active=FALSE is specified, the result will only include the deactivated configurations.										
Working	TRUE FALSE	In case this parameter is specified, a filter will be applied and only the configurations that matches this filter will be returned, thus, in case the value Working=TRUE is specified, the result will only										

		include the working configurations, while if the value Working=FALSE is specified, the result will only include the configurations that are not working.
--	--	--

Example 1: Request the status of all configurations with all fields and response in XML

```
http://192.168.0.1:8601/Interface/Analytics/GetStatus?ResponseFormat=XML
```

Example 2: Request the status of all active configurations with response in text

```
http://192.168.0.1:8601/Interface/Analytics/GetStatus?Active=TRUE&ResponseFormat=Text
```

Example 3: Request the status of all active configurations starting with A, with response in text

```
http://192.168.0.1:8601/Interface/Analytics/GetStatus?AnalyticsConfigurations=A*&Active=TRUE&ResponseFormat=Text
```

Example 4: Request the status of all active configurations that are not working, with response in XML and authentication with admin user (No password)

```
http://192.168.0.1:8601/Interface/Analytics/GetStatus?Active=TRUE&Working=FALSE&ResponseFormat=XML&AuthUser=admin
```

Response:

A list with the status of all of the configurations over which the user has rights is returned. The fields returned in the will depend on the values informed in the argument Fields

HTTP Return: 200 OK

Parameters of return:**Fixed Parameter:**

Parameter	Type	Description
COUNT	Integer	Total number of configurations (Not included in JSON response)

Parameters in the list of status of configurations:

Parameter	Type	Description												
NAME	String	Name of the configuration												
ACTIVE	Boolean	Identify if the configuration is active												
WORKING	Boolean	Identify if the configuration is working												
STATUS	String	The following table lists the possible values for status: <table border="1"> <thead> <tr> <th>Values</th> </tr> </thead> <tbody> <tr><td>CAMERA_NOT_FOUND</td></tr> <tr><td>CAMERA_DEACTIVATED</td></tr> <tr><td>CAMERA_NOT_WORKING</td></tr> <tr><td>INVALID_MEDIA_PROFILE</td></tr> <tr><td>INVALID_ENGINE_CONFIGURATION_DATA</td></tr> <tr><td>INVALID_ANALYTICS_DRIVER</td></tr> <tr><td>VIDEO_NODE_LOST</td></tr> <tr><td>DISABLED_BY_SCHEDULE</td></tr> <tr><td>NOT_IN_PRESET</td></tr> <tr><td>DRIVER</td></tr> <tr><td>INTERNAL_ERROR</td></tr> </tbody> </table>	Values	CAMERA_NOT_FOUND	CAMERA_DEACTIVATED	CAMERA_NOT_WORKING	INVALID_MEDIA_PROFILE	INVALID_ENGINE_CONFIGURATION_DATA	INVALID_ANALYTICS_DRIVER	VIDEO_NODE_LOST	DISABLED_BY_SCHEDULE	NOT_IN_PRESET	DRIVER	INTERNAL_ERROR
Values														
CAMERA_NOT_FOUND														
CAMERA_DEACTIVATED														
CAMERA_NOT_WORKING														
INVALID_MEDIA_PROFILE														
INVALID_ENGINE_CONFIGURATION_DATA														
INVALID_ANALYTICS_DRIVER														
VIDEO_NODE_LOST														
DISABLED_BY_SCHEDULE														
NOT_IN_PRESET														
DRIVER														
INTERNAL_ERROR														

Parameter	Type	Description
STATUSMESSAGE	String	This parameter will contain a human readable message describing the current status of the configuration

Example of return in text:

```
RESPONSE_CODE=0
RESPONSE_MESSAGE=OK
COUNT=2
ANALYTICSConfiguration_1_NAME=Entrance
ANALYTICSConfiguration_1_ACTIVE=TRUE
ANALYTICSConfiguration_1_WORKING=TRUE
ANALYTICSConfiguration_1_STATUS=DRIVER
ANALYTICSConfiguration_1_STATUSMESSAGE=Data: Processing...
ANALYTICSConfiguration_2_NAME=Exit
ANALYTICSConfiguration_2_ACTIVE=TRUE
ANALYTICSConfiguration_2_WORKING=TRUE
ANALYTICSConfiguration_2_STATUS=DRIVER
ANALYTICSConfiguration_2_STATUSMESSAGE=Data: Processing...
```

Example of return in XML:

```
<Response>
  <Code>0</Code>
  <Message>OK</Message>
  <Data>
    <AnalyticsConfigurations>
      <Count>2</Count>
      <AnalyticsConfiguration>
        <Name>Entrance</Name>
        <Active>TRUE</Active>
        <Working>TRUE</Working>
        <Status>DRIVER</Status>
        <StatusMessage>Data: Processing...</StatusMessage>
      </AnalyticsConfiguration>
      <AnalyticsConfiguration>
        <Name>Exit</Name>
        <Active>TRUE</Active>
        <Working>TRUE</Working>
        <Status>DRIVER</Status>
        <StatusMessage>Data: Processing...</StatusMessage>
      </AnalyticsConfiguration>
    </AnalyticsConfigurations>
  </Data>
</Response>
```

Example of return in JSON:

```
{
  "Response": {
    "Code": 0,
    "Message": "OK",
    "Data": {
      "AnalyticsConfigurations": [
        {
          "Name": "Entrance",
          "Active": true,
          "Working": true,
          "Status": "DRIVER",
          "StatusMessage": "Data: Processing..."
        },
        {
          "Name": "Exit",
          "Active": true,
          "Working": true,
          "Status": "DRIVER",
          "StatusMessage": "Data: Processing..."
        }
      ]
    }
  }
}
```

4.15.3 Activating / Deactivating Analytics configurations

Allows the activation or deactivation of multiple analytics configurations simultaneously

Compatibility: Standard, Professional, Enterprise

Security level: Requires user authentication with rights to configure Analytics Configurations

Method: HTTP GET

Syntax:

```
http://<server_address>/Interface/Analytics/Activation?<argument=value>
[&<argument=value>...] [&<general_argument>...]
```

Arguments:

Argument	Valid values	Description				
AnalyticsConfigurations*	String	<p>List of analytics configurations to activate or deactivate.</p> <p>The configuration names must be separated by commas.</p> <p>This command is only mandatory when Action=Activate or Action=Deactivate</p>				
Action*	Activate Deactivate ActivateAll DeactivateAll	<p>Action to be executed</p> <table border="1" style="margin-left: auto; margin-right: auto;"> <thead> <tr> <th>Operation</th><th>Description</th></tr> </thead> <tbody> <tr> <td>Activate</td><td>Activate the configurations</td></tr> </tbody> </table>	Operation	Description	Activate	Activate the configurations
Operation	Description					
Activate	Activate the configurations					

			on AnalyticsConfigurations argument
		Deactivate	Deactivate the configurations on AnalyticsConfigurations argument
		ActivateAll	Activate all analytics configurations from the server
		DeactivateAll	Deactivate all analytics configurations from the server

* Mandatory parameters

Example 1: Activate all analytics configurations from the server

```
http://192.168.0.1:8601/Interface/Analytics/Activation?Action=ActivateAll
```

Example 2: Activate analytics configurations Analytics1 and Analytics2

```
http://192.168.0.1:8601/Interface/Analytics/Activation?Action=Activate&AnalyticsConfigurations=Analytics1,Analytics2
```

Example 3: Deactivate all analytics configurations from the server

```
http://192.168.0.1:8601/Interface/Analytics/Activation?Action=DeactivateAll
```

Response:

Default response of API.

HTTP Return HTTP: 200 OK

Parameters of return: [Default return of API](#)

4.15.4 Requesting the value of counters in an Analytics Configuration

Request the value of all counters of an Analytics Configurations over which the user has rights.

Compatibility: Standard, Professional, Enterprise

Security level: Requires user authentication

Method: HTTP GET

Syntax:

```
http://<server_address>/Interface/Analytics/GetCounters?<argument=value> [&<argument=value>...] [&<general\_argument>...]
```

Arguments:

Argument	Valid values	Description
AnalyticsConfiguration*	String	Name of the analytics configuration to query the counter values

* Mandatory parameters

Example 1: Request the value of all counters of analytics configuration 01 with response in XML

```
http://192.168.0.1:8601/Interface/Analytics/GetCounters?  
AnalyticsConfiguration=01&ResponseFormat=XML
```

Example 2: Request the value of all counters of analytics configuration 01 with response in text

```
http://192.168.0.1:8601/Interface/Analytics/GetCounters?  
AnalyticsConfiguration=01&ResponseFormat=Text
```

Response:

A list with the value of all counters of the specified analytics configuration is returned.

HTTP Return: 200 OK

Parameters of return:**Fixed Parameter:**

Parameter	Type	Description
COUNT	Integer	Total number of counters (Not included in JSON response)

Parameters in the list of counters:

Parameter	Type	Description
ID	Integer	ID of the counter
NAME	String	Name of the counter
VALUE	Integer	Current value of the counter

Example of return in text:

```
RESPONSE_CODE=0  
RESPONSE_MESSAGE=OK  
COUNT=2  
COUNTER_1_ID=0  
COUNTER_1_NAME=Entrance  
COUNTER_1_VALUE=4322  
COUNTER_2_ID=1  
COUNTER_2_NAME=Exit  
COUNTER_2_VALUE=4563
```

Example of return in XML:

```

<Response>
  <Code>0</Code>
  <Message>OK</Message>
  <Data>
    <Counters>
      <Count>COUNT</Count>
      <Counter>
        <ID>0</ID>
        <Name>Entrance</Name>
        <Value>4322</Value>
      </Counter>
      <Counter>
        <ID>1</ID>
        <Name>Exit</Name>
        <Value>4563</Value>
      </Counter>
    </Counters>
  </Data>
</Response>

```

Example of return in JSON:

```

{
  "Response": {
    "Code": 0,
    "Message": "OK",
    "Data": {
      "Counters": [
        {
          "ID": 0,
          "Name": "Entrance",
          "Value": 4322
        },
        {
          "ID": 1,
          "Name": "Exit",
          "Value": 4563
        }
      ]
    }
  }
}

```

4.15.5 Resetting the value of counters of an Analytics Configuration

This command will reset the value of one or all counters of an Analytics Configuration

Compatibility: Standard, Professional, Enterprise

Security level: Requires user authentication

Method: HTTP GET

Syntax:

```
http://<server_address>/Interface/Analytics/ResetCounter?<argument=value>
[&<argument=value>...] [&<general argument>...]
```

Arguments:

Argument	Valid values	Description
AnalyticsConfiguration*	String	Name of the analytics configuration to reset the counter values
CounterID*	Integer	ID of the counter to reset A value of -1 will reset all counters

* Mandatory parameters

Example 1: Reset the value of counter 2 from analytics configuration 01

```
http://192.168.0.1:8601/Interface/Analytics/ResetCounter?
AnalyticsConfiguration=01&CounterID=2
```

Example 2: Reset the value of all counters from analytics configuration 01

```
http://192.168.0.1:8601/Interface/Analytics/ResetCounter?
AnalyticsConfiguration=01&CounterID=-1
```

Response:

Default response of API.

HTTP Return: 200 OK

Parameters of return: [Default return of API](#)

4.15.6 Searching for Analytics records

Perform a search for Analytics records in the database

Compatibility: Standard, Professional, Enterprise

Security level: Requires user authentication with rights to search for Analytics records

Method: HTTP GET

Syntax:

```
http://<server_address>/Interface/Analytics/Search
[?<argument=value>[&<argument=value>...] [&<general argument>...]]
```

Arguments:

Argument	Valid values	Description
StartDate	APIDate	Search for records starting with the specified date
StartTime	APITime	Search for records starting with the specified time
EndDate	APIDate	End date of search. Search for all records inside the range of start and end. If this parameter is omitted, the value specified in <code>StartDate</code> parameter will be used
EndTime	APITime	End time of search. Search for all records inside the range of start and end.

