High Performance Network Security

FortiRecorder™ REST API Reference

Version 7.2.3

FORTINET DOCUMENT LIBRARY

https://docs.fortinet.com

FORTINET VIDEO GUIDE

https://video.fortinet.com

FORTINET BLOG

https://blog.fortinet.com

CUSTOMER SERVICE & SUPPORT

https://support.fortinet.com

FORTINET TRAINING SERVICES

https://www.fortinet.com/training

FORTIGUARD CENTER

https://www.fortiguard.com

END USER LICENSE AGREEMENT

https://www.fortinet.com/doc/legal/EULA.pdf

FEEDBACK

Email: techdocs@fortinet.com

May 21, 2025

TABLE OF CONTENTS

Introduction

This document provides information about the REST APIs on FortiRecorder:

- **Video clip service**
  This API allows access to recorded video clips and snapshots from recordings.
- **FortiRecorder system-level resources**
  These APIs can be used to retrieve, create, update and delete configuration settings, to retrieve dynamic system statistics, and to perform basic administrative actions such as reboot and shut down.
- **Face recognition**
  These APIs provides eservices related to the built-in face recognition of FortiRecorder. Event logs and the user database can be accessed.

HTTP methods and response codes

FortiRecorder REST APIs use the following HTTP method conventions:

HTTP Method	Usage
HTTP GET	Retrieve all resources or a specific resource
HTTP POST	Create a new resource or perform certain administrative actions
HTTP PUT	Update an existing resource
HTTP DELETE	Delete an existing resource

FortiRecorder REST APIs use the following well-defined HTTP status codes to indicate query results:

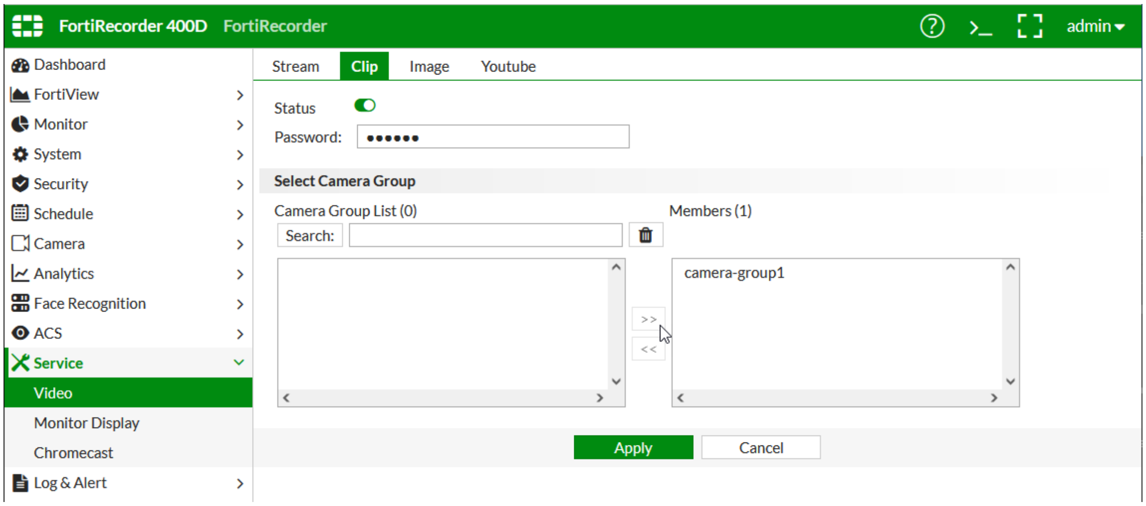
HTTP Response Code	Description
200 - OK	Successful request
400 - Bad Request	Bad request
403 - Forbidden	Request is missing authentication token or administrator does not have the required permissions in their access profile
404 - Not Found	Resource not found
405 - Method Not Allowed	Specified HTTP method is not allowed for this resource
500 – Server Error	Internal server error

Video clip service

This API provides system integrators with the ability to retrieve recordings either as a video clip or still image snapshot.

Enabling the video clip service

1. On FortiRecorder, go to *Service > Video > Clip.*
2. Enter a password.
3. Select a group of cameras that should be accessible through this service.
4. Click *Apply*.