		If this parameter is omitted, the value specified in StartTime parameter will be used
Cameras	String	List of cameras to search. The list with camera names must be comma-separated
ObjectClasses	String	List of object classes to search. The list of values must be comma-separated
Zones	String	List of zones to search. The list of values must be comma-separated
EventTypes	String	<p>List of event types to search. The list of values must be comma-separated</p> <p>Valid event type values:</p> <ul style="list-style-type: none"> PRESENCE ENTER EXIT APPEAR DISAPPEAR STOPPED LOITERING DIRECTION SPEED TAILGATING COUNTING_LINE_A COUNTING_LINE_B TAMPERING ABANDONED_OBJECT REMOVED_OBJECT SMOKE FIRE FACE_DETECTION OBJECT_CONDITION_CHANGE FOLLOWING_ROUTE SIMILARITY OCCUPANCY VIDEO_SYNOPSIS DETECTION NON_DETECTION SOUND_CLASSIFICATION QUEUE WRONG_DIRECTION ILLEGAL_PARKING ILLEGAL_CONVERSION VEHICLE_DETECTION VEHICLE_OVER_CROSS_LANE LANE_CHANGE LOGICAL_RULE THERMAL FACE_RECOGNITION SCENE_CHANGE COUNTER COUNTING_LINE OBJECT_FILTER COLOR_FILTER

		CONTINUOUSLY PEDESTRIAN_DETECTION ROAD_BLOCKED PREVIOUS DEEP_LEARNING_FILTER SMART_TRACKING SOCIAL_DISTANCING ILLEGAL_VEHICLE_DETECTION TRAFFIC_ACCIDENT IRREGULAR_CONSTRUCTION IRREGULAR_PARKING PARKING_STATE TRAFFIC_CONGESTION									
RuleNames	String	List of rule names to search. The list of values must be comma-separated									
OrderBy	String	<p>Sort the records by the specified column.</p> <table border="1" style="margin-left: auto; margin-right: auto;"> <thead> <tr style="background-color: #0072BD; color: white;"> <th>Columns</th> </tr> </thead> <tbody> <tr><td>RecordCode</td></tr> <tr><td>Camera</td></tr> <tr><td>StartDate</td></tr> <tr><td>EndDate</td></tr> <tr><td>Zone</td></tr> <tr><td>EventType</td></tr> <tr><td>ObjectClass</td></tr> <tr><td>RuleName</td></tr> </tbody> </table> <p>If this parameter is omitted, the records will be displayed in retrieval order (Faster method with less server memory usage)</p>	Columns	RecordCode	Camera	StartDate	EndDate	Zone	EventType	ObjectClass	RuleName
Columns											
RecordCode											
Camera											
StartDate											
EndDate											
Zone											
EventType											
ObjectClass											
RuleName											

Example 1: Search for all records between January 01, 2014 and February 01, 2014

```
http://192.168.0.1:8601/Interface/Analytics/Search?StartDate=2014.01.01&StartTime=00.00.00.000&EndDate=2014.02.01&EndTime=23.59.59.999
```

Example 2: Search for all records of Presence and Direction events, sorting by start date

```
http://192.168.0.1:8601/Interface/Analytics/Search?  
EventTypes=PRESENCE,DIRECTION&OrderBy=StartDate
```

Response:

A list with all found Analytics records is returned.

HTTP Return: 200 OK

Parameters of return:

Fixed Parameter:

Parameter	Type	Description
COUNT	Integer	Total of records (Not included in JSON response)

Parameters of records list:

Parameter	Type	Description
RECORDCODE	Integer	Database record number

Parameter	Type	Description
CAMERA	String	Camera name
STARTDATE	APITimestamp	Record start date and time
ENDDATE	APITimestamp	Record end date and time
ZONE	String	Name of the zone
EVENTTYPE	PRESENCE ENTER EXIT APPEAR DISAPPEAR STOPPED LOITERING DIRECTION SPEED TAILGATING COUNTING_LINE_A COUNTING_LINE_B TAMPERING ABANDONED_OBJECT REMOVED_OBJECT SMOKE FIRE FACE_DETECTION OBJECT_CONDITION_CHANGE FOLLOWING_ROUTE SIMILARITY OCCUPANCY DETECTION NON_DETECTION SOUND_CLASSIFICATION QUEUE WRONG_DIRECTION ILLEGAL_PARKING TURN_ROUND VEHICLE_EXIST CROSS_LANE LANE_CHANGE LOGICAL_RULE THERMAL FACE_RECOGNITION	Type of analytics event
RULENAME	String	Name of the rule
OBJECTCLASS	String	Class of object
METADATAPRESENT	Boolean	Record has extra metadata (See Requesting the metadata of a record)

Example of return in text:

```
RESPONSE_CODE=0
RESPONSE_MESSAGE=OK
COUNT=2
ANALYTICSRECORD_1_RECORDCODE=98370
ANALYTICSRECORD_1_CAMERA=04
ANALYTICSRECORD_1_STARTDATE=2015-09-27 00:12:21.351
ANALYTICSRECORD_1_ENDDATE=2015-09-27 00:12:21.351
ANALYTICSRECORD_1_ZONE=Entrance
ANALYTICSRECORD_1_EVENTTYPE=PRESENCE
ANALYTICSRECORD_1_RULENAME=Presence
ANALYTICSRECORD_1_OBJECTCLASS=Person
ANALYTICSRECORD_1_METADATAPRESENT=FALSE
ANALYTICSRECORD_2_RECORDCODE=98371
ANALYTICSRECORD_2_CAMERA=02
ANALYTICSRECORD_2_STARTDATE=2015-09-27 00:25:40.000
ANALYTICSRECORD_2_ENDDATE=2015-09-27 00:25:40.000
ANALYTICSRECORD_2_ZONE=Exit
ANALYTICSRECORD_2_EVENTTYPE=COUNTING_LINE_A
ANALYTICSRECORD_2_RULENAME=Countine Line A
ANALYTICSRECORD_2_OBJECTCLASS=Unclassified
ANALYTICSRECORD_2_METADATAPRESENT=FALSE
```

Example of return in XML:

```
<Response>
  <Code>0</Code>
  <Message>OK</Message>
  <Data>
    <AnalyticsRecords>
      <Count>2</Count>
      <AnalyticsRecord>
        <RecordCode>98370</RecordCode>
        <Camera>04</Camera>
        <StartDate>2015-09-27 00:12:21.351</StartDate>
        <EndDate>2015-09-27 00:12:21.351</EndDate>
        <Zone>Entrance</Zone>
        <EventType>PRESENCE</EventType>
        <RuleName>Presence</RuleName>
        <ObjectClass>Person</ObjectClass>
        <MetadataPresent>FALSE</MetadataPresent>
      </AnalyticsRecord>
      <AnalyticsRecord>
        <RecordCode>98371</RecordCode>
        <Camera>02</Camera>
        <StartDate>2015-09-27 00:25:40.000</StartDate>
        <EndDate>2015-09-27 00:25:40.000</EndDate>
        <Zone>Exit</Zone>
        <EventType>COUNTING_LINE_A</EventType>
        <RuleName>Counting Line A</RuleName>
        <ObjectClass>Unclassified</ObjectClass>
        <MetadataPresent>FALSE</MetadataPresent>
      </AnalyticsRecord>
    </AnalyticsRecords>
  </Data>
</Response>
```

Example of return in JSON:

```
{
  "Response": {
    "Code": 0,
    "Message": "OK",
    "Data": {
      "AnalyticsRecords": [
        {
          "RecordCode": 98370,
          "Camera": "04",
          "StartDate": "2015-09-27T00:12:21.351Z",
          "EndDate": "2015-09-27T00:12:21.351Z",
          "Zone": "Entrance",
          "EventType": "PRESENCE",
          "RuleName": "Presence",
          "ObjectClass": "Person",
          "MetadataPresent": false
        },
        {
          "RecordCode": 98371,
          "Camera": "37",
          "StartDate": "2015-09-27T00:25:40.000Z",
          "EndDate": "2020-05-28T00:01:42.000Z",
          "Zone": "Exit",
          "EventType": "COUNTING_LINE_A",
          "RuleName": "Counting Line A",
          "ObjectClass": "Unclassified",
          "MetadataPresent": false
        }
      ]
    }
  }
}
```

4.15.7 Requesting the metadata of a record

Return the internal metadata of an analytics record from the database.

Important: This command is not intended to be used by third parties as the format of Metadata is not publicly available.

Compatibility: Standard, Professional, Enterprise

Security level: Requires user authentication with rights to search for Analytics records

Method: HTTP GET

Syntax:

```
http://<server_address>/Interface/Analytics/GetMetadata?<argument=value>
[&<argument=value>...] [&<general\_argument>...]
```

Arguments:

Argument	Valid values	Description
RecordCode*	Integer	Record number

* Mandatory parameters

Example 1: Retrieve the metadata of record 100

```
http://192.168.0.1:8601/Interface/Analytics/GetMetadata?RecordCode=100
```

Example 1: Retrieve the metadata of record 100 with response in JSON format

```
http://192.168.0.1:8601/Interface/Analytics/GetMetadata?RecordCode=100&
ResponseFormat=JSON
```

Response:

A list of parameter-value pairs is returned

HTTP Return: 200 OK

Parameters of return:

Parameter	Type	Description
RECORDCODE	Integer	Record number
METADATA	String	Metadata in Base64 format

Example of return in text:

```
RESPONSE_CODE=0
RESPONSE_MESSAGE=OK
RECORDCODE=10
METADATA=AQIAAAIAS1BFR0RhdGEJ0wQAAP
```

Example of return in XML:

```
<Response>
    <Code>0</Code>
    <Message>OK</Message>
    <Data>
        <Analytics>
            <RecordCode>19492</RecordCode>
            <Metadata>AQIAAAIAS1BFR0RhdGEJ0wQAAP</Metadata>
        </Analytics>
    </Data>
</Response>
```

Example of return in JSON:

```
{
    "Response": {
        "Code": 0,
        "Message": "OK",
        "Data": {
            "Analytics": {
                "RecordCode": 19492,
                "Metadata": "AQIAAAIAS1BFR0RhdGEJ0wQAAP"
            }
        }
    }
}
```

4.16 Web Pages

4.16.1 Requesting the list of Web Pages

Requests the list of web pages over which the user has rights to access.

Compatibility: Standard, Professional, Enterprise

Security level: Requires user authentication

Method: HTTP GET

Syntax:

```
http://<server_address>/Interface/WebPages/GetWebPages [ ?<argument=value>
[ &<argument=value>... ] ] [ &<general_argument>... ]
```

Arguments:

Argument	Valid values	Description										
WebPages	APIMasks	Mask to filter the results. Specify which Web Pages must be returned based on the provided masks. Both DUID and Name will be compared with mask.										
Fields	DUID Name Description URL	Specifies the list of desired fields. If this parameter is omitted, all of the fields will be sent. The fields must be separated by commas <table border="1" style="margin-left: 20px;"> <thead> <tr> <th>Name</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>DUID</td> <td>DUID of the Web Page</td> </tr> <tr> <td>Name</td> <td>Name of the Web Page</td> </tr> <tr> <td>Description</td> <td>Description of the Web Page</td> </tr> <tr> <td>URL</td> <td>URL of the Web Page</td> </tr> </tbody> </table>	Name	Description	DUID	DUID of the Web Page	Name	Name of the Web Page	Description	Description of the Web Page	URL	URL of the Web Page
Name	Description											
DUID	DUID of the Web Page											
Name	Name of the Web Page											
Description	Description of the Web Page											
URL	URL of the Web Page											

Example 1: Request the list of web pages with all of the fields and response in XML

```
http://192.168.0.1:8601/Interface/WebPages/GetWebPages?ResponseFormat=XML
```

Example 2: Request the list of web pages with all of the fields and response in text

```
http://192.168.0.1:8601/Interface/WebPages/GetWebPages?ResponseFormat=Text
```

Example 3: Request the list of web pages with only name, response in JSON and authentication of the user Admin

```
http://192.168.0.1:8601/Interface/WebPages/GetWebPages?Fields=Name&
ResponseFormat=JSON&AuthUser=admin&AuthPass=pass
```

Example 4: Request the list of web pages starting with "A", with only name, response in XML and authentication with Admin user

```
http://192.168.0.1:8601/Interface/WebPages/GetWebPages?WebPages=A*&
Fields=Name&ResponseFormat=XML&AuthUser=admin&AuthPass=pass
```

Response:

A list with all of the web pages over which the user has rights to access is returned. The fields returned in the list will depend on the values informed in the argument **Fields**

HTTP Return: 200 OK

Parameters of return:

Fixed parameters:

Parameter	Type	Description
COUNT	Integer	Total number of web pages (Not included in JSON response)

Parameters of the list of maps:

Parameter	Type	Description
DUID	DUID	DUID of the Web Page
NAME	String	Name of the Web Page
DESCRIPTION	String	Description of the Web Page
URL	String	URL of the Web Page

Example of return in text:

```
RESPONSE_CODE=0
RESPONSE_MESSAGE=OK
COUNT=2
WEBPAGE_1_DUID=318D9BE6-58C8-459B-95A6-90D72AF52260
WEBPAGE_1_NAME=Blank
WEBPAGE_1_DESCRIPTION=Open Navigation
WEBPAGE_1_URL=
WEBPAGE_2_DUID=CD183EA5-DD06-4E2E-80C0-781A867BC828
WEBPAGE_2_NAME=Digifort
WEBPAGE_2_DESCRIPTION=Digifort Website
WEBPAGE_2_URL=http://www.digifort.com/
```

Example of return in XML:

```
<Response>
  <Code>0</Code>
  <Message>OK</Message>
  <Data>
    <WebPages>
      <Count>2</Count>
      <WebPage>
        <DUID>318D9BE6-58C8-459B-95A6-90D72AF52260</DUID>
        <Name>Blank</Name>
        <Description>Open Navigation</Description>
        <URL/>
      </WebPage>
      <WebPage>
        <DUID>CD183EA5-DD06-4E2E-80C0-781A867BC828</DUID>
        <Name>Digifort</Name>
        <Description>Digifort Website</Description>
        <URL>http://www.digifort.com/</URL>
      </WebPage>
    </WebPages>
  </Data>
</Response>
```

Example of return in JSON:

```
{
  "Response": {
    "Code": 0,
    "Message": "OK",
    "Data": {
      "WebPages": [
        {
          "DUID": "318D9BE6-58C8-459B-95A6-90D72AF52260",
          "Name": "Blank",
          "Description": "Open Navigation",
          "URL": ""
        },
        {
          "DUID": "CD183EA5-DD06-4E2E-80C0-781A867BC828",
          "Name": "Digifort",
          "Description": "Digifort Website",
          "URL": "http://www.digifort.com/"
        }
      ]
    }
  }
}
```

4.16.2 Activating / Deactivating Web Pages

Allows the activation or deactivation of multiple Web Pages simultaneously

Compatibility: Standard, Professional, Enterprise

Security level: Requires user authentication with rights to configure Web Pages

Method: HTTP GET

Syntax:

```
http://<server_address>/Interface/WebPages/Activation?  
<argument=value>[&<argument=value>...][&<general argument>...]
```

Arguments:

Argument	Valid values	Description						
WebPages*	String	<p>List of Web Pages (DUIDs or Names) to activate or deactivate.</p> <p>The Web Page DUIDs or Names must be separated by commas.</p> <p>This command is only mandatory when Action=Activate or Action=Deactivate</p>						
Action*	Activate Deactivate ActivateAll DeactivateAll	<p>Action to be executed</p> <table border="1" style="margin-left: auto; margin-right: auto;"> <thead> <tr> <th>Operation</th><th>Description</th></tr> </thead> <tbody> <tr> <td>Activate</td><td>Activate the Web Pages on WebPages argument</td></tr> <tr> <td>Deactivate</td><td>Deactivate the Web Pages on WebPages argument</td></tr> </tbody> </table>	Operation	Description	Activate	Activate the Web Pages on WebPages argument	Deactivate	Deactivate the Web Pages on WebPages argument
Operation	Description							
Activate	Activate the Web Pages on WebPages argument							
Deactivate	Deactivate the Web Pages on WebPages argument							

		ActivateAll	Activate all Web Pages from the server
		DeactivateAll	Deactivate all Web Pages from the server

* Mandatory parameters

Example 1: Activate all Web Pages from the server

```
http://192.168.0.1:8601/Interface/WebPages/Activation?Action=ActivateAll
```

Example 2: Activate Web Pages 3346826D-3CE3-4239-A56A-D81757A9C309 and 3EC95C91-E414-4B25-8D8B-D6DB68922127

```
http://192.168.0.1:8601/Interface/WebPages/Activation?
Action=Activate&WebPages=3346826D-3CE3-4239-A56A-D81757A9C309,
3EC95C91-E414-4B25-8D8B-D6DB68922127
```

Example 3: Activate Web Pages Web1 and Web2

```
http://192.168.0.1:8601/Interface/WebPages/Activation?
Action=Activate&WebPages=Web1,Web2
```

Example 4: Deactivate all Web Pages from the server

```
http://192.168.0.1:8601/Interface/WebPages/Activation?
Action=DeactivateAll
```

Response:

Default response of API.

HTTP Return HTTP: 200 OK

Parameters of return: [Default return of API](#)

4.17 Audit

4.17.1 Searching for audit records

Perform a search for audit records in the database

Compatibility: All editions

Security level: Requires user authentication with rights to search for audit records

Method: HTTP GET

Syntax:

```
http://<server_address>/Interface/Audit/Search
[?<argument=value>[&<argument=value>...][&<general\_argument>...]]
```

Arguments:

Argument	Valid values	Description
StartDate	APIDate	Search for records starting with the specified date
StartTime	APITime	Search for records starting with the specified time
EndDate	APIDate	End date of search. Search for all records inside the range of start and end. If this parameter is omitted, the value specified in

		StartDate parameter will be used
EndTime	APITime	End time of search. Search for all records inside the range of start and end. If this parameter is omitted, the value specified in StartTime parameter will be used
Category	USER_ACTION SERVER CONNECTION	Category of audit
Keyword	String	Keyword to search for records
ExactKeyword	TRUE FALSE	TRUE to search for exact keyword (Faster search) FALSE to search for keyword (Slower search)

Example 1: Search for all records between January 01, 2014 and February 01, 2014

```
http://192.168.0.1:8601/Interface/Audit/Search?StartDate=2014.01.01&StartTime=00.00.00.000&EndDate=2014.02.01&EndTime=23.59.59.999
```

Example 2: Search for all records of user Operator1

```
http://192.168.0.1:8601/Interface/Audit/Search?Keyword=Operator1
```

Example 3: Search for all records of user actions

```
http://192.168.0.1:8601/Interface/Audit/Search?Category=USER_ACTION
```

Response:

A list with all found audit records is returned.

HTTP Return: 200 OK

Parameters of return:

Fixed Parameter:

Parameter	Type	Description
COUNT	Integer	Total of records (Not included in JSON response)

Parameters of records list:

Parameter	Type	Description																								
DATETIME	APITimestamp	Record date and time																								
USER	String	Username																								
IP	String	IP address																								
EVENT	String	Type of audit event <table border="1" data-bbox="775 1510 1395 1900"> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>ADDED</td> <td>Object was added</td> </tr> <tr> <td>CONNECTED</td> <td>User has connected</td> </tr> <tr> <td>DISCONNECTED</td> <td>User disconnected</td> </tr> <tr> <td>BLOCKED</td> <td>User was blocked</td> </tr> <tr> <td>UNBLOCKED</td> <td>User was unblocked</td> </tr> <tr> <td>RESET</td> <td>Object was resetted</td> </tr> <tr> <td>MODIFIED</td> <td>Object was modified</td> </tr> <tr> <td>DELETED</td> <td>Object was deleted</td> </tr> <tr> <td>CREATED</td> <td>Object was created</td> </tr> <tr> <td>ACTIVATED</td> <td>Object was activated</td> </tr> <tr> <td>DEACTIVATED</td> <td>Object was deactivated</td> </tr> </tbody> </table>	Value	Description	ADDED	Object was added	CONNECTED	User has connected	DISCONNECTED	User disconnected	BLOCKED	User was blocked	UNBLOCKED	User was unblocked	RESET	Object was resetted	MODIFIED	Object was modified	DELETED	Object was deleted	CREATED	Object was created	ACTIVATED	Object was activated	DEACTIVATED	Object was deactivated
Value	Description																									
ADDED	Object was added																									
CONNECTED	User has connected																									
DISCONNECTED	User disconnected																									
BLOCKED	User was blocked																									
UNBLOCKED	User was unblocked																									
RESET	Object was resetted																									
MODIFIED	Object was modified																									
DELETED	Object was deleted																									
CREATED	Object was created																									
ACTIVATED	Object was activated																									
DEACTIVATED	Object was deactivated																									

Parameter	Type	Description																																							
		<table border="1"> <tr><td>STARTED</td><td>Operation has started</td></tr> <tr><td>GRANTED RIGHT</td><td>Rights has been granted</td></tr> <tr><td>DENIED RIGHT</td><td>Rights has been denied</td></tr> <tr><td>VIEWED</td><td>Object was viewed</td></tr> <tr><td>LOGIN</td><td>Connection was authenticated</td></tr> <tr><td>ERROR</td><td>Error</td></tr> <tr><td>START_CONTROL</td><td>User has started controlling a PTZ camera</td></tr> <tr><td>LOGOUT</td><td>User logged out</td></tr> <tr><td>ENDED</td><td>Operation has ended</td></tr> </table>	STARTED	Operation has started	GRANTED RIGHT	Rights has been granted	DENIED RIGHT	Rights has been denied	VIEWED	Object was viewed	LOGIN	Connection was authenticated	ERROR	Error	START_CONTROL	User has started controlling a PTZ camera	LOGOUT	User logged out	ENDED	Operation has ended																					
STARTED	Operation has started																																								
GRANTED RIGHT	Rights has been granted																																								
DENIED RIGHT	Rights has been denied																																								
VIEWED	Object was viewed																																								
LOGIN	Connection was authenticated																																								
ERROR	Error																																								
START_CONTROL	User has started controlling a PTZ camera																																								
LOGOUT	User logged out																																								
ENDED	Operation has ended																																								
OBJECTTYPE	String	Identifies the type of object from audit <table border="1"> <tr><th>Object Type</th></tr> <tr><td>SERVER CONFIGURATION</td></tr> <tr><td>DIRECTORY</td></tr> <tr><td>ALERT CONTACT</td></tr> <tr><td>ALERT GROUP</td></tr> <tr><td>USER</td></tr> <tr><td>CAMERA</td></tr> <tr><td>LOG SERVER SETTINGS</td></tr> <tr><td>LICENSE</td></tr> <tr><td>WEB SERVER SETTINGS</td></tr> <tr><td>IP FILTER</td></tr> <tr><td>SCREENSTYLE</td></tr> <tr><td>EVENT LOG SETTINGS</td></tr> <tr><td>VIEW</td></tr> <tr><td>DISK LIMIT</td></tr> <tr><td>ADVANCED SEARCH</td></tr> <tr><td>MEDIA PLAYBACK</td></tr> <tr><td>ARCHIVING</td></tr> <tr><td>IO DEVICE</td></tr> <tr><td>USER GROUP</td></tr> <tr><td>IP</td></tr> <tr><td>NETWORK UNIT</td></tr> <tr><td>MAP</td></tr> <tr><td>PLAYBACK VIEW</td></tr> <tr><td>MEDIA PROFILE</td></tr> <tr><td>ALARM OUTPUT ACTION</td></tr> <tr><td>SCHEDULED EVENT</td></tr> <tr><td>LPR LIST</td></tr> <tr><td>GLOBAL EVENT</td></tr> <tr><td>ANALYTICS CONFIGURATION</td></tr> <tr><td>LPR CONFIGURATION</td></tr> <tr><td>SERVER</td></tr> <tr><td>PUBLIC VIEW</td></tr> <tr><td>CAMERA PRIVACY MODE</td></tr> <tr><td>AUDIO OUTPUT DEVICE GROUP</td></tr> <tr><td>RTSP INTERFACE</td></tr> <tr><td>RTSP INTERFACE SETTINGS</td></tr> <tr><td>CAMERA PTZ</td></tr> <tr><td>LPR EVENT</td></tr> </table>	Object Type	SERVER CONFIGURATION	DIRECTORY	ALERT CONTACT	ALERT GROUP	USER	CAMERA	LOG SERVER SETTINGS	LICENSE	WEB SERVER SETTINGS	IP FILTER	SCREENSTYLE	EVENT LOG SETTINGS	VIEW	DISK LIMIT	ADVANCED SEARCH	MEDIA PLAYBACK	ARCHIVING	IO DEVICE	USER GROUP	IP	NETWORK UNIT	MAP	PLAYBACK VIEW	MEDIA PROFILE	ALARM OUTPUT ACTION	SCHEDULED EVENT	LPR LIST	GLOBAL EVENT	ANALYTICS CONFIGURATION	LPR CONFIGURATION	SERVER	PUBLIC VIEW	CAMERA PRIVACY MODE	AUDIO OUTPUT DEVICE GROUP	RTSP INTERFACE	RTSP INTERFACE SETTINGS	CAMERA PTZ	LPR EVENT
Object Type																																									
SERVER CONFIGURATION																																									
DIRECTORY																																									
ALERT CONTACT																																									
ALERT GROUP																																									
USER																																									
CAMERA																																									
LOG SERVER SETTINGS																																									
LICENSE																																									
WEB SERVER SETTINGS																																									
IP FILTER																																									
SCREENSTYLE																																									
EVENT LOG SETTINGS																																									
VIEW																																									
DISK LIMIT																																									
ADVANCED SEARCH																																									
MEDIA PLAYBACK																																									
ARCHIVING																																									
IO DEVICE																																									
USER GROUP																																									
IP																																									
NETWORK UNIT																																									
MAP																																									
PLAYBACK VIEW																																									
MEDIA PROFILE																																									
ALARM OUTPUT ACTION																																									
SCHEDULED EVENT																																									
LPR LIST																																									
GLOBAL EVENT																																									
ANALYTICS CONFIGURATION																																									
LPR CONFIGURATION																																									
SERVER																																									
PUBLIC VIEW																																									
CAMERA PRIVACY MODE																																									
AUDIO OUTPUT DEVICE GROUP																																									
RTSP INTERFACE																																									
RTSP INTERFACE SETTINGS																																									
CAMERA PTZ																																									
LPR EVENT																																									