Status ●

Password: ••••••

Camera Group List (0)

Members (1)

camera-group1

Authentication

To establish a valid authentication session, you must make an HTTP POST request to the FortiRecorder login handler with the name `service-clip` and the password defined in *Service > Video > Clip*. The request should contain JSON data with `name` and `password` fields:

URL: `http(s)://host_or_ip/api/v1/ServiceLogin/`

Method: `POST`

JSON:

```
{"reqAction":1,
 "name":"service-clip",
 "password":"****"}
```

If login is successful, the response will contain the authentication token in the `APSCOOKIE` cookie value. This cookie must be included in any further requests.

Example

```
curl -k -v -c cookiefile -X POST -d "{\"reqAction\":1,\"name\":\"service-
clip\",\"password\":\"1234\"}" -H "Content-Type:application/json" https://
```

```
ip_or_host /api/v1/ServiceLogin/
```

REST API reference

VideoClip

Note: Most parameters are optional. If a request does not include them, then the video is downloaded as-is. Do not include optional parameters unless you want FortiRecorder to convert the video to a different size or quality, omit audio, etc. Conversion is best effort. It may not be possible in all cases to achieve a video clip with those characteristics. Time required increases if FortiRecorder converts the video.

URL: `http(s)://host_ip/api/v1/VideoClip/`

Method: `POST`

JSON:

```
{"reqAction":21,
 "camera":"camera_name",
 "begin":1584734700,
 "end":1584734900,
 "audio":0,
 "codec":"h265",
 "fps":30,
 "gop":60,
 "qvalue":50,
 "bitrate":1000,
 "height":1440,
 "width":2560
 "osdFormat":"24h",
 "osdPosition":"Top-Left",}
```

where:

audio: 0: do not include audio
1: include audio, if any

begin: required; timestamp for the start of the video clip in UTC

bitrate: the target video quality, in bits per second.
Note: `bitrate` is ignored if `qvalue` is used.

camera: required; camera name as defined on FortiRecorder

codec: h264 or h265

end: required; timestamp for the end of the video clip in UTC.
If `end` is equal to `begin,` then the API call downloads a snapshot in JPG image format. Otherwise it downloads an MP4 video clip.

fps: number of frames per second used to encode the video

gop: number of frames for each reference frame in the group of pictures; default GOP may vary by camera model

height: height in pixels

reqAction: 21 -- required, fixed

qvalue: quality value; valid values are from 10 to 50.
32 is a reasonable starting point. The greater the `qvalue`, the less the bitrate and video quality.
Note: `bitrate` is ignored if `qvalue` is used.

width: width in pixels

osdFortmat: On-Screen Display (OSD) format in 12h or 24h.

osdPosition: OSD positon: Top-Left, Top-Right, Bottom-Left, or Bottom-Right.

Note: Timestamps are in UTC, so the local time on FortiRecorder must be converted to UTC with the right time zone. For example, 1 PM PDT may convert to a timestamp that is equivalent to 8 PM GMT.

Example: Download an MP4 video clip

```
curl -k -v -b cookiefile -o video.mp4 -X POST -d "{\"reqAction\"
:21,\"camera\":\"camname\",\"begin\":1584734700,\"end\":1584734900}" -H
"Content-Type:application/json" https://ip_or_host/api/v1/VideoClip/
```

Example: Download a JPG snapshot image

```
curl -k -v -b cookiefile -o snapshot.jpg -X POST -d "{\"reqAction\"
:21,\"camera\":\"camname\",\"begin\":1584734700,\"end\":1584734700}" -H
"Content-Type:application/json" https://ip_or_host/api/v1/VideoClip/
```

System level resources

FortiRecorder supports retrieval and modification of system level configuration settings as well as system level statistics.

URL

The API can be accessed using the following URL scheme:

```
http(s)://host_ip/api/v1/res_name/res_id/sub_res_name/sub_res_id/
```

where:

`res_name` Specifies the type of resource to query (such as `SysInterface`), required.

`res_id` Unique ID of the resource as specified by res_name (such as port1), optional.
If not present, returns entire list of resources.