Parameter	Type	Description								
		<table border="1"> <tr><td>LPR PLATE</td></tr> <tr><td>MEDIA PLAYBACK EXPORTATION</td></tr> <tr><td>BOOKMARK</td></tr> <tr><td>FAILOVER SERVER MONITOR</td></tr> <tr><td>LPR PLATE CATEGORY GROUP</td></tr> <tr><td>OPERATIONAL MAP</td></tr> <tr><td>RECORDING LOCK</td></tr> </table>	LPR PLATE	MEDIA PLAYBACK EXPORTATION	BOOKMARK	FAILOVER SERVER MONITOR	LPR PLATE CATEGORY GROUP	OPERATIONAL MAP	RECORDING LOCK	
LPR PLATE										
MEDIA PLAYBACK EXPORTATION										
BOOKMARK										
FAILOVER SERVER MONITOR										
LPR PLATE CATEGORY GROUP										
OPERATIONAL MAP										
RECORDING LOCK										
OBJECTNAME	String	Name of the object (If applicable)								
COMPLEMENT	String	Additional information (If applicable)								
EVENTTYPE	String	<p>Type of event</p> <table border="1"> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr><td>INFO</td><td>Event is informational</td></tr> <tr><td>WARNING</td><td>Warning event</td></tr> <tr><td>ERROR</td><td>Error event</td></tr> </tbody> </table>	Value	Description	INFO	Event is informational	WARNING	Warning event	ERROR	Error event
Value	Description									
INFO	Event is informational									
WARNING	Warning event									
ERROR	Error event									
CATEGORY	String	<p>Category of the event</p> <table border="1"> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr><td>USER_ACTION</td><td>User action event</td></tr> <tr><td>SERVER_CONNECTION</td><td>Server connection event</td></tr> </tbody> </table>	Value	Description	USER_ACTION	User action event	SERVER_CONNECTION	Server connection event		
Value	Description									
USER_ACTION	User action event									
SERVER_CONNECTION	Server connection event									

Example of return in text:

```

RESPONSE_CODE=0
RESPONSE_MESSAGE=OK
COUNT=2
AUDITRECORD_1_DATETIME=2018-03-12 10:00:00.017
AUDITRECORD_1_USER=admin
AUDITRECORD_1_IP=192.168.0.10
AUDITRECORD_1_EVENT=ERROR
AUDITRECORD_1_OBJECTTYPE=RTSP_INTERFACE
AUDITRECORD_1_OBJECTNAME=87
AUDITRECORD_1_COMPLEMENT=Camera deactivated
AUDITRECORD_1_EVENTTYPE=ERROR
AUDITRECORD_1_CATEGORY=USER_ACTION
AUDITRECORD_2_DATETIME=2018-03-12 10:00:00.332
AUDITRECORD_2_USER=operator
AUDITRECORD_2_IP=192.168.0.11
AUDITRECORD_2_EVENT=VIEWED
AUDITRECORD_2_OBJECTTYPE=CAMERA
AUDITRECORD_2_OBJECTNAME=75
AUDITRECORD_2_COMPLEMENT=Live view via relay
AUDITRECORD_2_EVENTTYPE=INFO
AUDITRECORD_2_CATEGORY=USER_ACTION

```

Example of return in XML:

```
<Response>
  <Code>0</Code>
  <Message>OK</Message>
  <Data>
    <AuditRecords>
      <Count>2</Count>
      <AuditRecord>
        <DateTime>2018-03-12 10:00:00.017</DateTime>
        <User>admin</User>
        <IP>192.168.0.10</IP>
        <Event>ERROR</Event>
        <ObjectType>RTSP_INTERFACE</ObjectType>
        <ObjectName>87</ObjectName>
        <Complement>Camera deactivated</Complement>
        <EventType>ERROR</EventType>
        <Category>USER_ACTION</Category>
      </AuditRecord>
      <AuditRecord>
        <DateTime>2018-03-12 10:00:00.332</DateTime>
        <User>operator</User>
        <IP>192.168.0.11</IP>
        <Event>VIEWED</Event>
        <ObjectType>CAMERA</ObjectType>
        <ObjectName>75</ObjectName>
        <Complement>Live view via relay</Complement>
        <EventType>INFO</EventType>
        <Category>USER_ACTION</Category>
      </AuditRecord>
    </AuditRecords>
  </Data>
</Response>
```

Example of return in JSON:

```
{
  "Response": {
    "Code": 0,
    "Message": "OK",
    "Data": {
      "AuditRecords": [
        {
          "DateTime": "2018-03-12 10:00:00.017",
          "User": "admin",
          "IP": "192.168.0.10",
          "Event": "ERROR",
          "ObjectType": "RTSP_INTERFACE",
          "ObjectName": "87",
          "Complement": "Camera deactivated",
          "EventType": "ERROR",
          "Category": "USER_ACTION"
        },
        {
          "DateTime": "2018-03-12 10:00:00.332",
          "User": "operator",
          "IP": "192.168.0.11",
          "Event": "VIEWED",
          "ObjectType": "CAMERA",
          "ObjectName": "75",
          "Complement": "Live view via relay",
          "EventType": "INFO",
          "Category": "USER_ACTION"
        }
      ]
    }
  }
}
```

4.17.2 Monitoring the audit records

Starts the monitoring of audit records from the server. By using this command, you will receive in real time all events registered by audit.

Compatibility: All editions

Security level: Requires user authentication with rights to search for audit records

Method: HTTP GET

Syntax:

```
http://<server_address>/Interface/Audit/Monitor  
[?<argument=value>[&<argument=value>...]] [&<general argument>...]
```

Arguments:

Argument	Valid values	Description
KeepAliveInterval	Integer. 1 >= X <= 120	Specify the time (in seconds) of Keep-Alive packet sending. The Keep-Alive packet is used to control the connection. If this parameter is omitted, the default

	value of 5 will be used
--	-------------------------

Example 1: Start the audit record monitoring with response in XML

```
http://192.168.0.1:8601/Interface/Audit/Monitor?ResponseFormat=XML
```

Example 2: Start the audit record monitoring with response in text and keep-alive interval of 60 seconds

```
http://192.168.0.1:8601/Interface/Audit/Monitor?ResponseFormat=Text&KeepAliveInterval=60
```

Example 3: Start the audit record monitoring with response in XML and authentication with Admin user

```
http://192.168.0.1:8601/Interface/Audit/Monitor?ResponseFormat=XML&AuthUser=admin&AuthPass=pass
```

Response:

A stream of events data will be sent using HTTP Multipart x-mixed-replace transmission.

The response data stream is continuous, so an event of Keep-Alive will be sent every X seconds (Configurable by KeepAliveInterval parameter), with this event you can verify if the TCP connection is still open.

Each event is separated by the multipart boundary --DigifortBoundary.

HTTP Return: 200 OK

Parameter of return:

The return parameters are divided into two subcategories: EventData and AuditRecord.

EventData parameters:

Parameter	Type	Description						
TYPE	String	<p>Event type</p> <table border="1"> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>AUDIT_RECORD</td> <td>Contains audit record data</td> </tr> <tr> <td>KEEP_ALIVE</td> <td>HTTP Connection Keep-Alive message. Sent every 5 seconds to keep the TCP / HTTP connection</td> </tr> </tbody> </table>	Value	Description	AUDIT_RECORD	Contains audit record data	KEEP_ALIVE	HTTP Connection Keep-Alive message. Sent every 5 seconds to keep the TCP / HTTP connection
Value	Description							
AUDIT_RECORD	Contains audit record data							
KEEP_ALIVE	HTTP Connection Keep-Alive message. Sent every 5 seconds to keep the TCP / HTTP connection							

AuditRecord parameters:

Parameter	Type	Description																
TIMESTAMP	APITimestamp	Local date and time of the event																
UTCTIMESTAMP	APITimestamp	UTC date and time of the event																
USER	String	Username																
IP	String	IP address																
EVENT	String	<p>Type of audit event</p> <table border="1"> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>ADDED</td> <td>Object was added</td> </tr> <tr> <td>CONNECTED</td> <td>User has connected</td> </tr> <tr> <td>DISCONNECTED</td> <td>User disconnected</td> </tr> <tr> <td>BLOCKED</td> <td>User was blocked</td> </tr> <tr> <td>UNBLOCKED</td> <td>User was unblocked</td> </tr> <tr> <td>RESET</td> <td>Object was resetted</td> </tr> <tr> <td>MODIFIED</td> <td>Object was modified</td> </tr> </tbody> </table>	Value	Description	ADDED	Object was added	CONNECTED	User has connected	DISCONNECTED	User disconnected	BLOCKED	User was blocked	UNBLOCKED	User was unblocked	RESET	Object was resetted	MODIFIED	Object was modified
Value	Description																	
ADDED	Object was added																	
CONNECTED	User has connected																	
DISCONNECTED	User disconnected																	
BLOCKED	User was blocked																	
UNBLOCKED	User was unblocked																	
RESET	Object was resetted																	
MODIFIED	Object was modified																	

Parameter	Type	Description																																			
		<table border="1"> <tr><td>DELETED</td><td>Object was deleted</td></tr> <tr><td>CREATED</td><td>Object was created</td></tr> <tr><td>ACTIVATED</td><td>Object was activated</td></tr> <tr><td>DEACTIVATED</td><td>Object was deactivated</td></tr> <tr><td>STARTED</td><td>Operation has started</td></tr> <tr><td>GRANTED_RIGHT</td><td>Rights has been granted</td></tr> <tr><td>DENIED_RIGHT</td><td>Rights has been denied</td></tr> <tr><td>VIEWED</td><td>Object was viewed</td></tr> <tr><td>LOGIN</td><td>Connection was authenticated</td></tr> <tr><td>ERROR</td><td>Error</td></tr> <tr><td>START_CONTROL</td><td>User has started controlling a PTZ camera</td></tr> <tr><td>LOGOUT</td><td>User logged out</td></tr> <tr><td>ENDED</td><td>Operation has ended</td></tr> </table>	DELETED	Object was deleted	CREATED	Object was created	ACTIVATED	Object was activated	DEACTIVATED	Object was deactivated	STARTED	Operation has started	GRANTED_RIGHT	Rights has been granted	DENIED_RIGHT	Rights has been denied	VIEWED	Object was viewed	LOGIN	Connection was authenticated	ERROR	Error	START_CONTROL	User has started controlling a PTZ camera	LOGOUT	User logged out	ENDED	Operation has ended									
DELETED	Object was deleted																																				
CREATED	Object was created																																				
ACTIVATED	Object was activated																																				
DEACTIVATED	Object was deactivated																																				
STARTED	Operation has started																																				
GRANTED_RIGHT	Rights has been granted																																				
DENIED_RIGHT	Rights has been denied																																				
VIEWED	Object was viewed																																				
LOGIN	Connection was authenticated																																				
ERROR	Error																																				
START_CONTROL	User has started controlling a PTZ camera																																				
LOGOUT	User logged out																																				
ENDED	Operation has ended																																				
OBJECTTYPE	String	Identifies the type of object from audit <table border="1"> <thead> <tr><th>Object Type</th></tr> </thead> <tbody> <tr><td>SERVER CONFIGURATION</td></tr> <tr><td>DIRECTORY</td></tr> <tr><td>ALERT CONTACT</td></tr> <tr><td>ALERT GROUP</td></tr> <tr><td>USER</td></tr> <tr><td>CAMERA</td></tr> <tr><td>LOG SERVER SETTINGS</td></tr> <tr><td>LICENSE</td></tr> <tr><td>WEB SERVER SETTINGS</td></tr> <tr><td>IP FILTER</td></tr> <tr><td>SCREENSTYLE</td></tr> <tr><td>EVENT LOG SETTINGS</td></tr> <tr><td>VIEW</td></tr> <tr><td>DISK LIMIT</td></tr> <tr><td>ADVANCED SEARCH</td></tr> <tr><td>MEDIA PLAYBACK</td></tr> <tr><td>ARCHIVING</td></tr> <tr><td>IO DEVICE</td></tr> <tr><td>USER GROUP</td></tr> <tr><td>IP</td></tr> <tr><td>NETWORK UNIT</td></tr> <tr><td>MAP</td></tr> <tr><td>PLAYBACK VIEW</td></tr> <tr><td>MEDIA PROFILE</td></tr> <tr><td>ALARM OUTPUT ACTION</td></tr> <tr><td>SCHEDULED EVENT</td></tr> <tr><td>LPR LIST</td></tr> <tr><td>GLOBAL EVENT</td></tr> <tr><td>ANALYTICS CONFIGURATION</td></tr> <tr><td>LPR CONFIGURATION</td></tr> <tr><td>SERVER</td></tr> <tr><td>PUBLIC VIEW</td></tr> <tr><td>CAMERA PRIVACY MODE</td></tr> <tr><td>AUDIO OUTPUT DEVICE GROUP</td></tr> </tbody> </table>	Object Type	SERVER CONFIGURATION	DIRECTORY	ALERT CONTACT	ALERT GROUP	USER	CAMERA	LOG SERVER SETTINGS	LICENSE	WEB SERVER SETTINGS	IP FILTER	SCREENSTYLE	EVENT LOG SETTINGS	VIEW	DISK LIMIT	ADVANCED SEARCH	MEDIA PLAYBACK	ARCHIVING	IO DEVICE	USER GROUP	IP	NETWORK UNIT	MAP	PLAYBACK VIEW	MEDIA PROFILE	ALARM OUTPUT ACTION	SCHEDULED EVENT	LPR LIST	GLOBAL EVENT	ANALYTICS CONFIGURATION	LPR CONFIGURATION	SERVER	PUBLIC VIEW	CAMERA PRIVACY MODE	AUDIO OUTPUT DEVICE GROUP
Object Type																																					
SERVER CONFIGURATION																																					
DIRECTORY																																					
ALERT CONTACT																																					
ALERT GROUP																																					
USER																																					
CAMERA																																					
LOG SERVER SETTINGS																																					
LICENSE																																					
WEB SERVER SETTINGS																																					
IP FILTER																																					
SCREENSTYLE																																					
EVENT LOG SETTINGS																																					
VIEW																																					
DISK LIMIT																																					
ADVANCED SEARCH																																					
MEDIA PLAYBACK																																					
ARCHIVING																																					
IO DEVICE																																					
USER GROUP																																					
IP																																					
NETWORK UNIT																																					
MAP																																					
PLAYBACK VIEW																																					
MEDIA PROFILE																																					
ALARM OUTPUT ACTION																																					
SCHEDULED EVENT																																					
LPR LIST																																					
GLOBAL EVENT																																					
ANALYTICS CONFIGURATION																																					
LPR CONFIGURATION																																					
SERVER																																					
PUBLIC VIEW																																					
CAMERA PRIVACY MODE																																					
AUDIO OUTPUT DEVICE GROUP																																					