`sub_res_name` Some resources may have child resources, use this to query these sub-resources, optional.

`sub_res_id` Unique ID of the sub resource as specified by `sub_res_name`, optional.
If not present, returns entire list of sub resources.

Examples:

…`/api/v1/SysInterface/` --- returns list of network interfaces

…`/api/v1/SysInterface/port1/` --- returns details of network interface ‘port1’

…`/api/v1/SysGlobal/` --- returns details of global settings (only one instance)

Note: The commands are case sensitive.

For a list of frequently used system level resources, see *System resource list and URLs*.

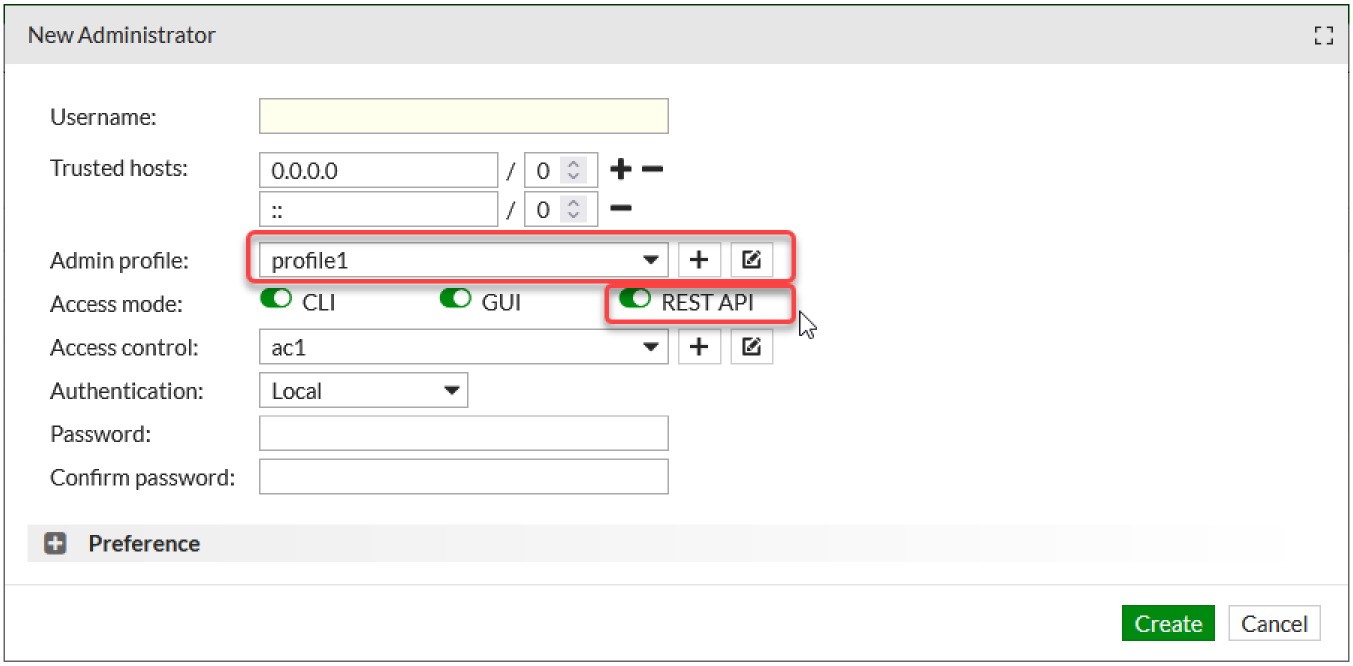
This enables integration tasks like:

- Enumerating cameras to get the available names – CameraStatus
- Editing Camera profiles and Video profiles
- Enumerate events or notifications - Timeline
- System status and resource information - SysStatusUsage
- Camera status information - CameraStatus
- Switching camera profiles - CameraCamera

Authentication

The FortiRecorder REST API for system-level resources requires authentication. Currently only local user password-based authentication is supported.

1. Go to *System > Administrator > Administrator*.
2. Either click *New* or double-click to edit an existing account that will use this REST API.
3. In *Access mode*, enable *REST API*.



New Administrator

Username:

Trusted hosts: 0.0.0.0 / 0
:: / 0

Admin profile: profile1

Access mode: CLI GUI REST API

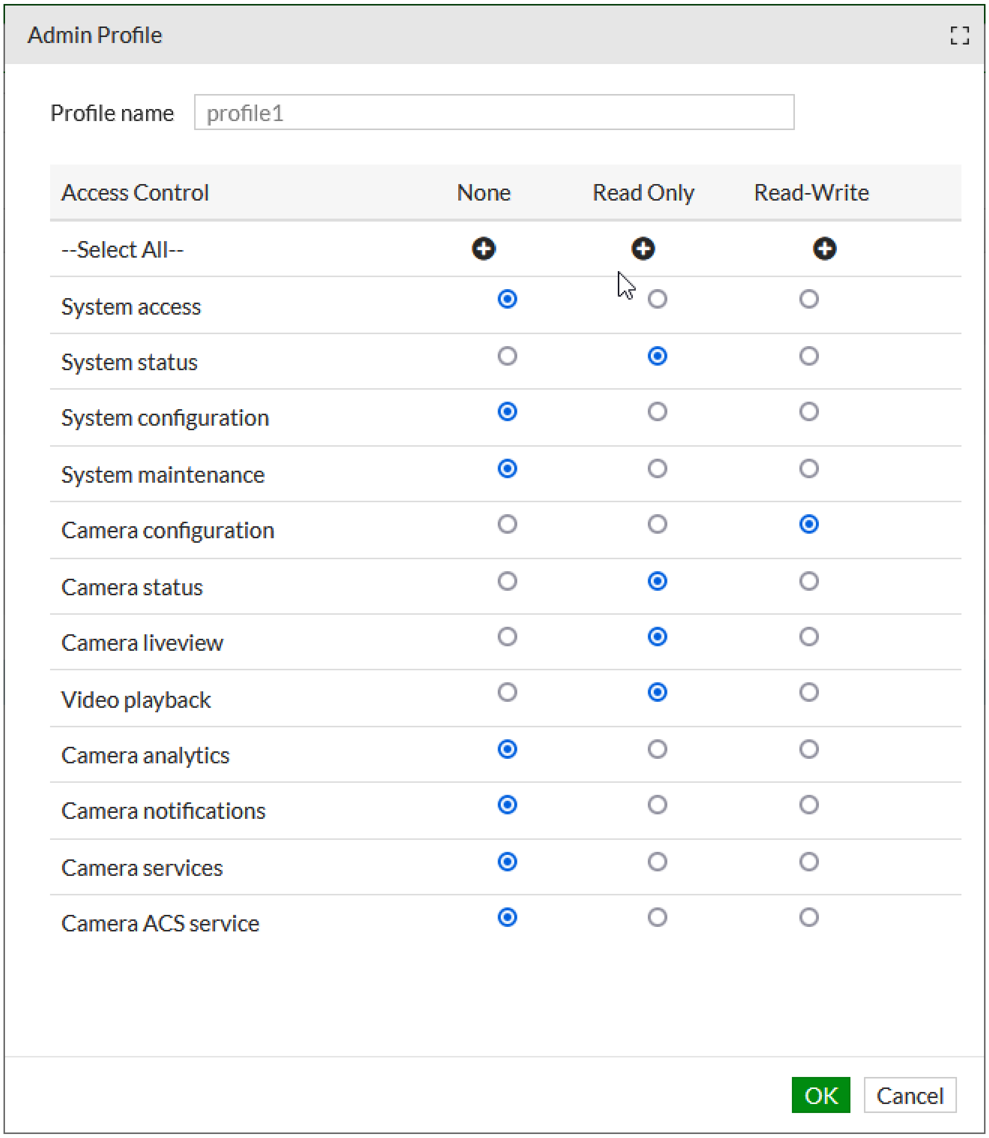
Access control: ac1

Authentication: Local

Password:

Confirm password:

4. In *Admin profile*, grant permissions to access each FortiRecorder resource (see *System resource list and URLs)*. If the account will use the HTTP GET method to access a resource, then at least *Read Only* permission is required. For the other methods, *Read-Write* is required.