Parameter	Type	Description								
		RTSP INTERFACE RTSP INTERFACE SETTINGS CAMERA PTZ LPR EVENT LPR PLATE MEDIA PLAYBACK EXPORTATION BOOKMARK FAILOVER SERVER MONITOR LPR PLATE CATEGORY GROUP OPERATIONAL MAP RECORDING LOCK								
OBJECTNAME	String	Name of the object (If applicable)								
COMPLEMENT	String	Additional information (If applicable)								
EVENTTYPE	String	Type of event <table border="1" style="margin-left: 20px;"> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>INFO</td><td>Event is informational</td></tr> <tr> <td>WARNING</td><td>Warning event</td></tr> <tr> <td>ERROR</td><td>Error event</td></tr> </tbody> </table>	Value	Description	INFO	Event is informational	WARNING	Warning event	ERROR	Error event
Value	Description									
INFO	Event is informational									
WARNING	Warning event									
ERROR	Error event									
CATEGORY	String	Category of the event <table border="1" style="margin-left: 20px;"> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>USER_ACTION</td><td>User action event</td></tr> <tr> <td>SERVER CONNECTION</td><td>Server connection event</td></tr> </tbody> </table>	Value	Description	USER_ACTION	User action event	SERVER CONNECTION	Server connection event		
Value	Description									
USER_ACTION	User action event									
SERVER CONNECTION	Server connection event									

Example of return in text:

```
--DigifortBoundary
Content-Type: text/plain; charset=UTF-8
Content-Length: 415

RESPONSE_CODE=0
RESPONSE_MESSAGE=OK
EVENTDATA_TYPE=AUDIT_RECORD
AUDITRECORD_TIMESTAMP=2018-03-23 14:28:39.559
AUDITRECORD_UTCTIMESTAMP=2018-03-23 18:28:39.559
AUDITRECORD_USER=admin
AUDITRECORD_IP=127.0.0.1
AUDITRECORD_EVENT=LOGIN
AUDITRECORD_OBJECTTYPE=SERVER
AUDITRECORD_OBJECTNAME=1
AUDITRECORD_COMPLEMENT=Administration Client
AUDITRECORD_EVENTTYPE=INFO
AUDITRECORD_CATEGORY=SERVER_CONNECTION
--DigifortBoundary
Content-Type: text/plain; charset=UTF-8
Content-Length: 65

RESPONSE_CODE=0
RESPONSE_MESSAGE=OK
EVENTDATA_TYPE=KEEP_ALIVE
--DigifortBoundary
Content-Type: text/plain; charset=UTF-8
Content-Length: 393
..
..
```

Example of return in XML:

```
--DigifortBoundary
Content-Type: text/xml; charset=UTF-8
Content-Length: 525

<?xml version="1.0" encoding="UTF-8" ?>
<Response>
    <Code>0</Code>
    <Message>OK</Message>
    <Data>
        <Event>
            <EventData>
                <Type>AUDIT_RECORD</Type>
            </EventData>
            <AuditRecord>
                <Timestamp>2018-03-23 14:28:39.559</Timestamp>
                <UTCTimestamp>2018-03-23 18:28:39.559</UTCTimestamp>
                <User>admin</User>
                <IP>127.0.0.1</IP>
                <Event>LOGIN</Event>
                <ObjectType>SERVER</ObjectType>
                <ObjectName>2</ObjectName>
                <Complement>Administration Client</Complement>
                <EventType>INFO</EventType>
                <Category>SERVER_CONNECTION</Category>
            </AuditRecord>
        </Event>
    </Data>
</Response>
--DigifortBoundary
Content-Type: text/xml; charset=UTF-8
Content-Length: 169

<?xml version="1.0" encoding="UTF-8" ?>
<Response>
    <Code>0</Code>
    <Message>OK</Message>
    <Data>
        <Event>
            <EventData>
                <Type>KEEP_ALIVE</Type>
            </EventData>
        </Event>
    </Data>
</Response>
--DigifortBoundary
Content-Type: text/xml; charset=UTF-8
Content-Length: 169
..
..
```

Example of return in text:

```
--DigifortBoundary
Content-Type: application/json; charset=UTF-8
Content-Length: 578

{
    "Response": {
        "Code": 0,
        "Message": "OK",
        "Data": {
            "Event": {
                "EventData": {
                    "Type": "AUDIT_RECORD"
                },
                "AuditRecord": {
                    "Timestamp": "2018-03-23 14:28:39.559",
                    "UTCTimestamp": "2018-03-23 18:28:39.559",
                    "User": "admin",
                    "IP": "127.0.0.1",
                    "Event": "LOGIN",
                    "ObjectType": "SERVER",
                    "ObjectName": "2",
                    "Complement": "Administration Client",
                    "EventType": "INFO",
                    "Category": "SERVER_CONNECTION"
                }
            }
        }
    }
}
--DigifortBoundary
Content-Type: text/xml; charset=UTF-8
Content-Length: 168

{
    "Response": {
        "Code": 0,
        "Message": "OK",
        "Data": {
            "Event": {
                "EventData": {
                    "Type": "KEEP_ALIVE"
                }
            }
        }
    }
}
..
```

4.18 Failover

4.18.1 Requesting the list of failover monitors

Requests the list of failover monitors from the server.

Compatibility: Enterprise

Security level: Requires user authentication with rights to configure the server

Method: HTTP GET

Syntax:

```
http://<server_address>/Interface/Failover/GetFailoverMonitors
[?<argument=value> [&<argument=value>...]] [&<general\_argument>...]
```

Arguments:

Argument	Valid values	Description												
FailoverMonitors	APIMasks	Mask to filter the results. Specify which failover monitors must be returned based on the provided masks.												
Fields	Name Description Active Address Port	<p>Specifies the list of desired fields. If this parameter is omitted, all of the fields will be sent.</p> <p>The fields must be separated by commas</p> <table border="1"> <thead> <tr> <th>Name</th><th>Description</th></tr> </thead> <tbody> <tr> <td>Name</td><td>Name of the failover monitor</td></tr> <tr> <td>Description</td><td>Description of the failover monitor</td></tr> <tr> <td>Active</td><td>Monitor activated / deactivated</td></tr> <tr> <td>Address</td><td>Address of the monitored server</td></tr> <tr> <td>Port</td><td>Port of the monitored server</td></tr> </tbody> </table>	Name	Description	Name	Name of the failover monitor	Description	Description of the failover monitor	Active	Monitor activated / deactivated	Address	Address of the monitored server	Port	Port of the monitored server
Name	Description													
Name	Name of the failover monitor													
Description	Description of the failover monitor													
Active	Monitor activated / deactivated													
Address	Address of the monitored server													
Port	Port of the monitored server													

Example 1: Requests the list of failover monitors with all of the fields and response in XML

```
http://192.168.0.1:8601/Interface/Failover/GetFailoverMonitors?
ResponseFormat=XML
```

Example 2: Requests the list of failover monitors with all of the fields and response in text

```
http://192.168.0.1:8601/Interface/Failover/GetFailoverMonitors?
ResponseFormat=Text
```

Example 3: Requests the list of failover monitors with only name, response in XML and authentication of the user Admin

```
http://192.168.0.1:8601/Interface/Failover/GetFailoverMonitors?Fields=Name&
ResponseFormat=XML&AuthUser=admin&AuthPass=pass
```

Example 4: Request the list of failover monitors starting with "A", with only name, response in XML and authentication with Admin user

```
http://192.168.0.1:8601/Interface/Failover/GetFailoverMonitors?
FailoverMonitors=A*&Fields=Name&ResponseFormat=XML&AuthUser=admin&
AuthPass=pass
```

Response:

A list with all of the failover monitors from the server is returned. The fields returned in the list will depend on the values informed in the argument `Fields`

HTTP Return: 200 OK

Parameters of return:**Fixed parameters:**

Parameter	Type	Description
COUNT	Integer	Total number of failover monitors (Not included in JSON response)

Parameters of the list of failover monitors:

Parameter	Type	Description
NAME	String	Name of the failover monitor
DESCRIPTION	String	Description of the failover monitor
ACTIVE	Boolean	Failover monitor activated / deactivated
ADDRESS	String	Address of the monitored server
PORT	Integer	Port of the monitored server

Example of return in text:

```
RESPONSE_CODE=0
RESPONSE_MESSAGE=OK
COUNT=2
FAILOVERMONITOR_1_NAME=Monitor1
FAILOVERMONITOR_1_DESCRIPTION=Failover Monitor 1
FAILOVERMONITOR_1_ACTIVE=TRUE
FAILOVERMONITOR_1_ADDRESS=192.168.0.1
FAILOVERMONITOR_1_PORT=8600
FAILOVERMONITOR_2_NAME=Monitor2
FAILOVERMONITOR_2_DESCRIPTION=Failover Monitor 2
FAILOVERMONITOR_2_ACTIVE=FALSE
FAILOVERMONITOR_2_ADDRESS=192.168.0.2
FAILOVERMONITOR_2_PORT=8600
```

Example of return in XML:

```

<Response>
  <Code>0</Code>
  <Message>OK</Message>
  <Data>
    <FailoverMonitors>
      <Count>2</Count>
      <FailoverMonitor>
        <Name>Monitor1</Name>
        <Description>Failover Monitor 1</Description>
        <Active>TRUE</Active>
        <Address>192.168.0.1</Address>
        <Port>8600</Port>
      </FailoverMonitor>
      <FailoverMonitor>
        <Name>Monitor2</Name>
        <Description>Failover Monitor 2</Description>
        <Active>FALSE</Active>
        <Address>192.168.0.2</Address>
        <Port>8600</Port>
      </FailoverMonitor>
    </FailoverMonitors>
  </Data>
</Response>

```

Example of return in JSON:

```
{
  "Response": {
    "Code": 0,
    "Message": "OK",
    "Data": {
      "FailoverMonitors": [
        {
          "Name": "Monitor1",
          "Description": "Failover Monitor 1",
          "Active": true,
          "Address": "192.168.0.1",
          "Port": 8600
        },
        {
          "Name": "Monitor2",
          "Description": "Failover Monitor 2",
          "Active": false,
          "Address": "192.168.0.2",
          "Port": 8600
        }
      ]
    }
  }
}
```

4.18.2 Requesting the status of failover monitors

Request the status of the failover monitors from the server.