Admin Profile

Profile name: profile1

Access Control	None	Read Only	Read-Write
--Select All--	⊕	⊕	⊕
System access	●	○	○
System status	○	●	○
System configuration	●	○	○
System maintenance	●	○	○
Camera configuration	○	○	●
Camera status	○	●	○
Camera liveview	○	●	○
Video playback	○	●	○
Camera analytics	●	○	○
Camera notifications	●	○	○
Camera services	●	○	○
Camera ACS service	●	○	○

5. Click *OK*.
6. Make an HTTP POST request to the FortiRecorder with the name `admin` and its password. The request should contain JSON data with `name` and `password` fields.

URL: `http(s)://host_or_ip/api/v1/AdminLogin/`

Method: `POST`

JSON:
```
{"name":"admin",
 "password":"****"}
```

If login is successful, the response will contain the authentication token in the `APSCOOKIE` cookie value. This cookie must be included in any further requests.

Example

```
curl -v -H "Content-Type: application/json" -X POST -d
'{"name":"admin","password":"*****"}' https://ip_or_host/api/v1/AdminLogin
-c cookie.txt
```

If login is successful, the cookies will be saved to `cookie.txt`, which will be used in later commands.

System resource list and URLs

URL	HTTP Methods	Admin Profile Permissions	Summary
/CameraStatus/	GET	Camera Status Camera Configuration	Camera status
/CameraProfile/	GET, POST, PUT, DELETE	Camera Configuration	Camera profile list
/CameraCamera/	PUT	Camera Configuration	Change camera profile
/ScheduleObject/	PUT	Camera Configuration	Change Schedule
/CameraVideoProfile/	GET, POST, PUT, DELETE	Camera Configuration	Camera video list
/SysInterface/	GET, POST, PUT, DELETE	System Access	Network interface list
/SysGlobal/	GET, PUT	System Access	System global settings
/SysStatusUsage/	GET		System resource usage
/SysStatusSysinfo/	GET	System Status	System status information
/SysStatusCommand/	POST	System Access	Restart / Shut down / Reload system command
/CameraEvent/	GET	Camera Configuration	Timeline and system related events

	enumeration

Supported values for 'reqAction' attribute of all JSON requests

- 1 --- GET
- 2 --- CREATE
- 3 --- DELETE
- 5 --- UPDATE
- 14 --- MOVE

Note: If `reqAction` exists in the JSON, then it takes precedence over HTTP method header (`GET`/`POST`/`PUT`/`DELETE`).

REST API reference

The following is a selection of REST API commands that are frequently useful for system integration. The list of parameters and responses is not exhaustive and other information may be contained in the JSON data.

If a Response value is listed in the following specification, it is for showing a typical format or value and depends on the individual host unless otherwise listed as fixed.

Collections return an enumeration of all the resources.

Sometimes the individual responses are more detailed than the collection entries.

Responses are JSON formatted. Example:

```
{"objectID": "SysInterfaceCollection:","reqAction": 1,"totalRemoteCount":
2,"subCount": 2,"remoteSorting": true,"nextPage": false,"nodePermission":
3,"collection": [

 {"mkey": "port1","type": 0,"aggregate_master": 0,"bridge_member":
true,"ip": "192.168.1.99/24","ip6": "::/0","status": true,"interface":
"","aggregate_member": "","incoming_mode": 2,"outgoing_mode": 0,"local":
true,"allowaccess": 151,"discover": true,"webaccess": 1,"link_status":
true,"isReferenced": 1,"modifyFlag": 1},

{"mkey": "port2","type": 0,"aggregate_master": 0,"bridge_member":
true,"ip": "192.168.2.99/24","ip6": "::/0","status": true,"interface":
"","aggregate_member": "","incoming_mode": 2,"outgoing_mode": 0,"local":
true,"allowaccess": 7,"discover": true,"webaccess": 1,"link_status":
true,"modifyFlag": 1},
]
```

Since collections can be quite large, it is possible to request a subset of the collection using the following parameters

startIndex	Index of the first entry to return, 0 is the first entry.
pageSize	Max number of entries to return.

The response will contain

totalRemoteCount	number of entries in collection
SubCount	number of entries returned

CameraStatus

Enumeration and status of cameras.

URL: `http(s)://host_ip/api/v1/CameraStatus[?showInactiveCamera=0]`

Method: `GET`

Where:

showInactiveCamera:	0: only return active camera 1: return all cameras, active and disabled

Response:

totalRemoteCount	2	number of entries in collection
SubCount	2	number of entries returned

Collection: []

mkey	Name of camera e.g. "Cam01-Door"
status	true: Camera is enabled false: Camera is disabled
action_scheduled	Scheduled actions represented as a bitmask, see action status bitmask section below
action_pending	Pending actions, started but not yet active, represented as a bitmask see action status bitmask section below
action_current	Current actions represented as a bitmask, see action status bitmask section below

action_problem	Incorrect actions represented as a bitmask, see action status bitmask section below
last_query	last time the camera was queried, in UTC.
state_flag	Camera state, see section below.
state_code	Low level error code
state_info	Text message for some error code and state flag value

Action Status Bitmask:

Bit values that are used in the action status

0: Idle
1 << 0: Continuous Recording
1 << 1: Motion Detection
1 << 2: Digital Input
1 << 3: Audio Detection
1 << 4: PIR detection
1 << 5: Tamper detection

1 << 8: Continuous Recording on SD Card
1 << 9: Motion Detection on SD Card
1 << 10: Digital Input on SD Card
1 << 11: Audio Detection on SD Card
1 << 12: PIR detection on SD Card
1 << 13: Tamper detection on SD Card

Camera State:

0: Not supported
1: Active
2: Inactive
3: Camera is not configured
4: Camera is unreachable
5: Camera is not configured and has default address.
6: Camera has an invalid address
7: Camera has default address
8: Camera is being configured
9: Camera has a configuration error
10: Camera is upgrading

11: Camera is rebooting
12: Camera is not configured and has invalid address
13: Duplicate IP, the camera and another device have the same IP
14: Camera is configured by another FortiRecorder

Example: Enumerate all cameras

```
curl -k -v -b cookie.txt https://192.168.1.99/api/v1/ CameraStatus
```

CameraVideoProfile

Enumerate the Video Profiles

URL: `http(s)://host_ip/api/v1/CameraVideoProfile`

Method: `GET`

Response:

totalRemoteCount	2	number of entries in collection
SubCount	2	number of entries returned

Collection: []

mkey	"2MP”	Name of Video Profile
video_resolution	6	Resolution

0: Low
1: Medium
2: High
3: Extra-High
4: 1/2 MP
5: 1 MP
6: 2 MP
7: 3 MP
8: 4 MP
9: 5 MP
10: 6 MP
11: 9 MP

		12: 12 MP
video_codec	0	Codec type 0: Default 3: H.264 4: H.265
video_fps	30	Number of frames per second
video_gop	3	Group of Pictures setting 0: auto 1: ¼ second 2: ½ second 3: 1 second 4: 2 seconds 5: 3 seconds 6: 4 seconds
video_bitrate_mode	0	Bitrate Mode 0: variable 1: Fixed 2: Constrained
audio	false	Is audio enabled

Example: How to modify a Video Profile.

```
http(s)://host_ip/api/v1/CameraVideoProfile/<mkey>
```

Method: `PUT`

Where: <mkey> is the name of video profile to be modified

JSON: See list of valid attributes and values above.

For example, to change the fps to 15

{ “video_fps”: 15}

Camera Profile

Retrieve and edit a Camera profile.

URL: `http(s)://host_ip/api/v1/CameraProfile`

Method: `GET`

Response:

totalRemoteCount	2	number of entries in collection
SubCount	2	number of entries returned

Collection: []

mkey	"HighQualityContAllDet”	Camera profile unique name
continuous_retention_disposition		storage option for continuous recordings 0: keep until overwritten 1: delete 2: move
continuous_retention_period:		type of period for the storage option delete and move 0: days 1: weeks 2: months 3: years 4: hours
continuous_retention_period_units		number of units of given period before deletion or move
continuous_retention_disposition:		storage option for detection recordings 0: keep until overwritten 1: delete 2: move
continuous_retention_period:		type of period for the storage option delete and move 0: days 1: weeks 2: months 3: years 4: hours
continuous_retention_period_units		number of units of given period before deletion or move
compression	true / false	indicates if compression is enabled
compression_period		type of period for compression when enabled 0: days