Compatibility: Enterprise

Security level: Requires user authentication with rights to monitor the server health and status

Method: HTTP GET

Syntax:

```
http://<server_address>/Interface/Failover/GetStatus[?<argument=value>
[&<argument=value>...][&<general\_argument>...]]
```

Arguments:

Argument	Valid values	Description										
FailoverMonitors	APIMasks	Mask to filter the results. Specify which failover monitors must be returned based on the provided masks.										
Fields	Active Working Status StatusMessage	Specifies the list of desired fields. In case this parameter is omitted, all of the fields will be sent. The fields must be separated by commas <table border="1"> <thead> <tr> <th>Name</th><th>Description</th></tr> </thead> <tbody> <tr> <td>Active</td><td>Identify if the monitor is active</td></tr> <tr> <td>Working</td><td>Identify if the monitor is working</td></tr> <tr> <td>Status</td><td>Status of monitor</td></tr> <tr> <td>StatusMessage</td><td>Friendly status message</td></tr> </tbody> </table>	Name	Description	Active	Identify if the monitor is active	Working	Identify if the monitor is working	Status	Status of monitor	StatusMessage	Friendly status message
Name	Description											
Active	Identify if the monitor is active											
Working	Identify if the monitor is working											
Status	Status of monitor											
StatusMessage	Friendly status message											
Active	TRUE FALSE	In case this parameter is specified, a filter will be applied and only the monitors that matches this filter will be returned, thus, in case the value Active=TRUE is specified, the result will only include the active monitors, while if the value Active=FALSE is specified, the result will only include the deactivated monitors.										
Working	TRUE FALSE	In case this parameter is specified, a filter will be applied and only the monitors that matches this filter will be returned, thus, in case the value Working=TRUE is specified, the result will only include the working monitors, while if the value Working=FALSE is specified, the result will only include the monitors that are out of order.										

Example 1: Request the status of all monitors with all fields and response in XML

```
http://192.168.0.1:8601/Interface/Failover/GetStatus?ResponseFormat=XML
```

Example 2: Request the status of all active monitors with response in text

```
http://192.168.0.1:8601/Interface/Failover/GetStatus?Active=TRUE&
ResponseFormat=Text
```

Example 3: Request the status of all active monitors starting with A, with response in text

```
http://192.168.0.1:8601/Interface/Failover/GetStatus?FailoverMonitors=A*&
Active=TRUE&ResponseFormat=Text
```

Example 4: Request the status of all active monitors that are not working, with response in XML and authentication with admin user (No password)

```
http://192.168.0.1:8601/Interface/Failover/GetStatus?Active=TRUE&Working=FALSE&ResponseFormat=XML&AuthUser=admin
```

Response:

A list with the status of all of the monitors from the server is returned. The fields returned in the will depend on the values informed in the argument **Fields**

HTTP Return: 200 OK

Parameters of return:**Fixed Parameter:**

Parameter	Type	Description
COUNT	Integer	Total number of failover monitors (Not included in JSON response)

Parameters in the list of status of failover monitors:

Parameter	Type	Description														
NAME	String	Name of the failover monitor														
ACTIVE	Boolean	Identify if the failover monitor is active														
WORKING	Boolean	Identify if the failover monitor is working														
STATUS	String	Status of failover monitor <table border="1" data-bbox="652 939 1428 1214"> <thead> <tr> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>NO_FAILOVER_LICENSE</td> <td>There are no failover licenses in the server</td> </tr> <tr> <td>DISCONNECTED</td> <td>Monitor is disconnected from server</td> </tr> <tr> <td>CONNECTING</td> <td>Monitor is connecting to server</td> </tr> <tr> <td>AUTHENTICATING</td> <td>Monitor is authenticating</td> </tr> <tr> <td>AUTHENTICATION_ERROR</td> <td>Admin password is wrong</td> </tr> <tr> <td>WORKING</td> <td>Monitored server is working</td> </tr> </tbody> </table>	Type	Description	NO_FAILOVER_LICENSE	There are no failover licenses in the server	DISCONNECTED	Monitor is disconnected from server	CONNECTING	Monitor is connecting to server	AUTHENTICATING	Monitor is authenticating	AUTHENTICATION_ERROR	Admin password is wrong	WORKING	Monitored server is working
Type	Description															
NO_FAILOVER_LICENSE	There are no failover licenses in the server															
DISCONNECTED	Monitor is disconnected from server															
CONNECTING	Monitor is connecting to server															
AUTHENTICATING	Monitor is authenticating															
AUTHENTICATION_ERROR	Admin password is wrong															
WORKING	Monitored server is working															
STATUSMESSAGE	String	Friendly status message														

Example of return in text:

```
RESPONSE_CODE=0
RESPONSE_MESSAGE=OK
COUNT=2
FAILOVERMONITOR_1_NAME=Monitor1
FAILOVERMONITOR_1_ACTIVE=TRUE
FAILOVERMONITOR_1_WORKING=TRUE
FAILOVERMONITOR_1_STATUS=WORKING
FAILOVERMONITOR_1_STATUSMESSAGE=Working...
FAILOVERMONITOR_2_NAME=Monitor2
FAILOVERMONITOR_2_ACTIVE=FALSE
FAILOVERMONITOR_2_WORKING=FALSE
FAILOVERMONITOR_2_STATUS=DISCONNECTED
FAILOVERMONITOR_2_STATUSMESSAGE=Disconnected...
```

Example of return in XML:

```

<Response>
  <Code>0</Code>
  <Message>OK</Message>
  <Data>
    <FailoverMonitors>
      <Count>2</Count>
      <FailoverMonitor>
        <Name>Monitor1</Name>
        <Active>TRUE</Active>
        <Working>TRUE</Working>
        <Status>WORKING</Status>
        <StatusMessage>Working...</StatusMessage>
      </FailoverMonitor>
      <FailoverMonitor>
        <Name>Monitor2</Name>
        <Active>FALSE</Active>
        <Working>FALSE</Working>
        <Status>DISCONNECTED</Status>
        <StatusMessage>Disconnected...</StatusMessage>
      </FailoverMonitor>
    </FailoverMonitors>
  </Data>
</Response>

```

Example of return in JSON:

```
{
  "Response": {
    "Code": 0,
    "Message": "OK",
    "Data": {
      "FailoverMonitors": [
        {
          "Name": "Monitor1",
          "Active": true,
          "Working": true,
          "Status": "WORKING",
          "StatusMessage": "Working..."
        },
        {
          "Name": "Monitor2",
          "Active": false,
          "Working": false,
          "Status": "DISCONNECTED",
          "StatusMessage": "Disconnected..."
        }
      ]
    }
  }
}
```

4.18.3 Activating / Deactivating failover monitors

Allows the activation or deactivation of multiple failover monitors simultaneously.

Compatibility: Enterprise

Security level: Requires user authentication with rights to configure the server

Method: HTTP GET

Syntax:

```
http://<server_address>/Interface/Failover/Activation?<argument=value>
[&<argument=value>...] [&<general\_argument>...]
```

Arguments:

Argument	Valid values	Description										
FailoverMonitors*	String	<p>List of failover monitors to activate or deactivate.</p> <p>The failover monitor names must be separated by commas.</p> <p>This command is only mandatory when Action=Activate or Action=Deactivate</p>										
Action*	Activate Deactivate ActivateAll DeactivateAll	<p>Action to be executed</p> <table border="1"> <thead> <tr> <th>Operation</th><th>Description</th></tr> </thead> <tbody> <tr> <td>Activate</td><td>Activate the monitors on FailoverMonitors argument</td></tr> <tr> <td>Deactivate</td><td>Deactivate the monitors on FailoverMonitors argument</td></tr> <tr> <td>ActivateAll</td><td>Activate all failover monitors from the server</td></tr> <tr> <td>DeactivateAll</td><td>Deactivate all failover monitors from the server</td></tr> </tbody> </table>	Operation	Description	Activate	Activate the monitors on FailoverMonitors argument	Deactivate	Deactivate the monitors on FailoverMonitors argument	ActivateAll	Activate all failover monitors from the server	DeactivateAll	Deactivate all failover monitors from the server
Operation	Description											
Activate	Activate the monitors on FailoverMonitors argument											
Deactivate	Deactivate the monitors on FailoverMonitors argument											
ActivateAll	Activate all failover monitors from the server											
DeactivateAll	Deactivate all failover monitors from the server											

* Mandatory parameters

Example 1: Activate all failover monitors from the server

```
http://192.168.0.1:8601/Interface/Failover/Activation?
Action=ActivateAll
```

Example 2: Activate failover monitors Monitor1 and Monitor2

```
http://192.168.0.1:8601/Interface/Failover/Activation?
Action=Activate&FailoverMonitors=Monitor1,Monitor2
```

Example 3: Deactivate all failover monitors from the server

```
http://192.168.0.1:8601/Interface/Failover/Activation?
Action=DeactivateAll
```

Response:

Default response of API.

HTTP Return HTTP: 200 OK

Parameters of return: [Default return of API](#)

4.19 Device Models

4.19.1 Media Devices

4.19.1.1 Requesting the list of supported media devices

Requests the list of supported media device models from the system

Compatibility: All editions

Security level: Requires user authentication

Method: HTTP GET

Syntax:

```
http://<svr_addr>/Interface/DeviceModels/MediaDevices/GetMediaDeviceModels  
[?<argument=value>[&<argument=value>...][&<general\_argument>...]]
```

Arguments:

Argument	Valid values	Description												
DeviceTypes	MEDIA_DEVICE IP_CAMERA VIDEO_SERVER DVR NVR	<p>Specifies the type of devices that should be returned in the list. In case this parameter is omitted, all device types will be sent.</p> <p>The devices must be separated by comma</p> <table border="1"> <thead> <tr> <th>Name</th><th>Description</th></tr> </thead> <tbody> <tr> <td>MEDIA_DEVICE</td><td>Generic media device that does not fit other categories</td></tr> <tr> <td>IP_CAMERA</td><td>IP Camera device</td></tr> <tr> <td>VIDEO_SERVER</td><td>Video Server device</td></tr> <tr> <td>DVR</td><td>DVR device</td></tr> <tr> <td>NVR</td><td>NVR device</td></tr> </tbody> </table>	Name	Description	MEDIA_DEVICE	Generic media device that does not fit other categories	IP_CAMERA	IP Camera device	VIDEO_SERVER	Video Server device	DVR	DVR device	NVR	NVR device
Name	Description													
MEDIA_DEVICE	Generic media device that does not fit other categories													
IP_CAMERA	IP Camera device													
VIDEO_SERVER	Video Server device													
DVR	DVR device													
NVR	NVR device													
Manufacturers	String	<p>Filter the resulting list to include only the specified manufacturers.</p> <p>Each manufacturer name must be separated by comma</p>												
Fields	Name Manufacturer DeviceType Firmwares	<p>Specifies the list of desired fields. In case this parameter is omitted, all of the fields will be sent.</p> <p>The fields must be separated by commas</p> <table border="1"> <thead> <tr> <th>Name</th><th>Description</th></tr> </thead> <tbody> <tr> <td>Name</td><td>Name of the device model</td></tr> <tr> <td>Manufacturer</td><td>Name of the manufacturer</td></tr> <tr> <td>DeviceType</td><td>Type of device</td></tr> </tbody> </table> <p>Supported values: MEDIA_DEVICE IP_CAMERA VIDEO_SERVER DVR NVR</p>	Name	Description	Name	Name of the device model	Manufacturer	Name of the manufacturer	DeviceType	Type of device				
Name	Description													
Name	Name of the device model													
Manufacturer	Name of the manufacturer													
DeviceType	Type of device													

		Firmwares	Include supported firmware features
--	--	-----------	-------------------------------------

Example 1: Requests the list of media device model database with all of the fields and response in XML
 http://<srv_addr>/Interface/DeviceModels/MediaDevices/GetMediaDeviceModels?
 ResponseFormat=XML

Example 2: Requests the list of media device models with all of the fields and response in text
 http://<srv_addr>/Interface/DeviceModels/MediaDevices/GetMediaDeviceModels?
 ResponseFormat=Text

Example 3: Requests the list of media device models with only name and manufacturer, response in XML and authentication of the user Admin

http://<srv_addr>/Interface/DeviceModels/MediaDevices/GetMediaDeviceModels?
 Fields=Name,Manufacturer&ResponseFormat=XML&AuthUser=admin

Response:

A list with media devices model data is returned. The fields returned in the will depend on the values informed in the argument Fields

HTTP Return: 200 OK

Parameters of return:

Fixed Parameter:

Parameter	Type	Description
COUNT	Integer	Total number of media device models (Not included in JSON response)

Parameters in the list of media device model:

Parameter	Type	Description												
NAME	String	Name of the media device model												
MANUFACTURER	String	Name of the manufacturer of the media device model												
DEVICETYPE	String	Type of device <table border="1" data-bbox="758 1362 1395 1573"> <thead> <tr> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>MEDIA_DEVICE</td> <td>Media Device</td> </tr> <tr> <td>IP_CAMERA</td> <td>IP Camera</td> </tr> <tr> <td>VIDEO_SERVER</td> <td>Video Server</td> </tr> <tr> <td>NVR</td> <td>Network Video Recorder (NVR)</td> </tr> <tr> <td>DVR</td> <td>Digital Video Recorder (DVR)</td> </tr> </tbody> </table>	Type	Description	MEDIA_DEVICE	Media Device	IP_CAMERA	IP Camera	VIDEO_SERVER	Video Server	NVR	Network Video Recorder (NVR)	DVR	Digital Video Recorder (DVR)
Type	Description													
MEDIA_DEVICE	Media Device													
IP_CAMERA	IP Camera													
VIDEO_SERVER	Video Server													
NVR	Network Video Recorder (NVR)													
DVR	Digital Video Recorder (DVR)													
FIRMWARES	FirmwareList	List with supported firmwares and firmware information												

Fixed parameters of firmware list:

Parameter	Type	Description
COUNT	Integer	Total number of firmwares (Not included in JSON response)

Parameters in the list of firmwares:

Parameter	Type	Description
VERSION	String	Firmware version identification

Parameter	Type	Description						
VIDEOINPUTPORTS	Integer	Number of video input ports						
AUDIOINPUTSUPPORT	TRUE FALSE	Support for audio input						
AUDIOINPUTSUPPORTTYPE	NATIVE GENERIC NONE	If audio input is supported, this field will be included in the result and will contain the type of audio input support <table border="1" data-bbox="897 475 1436 623"> <thead> <tr> <th>Type</th><th>Description</th></tr> </thead> <tbody> <tr> <td>NATIVE</td><td>Has native developed audio input driver</td></tr> <tr> <td>GENERIC</td><td>Uses generic RTSP driver</td></tr> </tbody> </table>	Type	Description	NATIVE	Has native developed audio input driver	GENERIC	Uses generic RTSP driver
Type	Description							
NATIVE	Has native developed audio input driver							
GENERIC	Uses generic RTSP driver							
AUDIOINPUTDRIVER	String	If audio input is supported with a native driver, this field will be included in the result and will contain the internal identification of the audio input driver assigned to the firmware						
AUDIOOUTPUTSUPPORT	TRUE FALSE	Support for audio output						
AUDIOOUTPUTDRIVER	String	If audio output is supported, this field will be included in the result and will contain the internal identification of the audio output driver assigned to the firmware						
IOINPUTSUPPORT	TRUE FALSE	Support for I/O input						
IOOUTPUTSUPPORT	TRUE FALSE	Support for I/O output						
IODRIVER	String	If either I/O Input or Output is supported, this field will be included in the result and will contain the internal identification of the I/O driver assigned to the firmware						
IOINPUTPORTS	Integer	If either I/O Input or Output is supported, this field will be included in the result and will contain the number of I/O Input ports						
IOOUTPUTPORTS	Integer	If either I/O Input or Output is supported, this field will be included in the result and will contain the number of I/O Output ports						
DEWARPSUPPORT	TRUE FALSE	Support for panoramic image dewarping						
EDGERECORDINGSUPPORT	TRUE FALSE	Support for Edge Recording						
EDGERECORDINGDRIVER	String	If edge recording is supported, this field will be included in the result and will contain the internal identification of the edge recording driver assigned to the firmware						
EDGEANALYTICSSUPPORT	TRUE FALSE	Support for Edge Analytics						
EDGEANALYTICSDRIVER	String	If edge analytics is supported, this field will be included in the result and will contain the internal identification of the edge analytics driver assigned to the firmware						
EDGELPRSUPPORT	TRUE FALSE	Support for Edge LPR						

Parameter	Type	Description															
EDGELPRDRIVER	String	If edge LPR is supported, this field will be included in the result and will contain the internal identification of the edge LPR driver assigned to the firmware															
DEVICEEVENTSSUPPORT	TRUE FALSE	Support for Device Events															
DEVICEEVENTSDRIVER	String	If device events is supported, this field will be included in the result and will contain the internal identification of the device events driver assigned to the firmware															
DEVICEEVENTSSUPPORTED	String	If device supports events, this field will be included in the result and will contain a comma-separated list of device supported event types															
DEVICESETTINGSSUPPORT	TRUE FALSE	Support for Advanced Device Settings															
DEVICESETTINGSDRIVER	String	If device supports advanced settings, this field will be included in the result and will contain the internal identification of the advanced device settings driver assigned to the firmware															
PTZSUPPORT	TRUE FALSE	Support for PTZ															
PTZDRIVER	String	If device supports PTZ, this field will be included in the result and will contain the internal identification of the PTZ driver assigned to the firmware															
PTZDEVICE	TRUE FALSE	Firmware supports for native PTZ commands															
PTZRELAY	TRUE FALSE	Firmware supports sending analog type PTZ commands through device serial port															
PTZSUPPORTCOMMANDS	String	If device supports PTZ, this field will be included and it contains a comma-separated list of supported PTZ commands:															
<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr style="background-color: #0070C0; color: white;"> <th>Supported PTZ Commands</th> </tr> </thead> <tbody> <tr><td>SIMPLE</td></tr> <tr><td>RELATIVE</td></tr> <tr><td>ABSOLUTE</td></tr> <tr><td>AREA_ZOOM</td></tr> <tr><td>SIMULTANEOUS</td></tr> <tr><td>CONTINUOUS</td></tr> <tr><td>AUTO_FOCUS</td></tr> <tr><td>AUTO_IRIS</td></tr> <tr><td>MENU_CONTROL</td></tr> <tr><td>CALL_PATTERN</td></tr> <tr><td>CALL_PRESET</td></tr> <tr><td>WIPER</td></tr> <tr><td>AUXILIARY</td></tr> <tr><td>POSITION_MONITORING</td></tr> </tbody> </table>			Supported PTZ Commands	SIMPLE	RELATIVE	ABSOLUTE	AREA_ZOOM	SIMULTANEOUS	CONTINUOUS	AUTO_FOCUS	AUTO_IRIS	MENU_CONTROL	CALL_PATTERN	CALL_PRESET	WIPER	AUXILIARY	POSITION_MONITORING
Supported PTZ Commands																	
SIMPLE																	
RELATIVE																	
ABSOLUTE																	
AREA_ZOOM																	
SIMULTANEOUS																	
CONTINUOUS																	
AUTO_FOCUS																	
AUTO_IRIS																	
MENU_CONTROL																	
CALL_PATTERN																	
CALL_PRESET																	
WIPER																	
AUXILIARY																	
POSITION_MONITORING																	
VIDEOCOMPRESSIONS	VideoCompressionList	List of video compression															

Fixed parameters of video compression list:

Parameter	Type	Description
COUNT	Integer	Total number of supported video compressions (Not included in JSON response)

Parameters in the list of video compressions:

Parameter	Type	Description
TYPE	Motion JPEG JPEG 2000 MPEG-4 MxPEG H.263 H.264 H.265	Identifies the type of video compression
VIDEOPROFILE	String	Internal identifier for video profile used by the compression
VIDEODRIVER	String	Internal identifier for video driver assigned to this compression

4.19.1.2 Requesting the data of a specific media device model

Requests the data of a specific supported media device model

Compatibility: All editions

Security level: Requires user authentication

Method: HTTP GET

Syntax:

```
http://<svr_addr>/Interface/DeviceModels/MediaDevices/GetMediaDeviceData
[?<argument=value> [&<argument=value>... ] [&<general\_argument>... ] ]
```

Arguments:

Argument	Valid values	Description												
DeviceName	String	Name of the media device model to query												
Fields	Name Manufacturer DeviceType Firmwares	<p>Specifies the list of desired fields. In case this parameter is omitted, all of the fields will be sent.</p> <p>The fields must be separated by commas</p> <table border="1"> <thead> <tr> <th>Name</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>Name</td> <td>Name of the device model</td> </tr> <tr> <td>Manufacturer</td> <td>Name of the manufacturer</td> </tr> <tr> <td>DeviceType</td> <td>Type of device</td> </tr> <tr> <td colspan="2">Supported values: MEDIA_DEVICE IP_CAMERA VIDEO_SERVER DVR NVR</td></tr> <tr> <td>Firmwares</td> <td>Include supported firmware</td> </tr> </tbody> </table>	Name	Description	Name	Name of the device model	Manufacturer	Name of the manufacturer	DeviceType	Type of device	Supported values: MEDIA_DEVICE IP_CAMERA VIDEO_SERVER DVR NVR		Firmwares	Include supported firmware
Name	Description													
Name	Name of the device model													
Manufacturer	Name of the manufacturer													
DeviceType	Type of device													
Supported values: MEDIA_DEVICE IP_CAMERA VIDEO_SERVER DVR NVR														
Firmwares	Include supported firmware													

		features
--	--	----------

Example 1: Requests the data of media device model "3S Security N5014C" with all of the fields and response in XML

```
http://<srv_addr>/Interface/DeviceModels/MediaDevices/GetMediaDeviceData?
DeviceName=3S Security N5014C&ResponseFormat=XML
```

Example 2: Requests the data of media device model "3S Security N5014C" with all of the fields and response in text

```
http://<srv_addr>/Interface/DeviceModels/MediaDevices/GetMediaDeviceData?
DeviceName=3S Security N5014C&ResponseFormat=Text
```

Response:

The data of the specified media device model is returned. The fields returned in the will depend on the values informed in the argument `Fields`

HTTP Return: 200 OK

Parameters of return:

Fixed Parameters:

Parameter	Type	Description												
NAME	String	Name of the media device model												
MANUFACTURER	String	Name of the manufacturer of the media device model												
DEVICETYPE	String	Type of device <table border="1" data-bbox="758 1098 1395 1298"> <thead> <tr> <th>Type</th><th>Description</th></tr> </thead> <tbody> <tr> <td>MEDIA_DEVICE</td><td>Media Device</td></tr> <tr> <td>IP_CAMERA</td><td>IP Camera</td></tr> <tr> <td>VIDEO_SERVER</td><td>Video Server</td></tr> <tr> <td>NVR</td><td>Network Video Recorder (NVR)</td></tr> <tr> <td>DVR</td><td>Digital Video Recorder (DVR)</td></tr> </tbody> </table>	Type	Description	MEDIA_DEVICE	Media Device	IP_CAMERA	IP Camera	VIDEO_SERVER	Video Server	NVR	Network Video Recorder (NVR)	DVR	Digital Video Recorder (DVR)
Type	Description													
MEDIA_DEVICE	Media Device													
IP_CAMERA	IP Camera													
VIDEO_SERVER	Video Server													
NVR	Network Video Recorder (NVR)													
DVR	Digital Video Recorder (DVR)													
FIRMWARES	FirmwareList	List with supported firmwares and firmware information												

Fixed parameters of firmware list:

Parameter	Type	Description
COUNT	Integer	Total number of firmwares (Not included in JSON response)

Parameters in the list of firmwares:

Parameter	Type	Description		
VERSION	String	Firmware version identification		
VIDEOINPUTPORTS	Integer	Number of video input ports		
AUDIOINPUTSUPPORT	TRUE FALSE	Support for audio input		
AUDIOINPUTSUPPORTTYPE	NATIVE GENERIC NONE	If audio input is supported, this field will be included in the result and will contain the type of audio input support <table border="1" data-bbox="905 1858 1428 1911"> <thead> <tr> <th>Type</th><th>Description</th></tr> </thead> </table>	Type	Description
Type	Description			

Parameter	Type	Description	
		NATIVE	Has native developed audio input driver
		GENERIC	Uses generic RTSP driver
AUDIOINPUTDRIVER	String	If audio input is supported with a native driver, this field will be included in the result and will contain the internal identification of the audio input driver assigned to the firmware	
AUDIOOUTPUTSUPPORT	TRUE FALSE	Support for audio output	
AUDIOOUTPUTDRIVER	String	If audio output is supported, this field will be included in the result and will contain the internal identification of the audio output driver assigned to the firmware	
IOINPUTSUPPORT	TRUE FALSE	Support for I/O input	
IOOUTPUTSUPPORT	TRUE FALSE	Support for I/O output	
IODRIVER	String	If either I/O Input or Output is supported, this field will be included in the result and will contain the internal identification of the I/O driver assigned to the firmware	
IOINPUTPORTS	Integer	If either I/O Input or Output is supported, this field will be included in the result and will contain the number of I/O Input ports	
IOOUTPUTPORTS	Integer	If either I/O Input or Output is supported, this field will be included in the result and will contain the number of I/O Output ports	
DEWARPSUPPORT	TRUE FALSE	Support for panoramic image dewarping	
EDGERECORDINGSUPPORT	TRUE FALSE	Support for Edge Recording	
EDGERECORDINGDRIVER	String	If edge recording is supported, this field will be included in the result and will contain the internal identification of the edge recording driver assigned to the firmware	
EDGEANALYTICSSUPPORT	TRUE FALSE	Support for Edge Analytics	
EDGEANALYTICSDRIVER	String	If edge analytics is supported, this field will be included in the result and will contain the internal identification of the edge analytics driver assigned to the firmware	
EDGELPRSUPPORT	TRUE FALSE	Support for Edge LPR	
EDGELPRDRIVER	String	If edge LPR is supported, this field will be included in the result and will contain the internal identification of the edge LPR driver assigned to the firmware	
DEVICEEVENTSSUPPORT	TRUE FALSE	Support for Device Events	
DEVICEEVENTSDRIVER	String	If device events is supported, this field will be included in the result and will contain the internal	