		1: weeks 2: months 3: years
compression_period_units		Number of units of given period before compression of video files
viewing_stream	“Always:HighRes”	Name of schedule:Name of video profile
recording_stream	“Always:HighRes”	Name of schedule:Name of video profile
recording_type	“Always:17”	Name of schedule:type where type is bit set 1<<0 Store on FortiRecorder 1<<1 Store on camera SD card 1<<4 Continuous 1<<5 Motion Detection 1<<6 DI 1<<7 Audio Detection 1<<8 PIR 1<<9 Tamper Detection

Example: To retrieve a specific profile

```
http(s)://host_ip/api/v1/CameraProfile/<profile name>
```

Example: How Modify the Video Profile of a Camera Profile.

```
http(s)://host_ip/api/v1/CameraProfile/<camera profile mkey>
/CameraProfileVideoSchedule/<schedule mkey>
```

Where: There are 2 mkey in the URL, one is for the camera profile that will be modified. The 2nd one is the schedule name that will be modified.

Method: `PUT`

JSON: Use {"recording_stream":"high-resolution"}

CameraCamera

Modify the active profile for a camera

URL: `http(s)://host_ip/api/v1/CameraCamera/<mkey>`

Method: `PUT`

Where:

mkey	Camera Name

JSON: Example to change the profile of the camera to use the Camera Profile HighResContinuous

{"profile": "HighResContinuous"}

PTZ commands

URL: `http(s)://host_ip/api/v1/CameraCamera/<mkey>`

Method: `POST`

JSON:

```
{“ reqAction”: action_value, “camMove”: move_value,
“ magnitude”: mag_value, “presetName”: preset_value}
```

Where:

mkey	Camera Name
action_value	7
mag_value	speed of movement 1 … 10
preset_value	name of preset (only for move_value 16)
move_value	

0	---	move home
1	---	move right
2	---	move left
3	---	move up
4	---	move down
5	---	move up/right
6	---	move up/left
7	---	move down/right
8	---	move down/left
9	---	zoom in
10	---	zoom out
11	---	set home
12	---	focus in
13	---	focus out
14	---	focus semi-auto
15	---	focus full auto
16	---	move to preset

JSON: Example to move the camera into home position

{"reqAction":7, "camMove":0, "magnitude":4}

Example to move the camera into preset_1

{"reqAction":7, "camMove":16, "presetName:":"preset_1"}

CameraCameraPtzPresets

Add or modify presets for PTZ cameras.

When adding / modifying a preset, position the camera first to the desired position.

URL:

```
http(s)://host_ip/api/v1/CameraCamera/<camera_name>/
CameraCameraPtzPresets/<presetname>
```

Method: `POST`

Where

camera_name: camera name as defined on FortiRecorder

Note: POST for add a preset and PUT for setting preset at current position.

Example: Modify preset for camera

```
curl -k -v -b cookiefile -X PUT -d "{}" -H "Content-Type:application/json"
https://ip_or_host/api/v1/ CameraCamera/<camera_name>/
CameraCameraPtzPresets/<presetname>
```

Delete a preset for PTZ cameras.

URL:

```
http(s)://host_ip/api/v1/CameraCamera/<camera name>/
CameraCameraPtzPresets/<presetname>
```

Method: `DELETE`

Where

camera_name: camera name as defined on FortiRecorder

presetname: preset name as defined on FortiRecorder

List presets for PTZ cameras.

URL:

```
     http(s)://host_ip/api/v1/CameraCamera/<camera name>
/CameraCameraPtzPresets/
```

Method: `GET`

ScheduleObject

Enable or disable Schedules on the recorder. The schedules must be of type Assistant. This can be used in all places that support schedules like enabling and disabling recordings in a Camera Profile under Recording.

URL: `http(s)://host_ip/api/v1/ScheduleObject/<mkey>`

Method: `PUT`

Where: mkey Assistant type Schedule Name (e.g predefined Away)

JSON: Example to activate a schedule.

{"assist_active" : true}

SysInterface

Properties and enumeration of system interfaces (ports).