Parameter	Type	Description															
		identification of the device events driver assigned to the firmware															
DEVICEEVENTSSUPPORTED	String	If device supports events, this field will be included in the result and will contain a comma-separated list of device supported event types															
DEVICESETTINGSSUPPORT	TRUE FALSE	Support for Advanced Device Settings															
DEVICESETTINGSDRIVER	String	If device supports advanced settings, this field will be included in the result and will contain the internal identification of the advanced device settings driver assigned to the firmware															
PTZSUPPORT	TRUE FALSE	Support for PTZ															
PTZDRIVER	String	If device supports PTZ, this field will be included in the result and will contain the internal identification of the PTZ driver assigned to the firmware															
PTZDEVICE	TRUE FALSE	Firmware supports for native PTZ commands															
PTZRELAY	TRUE FALSE	Firmware supports sending analog type PTZ commands through device serial port															
PTZSUPPORTCOMMANDS	String	If device supports PTZ, this field will be included and it contains a comma-separated list of supported PTZ commands:															
		<table border="1"> <thead> <tr> <th>Supported PTZ Commands</th> </tr> </thead> <tbody> <tr><td>SIMPLE</td></tr> <tr><td>RELATIVE</td></tr> <tr><td>ABSOLUTE</td></tr> <tr><td>AREA_ZOOM</td></tr> <tr><td>SIMULTANEOUS</td></tr> <tr><td>CONTINUOUS</td></tr> <tr><td>AUTO_FOCUS</td></tr> <tr><td>AUTO_IRIS</td></tr> <tr><td>MENU_CONTROL</td></tr> <tr><td>CALL_PATTERN</td></tr> <tr><td>CALL_PRESET</td></tr> <tr><td>WIPER</td></tr> <tr><td>AUXILIARY</td></tr> <tr><td>POSITION_MONITORING</td></tr> </tbody> </table>	Supported PTZ Commands	SIMPLE	RELATIVE	ABSOLUTE	AREA_ZOOM	SIMULTANEOUS	CONTINUOUS	AUTO_FOCUS	AUTO_IRIS	MENU_CONTROL	CALL_PATTERN	CALL_PRESET	WIPER	AUXILIARY	POSITION_MONITORING
Supported PTZ Commands																	
SIMPLE																	
RELATIVE																	
ABSOLUTE																	
AREA_ZOOM																	
SIMULTANEOUS																	
CONTINUOUS																	
AUTO_FOCUS																	
AUTO_IRIS																	
MENU_CONTROL																	
CALL_PATTERN																	
CALL_PRESET																	
WIPER																	
AUXILIARY																	
POSITION_MONITORING																	
VIDEOCOMPRESSIONS	VideoCompressionList	List of video compression															

Fixed parameters of video compression list:

Parameter	Type	Description
COUNT	Integer	Total number of supported video compressions (Not included in JSON response)

Parameters in the list of video compressions:

Parameter	Type	Description
TYPE	Motion JPEG JPEG 2000 MPEG-4 MxPEG H.263 H.264 H.265	Identifies the type of video compression
VIDEOPROFILE	String	Internal identifier for video profile used by the compression
VIDEODRIVER	String	Internal identifier for video driver assigned to this compression

4.19.2 I/O Devices

4.19.2.1 Requesting the list of supported I/O devices

Requests the list of supported I/O device models from the system

Compatibility: All editions

Security level: Requires user authentication

Method: HTTP GET

Syntax:

```
http://<svr_addr>/Interface/DeviceModels/IODevices/GetIODeviceModels
[?<argument=value> [&<argument=value>...]] [&<general\_argument>...]]
```

Arguments:

Argument	Valid values	Description								
Manufacturers	String	<p>Filter the resulting list to include only the specified manufacturers.</p> <p>Each manufacturer name must be separated by comma</p>								
Fields	Name Manufacturer Firmwares	<p>Specifies the list of desired fields. In case this parameter is omitted, all of the fields will be sent.</p> <p>The fields must be separated by commas</p> <table border="1"> <thead> <tr> <th>Name</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>Name</td> <td>Name of the device model</td> </tr> <tr> <td>Manufacturer</td> <td>Name of the manufacturer</td> </tr> <tr> <td>Firmwares</td> <td>Include supported firmware features</td> </tr> </tbody> </table>	Name	Description	Name	Name of the device model	Manufacturer	Name of the manufacturer	Firmwares	Include supported firmware features
Name	Description									
Name	Name of the device model									
Manufacturer	Name of the manufacturer									
Firmwares	Include supported firmware features									

Example 1: Requests the list of I/O device model database with all of the fields and response in XML

```
http://<svr_addr>/Interface/DeviceModels/IODevices/GetIODeviceModels?
ResponseFormat=XML
```

Example 2: Requests the list of I/O device models with all of the fields and response in text

```
http://<srv_addr>/Interface/DeviceModels/IODevices/GetIODeviceModels?
ResponseFormat=Text
```

Example 3: Requests the list of I/O device models with only name and manufacturer, response in XML and authentication of the user Admin

```
http://<srv_addr>/Interface/DeviceModels/IODevices/GetIODeviceModels?
Fields=Name,Manufacturer&ResponseFormat=XML&AuthUser=admin
```

Response:

A list with I/O devices model data is returned. The fields returned in the will depend on the values informed in the argument Fields

HTTP Return: 200 OK

Parameters of return:

Fixed Parameter:

Parameter	Type	Description
COUNT	Integer	Total number of I/O device models (Not included in JSON response)

Parameters in the list of media device model:

Parameter	Type	Description
NAME	String	Name of the media device model
MANUFACTURER	String	Name of the manufacturer of the media device model
FIRMWARES	FirmwareList	List with supported firmwares and firmware information

Fixed parameters of firmware list:

Parameter	Type	Description
COUNT	Integer	Total number of firmwares (Not included in JSON response)

Parameters in the list of firmwares:

Parameter	Type	Description
VERSION	String	Firmware version identification
IODRIVER	String	Internal identification of the I/O driver assigned to the firmware
IOINPUTPORTS	Integer	Number of I/O Input ports
IOOUTPUTPORTS	Integer	Number of I/O Output ports

4.19.2.2 Requesting the data of a specific I/O device

Requests the data of a specific supported media device model

Compatibility: All editions

Security level: Requires user authentication

Method: HTTP GET

Syntax:

```
http://<svr_addr>/Interface/DeviceModels/IODevices/GetIODeviceData
[?<argument=value> [&<argument=value>...]] [&<general\_argument>...]]
```

Arguments:

Argument	Valid values	Description
DeviceName	String	Name of the media device model to query
Fields	Name Manufacturer Firmwares	Specifies the list of desired fields. In case this parameter is omitted, all of the fields will be sent. The fields must be separated by commas

Name	Description
Name	Name of the device model
Manufacturer	Name of the manufacturer
Firmwares	Include supported firmware features

Example 1: Requests the data of I/O device model "IP Extreme IPX8001" with all of the fields and response in XML

```
http://<srv_addr>/Interface/DeviceModels/IODevices/GetIODeviceData?
DeviceName=IP Extreme IPX8001&ResponseFormat=XML
```

Example 2: Requests the data of media device model "IP Extreme IPX8001" with all of the fields and response in text

```
http://<srv_addr>/Interface/DeviceModels/IODevices/GetIODeviceData?
DeviceName=IP Extreme IPX8001&ResponseFormat=Text
```

Response:

The data of the specified I/O device model is returned. The fields returned in the will depend on the values informed in the argument Fields

HTTP Return: 200 OK

Parameters of return:**Fixed Parameters:**

Parameter	Type	Description
COUNT	Integer	Total number of I/O device models (Not included in JSON response)

Parameters in the list of media device model:

Parameter	Type	Description
NAME	String	Name of the media device model
MANUFACTURER	String	Name of the manufacturer of the media device model
FIRMWARES	FirmwareList	List with supported firmwares and firmware information

Fixed parameters of firmware list:

Parameter	Type	Description
COUNT	Integer	Total number of firmwares (Not included in JSON response)

Parameters in the list of firmwares:

Parameter	Type	Description
VERSION	String	Firmware version identification
IODRIVER	String	Internal identification of the I/O driver assigned to the firmware

Parameter	Type	Description
IOINPUTPORTS	Integer	Number of I/O Input ports
IOOUTPUTPORTS	Integer	Number of I/O Output ports

4.20 RTSP

4.20.1 Requesting the RTSP server settings

Request the RTSP server settings

Compatibility: All editions

Security level: Requires user authentication

Method: HTTP GET

Syntax:

```
http://<server_address>/Interface/RTSP/GetConfig  
[?<general_argument>[&<general_argument>...]]
```

Example 1: Request the RTSP server settings with response in XML format

```
http://192.168.0.1:8601/Interface/RTSP/GetConfig?ResponseFormat=XML
```

Example 2: Request the RTSP server settings with response in text format and authentication with admin user

```
http://192.168.0.1:8601/Interface/RTSP/GetConfig?  
ResponseFormat=Text&AuthUser=admin
```

Response:

A list of parameter-value pairs is returned

HTTP Return: 200 OK

Parameters of return:

Parameter	Type	Description
ACTIVATE	Boolean	Server activated or deactivated
PORT	Integer	RTSP server port
RTSPS	Boolean	RTSPS activated
RTSPSPORT	Integer	RTSPS server port

Example of return in text:

```
RESPONSE_CODE=0  
RESPONSE_MESSAGE=OK  
ACTIVATE=TRUE  
PORT=554  
RTSPS=TRUE  
RTSPSPORT=322
```

Example of return in XML:

```
<Response>
<Code>0</Code>
<Message>OK</Message>
<Data>
<Config>
<Activate>TRUE</Activate>
<Port>554</Port>
<RTSPS>TRUE</RTSPS>
<RTSPSPort>322</RTSPSPort>
</Config>
</Data>
</Response>
```

Example of return in JSON:

```
{
  "Response": {
    "Code": 0,
    "Message": "OK",
    "Data": {
      "Config": {
        "Activate": true,
        "Port": 554,
        "RTSPS": true,
        "RTSPSPort": 322
      }
    }
  }
}
```

4.20.2 Requesting the status of RTSP server

Request the status of the RTSP server

Compatibility: All editions

Security level: Requires user authentication with rights to monitor the server health and status

Method: HTTP GET

Syntax:

```
http://<server_address>/Interface/RTSP/GetStatus
[?<general_argument>[&<general_argument>...]]
```

Example 1: Request the status of RTSP server with response in XML format

```
http://192.168.0.1:8601/Interface/RTSP/GetStatus?ResponseFormat=XML
```

Example 2: Request the status of RTSP server with response in text format and authentication with admin user

```
http://192.168.0.1:8601/Interface/RTSP/GetStatus?
ResponseFormat=Text&AuthUser=admin
```

Response:

A list of parameter-value pairs is returned

HTTP Return: 200 OK

Parameters of return:

Parameters	Type	Description
ACTIVE	Boolean	RTSP server is active or not
PORT	Integer	RTSP server port
CONNECTIONS	Integer	Total connections with the server
USERS	Integer	Total authenticated connections with the server
OUTPUTTRAFFIC	Integer	RTSP server output traffic measured in bytes per second

Example of return in text:

```
RESPONSE_CODE=0
RESPONSE_MESSAGE=OK
ACTIVE=TRUE
PORT=554
CONNECTIONS=10
USERS=10
OUTPUTTRAFFIC=1729405
```

Example of return in XML:

```
<Response>
  <Code>0</Code>
  <Message>OK</Message>
  <Data>
    <Status>
      <Active>TRUE</Active>
      <Port>554</Port>
      <Connections>10</Connections>
      <Users>10</Users>
      <OutputTraffic>1729405</OutputTraffic>
    </Status>
  </Data>
</Response>
```

Example of return in JSON:

```
{
  "Response": {
    "Code": 0,
    "Message": "OK",
    "Data": {
      "Status": {
        "Active": true,
        "Port": 554,
        "Connections": 10,
        "Users": 10,
        "OutputTraffic": 1729405
      }
    }
  }
}
```