URL: `http(s)://host_ip/api/v1/SysInterface`

Method: `GET`

Response:

totalRemoteCount	2	number of entries in collection
SubCount	2	number of entries returned

Collection: []

mkey	Name of port e.g. "port1"
type	0: physical 1: vlan 2: aggregate 3: redundant

ip	IP address e.g. "192.168.1.99/24"	
ip6	IPv6 address	
status	true false	interface is enabled interface is disabled
allowaccess	151	
discover	true	camera discovery enabled
webaccess	1	GUI access enabled
link_status	true false	interface is up interface is down
mac_addr	MAC address of interface	

Example: Enumerate all interfaces

```
curl -k -v -b cookie.txt https://192.168.1.99/api/v1/SysInterface
```

Example: Retrieve the port1 interface settings

```
curl -k -v -b cookie.txt https://192.168.1.99/api/v1/SysInterface/port1
```

SysGlobal

IP protocol ports for various services.

URL: `http(s)://host_ip/api/v1/SysGlobal`

Method: `GET`

Response:

hostname		Serial number. Example:FK-SVM0000000000
port_http	80	HTTP port number
port_https	443	HTTPS port number
port_ssh	22	SSH port number
port_telnet	23	Telnet port number
port_frc_central	8550	FortiCentral port number

frc_central_secure		true: force SSL/TLS connection for FortiCentral
port_rtsp	554	RTSP port number
public_address		public IP address as seen from outside gateway
public_https_port	443	public HTTPS port number
public_http_port	80	public HTTP port number
public_rtsp_port	554	public RTSP port number
public_ftp_port	21	public FTP port number
public_frc_central_port	8550	public FortiCentral port number
public_notify_tcp_port	3010	public event notification TCP port number

SysStatusUsage

System status information with current and historical values.

URL: `http(s)://host_ip/api/v1/SysStatusUsage`

Method: `GET`

Response:

hostname		Serial number e.g. FK-SVM0000000000
cpu	7	current CPU usage in %
memory	14	current memory usage in %
log_disk	0	current log disk usage in %
mail_disk	96	current video disk usage in %
remote_video_disk	0	current remote storage usage in %
system_load	6	current indicator for system load including i/o
active_sessions	4	currently active sessions

In addition, there is historical data available to build charts for some of the indicators, e.g:

cpu_history

values	[7, 7, 8, 9, 7, 9, 9, 8, 8, 9, 9, 8, 9, 8, 8, 8, 8, 8, 8, 8, 7]
x_labels	["18:20", ... , "18:38", "18:39", "18:40"]

y_legend	"%"
y_step	10
y_max	100

memory_history

values	[14, 14, 14, 14, 14, … , 14, 14, 14]
x_labels	["18:20", "18:21", … , "18:40"]
y_legend	"%"
y_step	10
y_max	100

session_history

values	[3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 2, 2, 2, 2, 2, 2, 2, 3, 4]
x_labels	["18:20", "18:2", "18:22", … , "18:39", "18:40"]
y_step	10
y_max	10

network_history

values	[6263, 6227, 6213, 6287, … , 6260, 6257, 6239, 6286]
x_labels	["18:20", "18:21", "18:22", … , "18:39",\"18:40"]
y_legend	"Kbps"
y_step	1000
y_max	7000

SysStatusSysinfo

Enumeration and status of cameras.

URL: `http(s)://host_ip/api/v1/SysStatusSysinfo`

Method: `GET`

Response:

serial_number		e.g. "FK-SVM0000000000"
up_time		time since startup in days/hours/min/s e.g. "0 4 51 56"
system_time		UTC system time e.g. 1594259377
firmware_version		e.g. "v6.0.2, build124, 2020.05.12"
current_admin		currently logged in user name e.g. "admin"
admin_num	1	number of logged in administrators

log_disk_info		e.g. "Capacity 1475 MB, Used 4 MB (0.32%), Free 1471 MB",
log_disk_capacity	1475	log disk capacity in MB
log_disk_used	4	log disk usage in MB
log_disk_status		0: available 1: not available
log_storage_status		0: Ok 1: Not mounted 2: inaccessible 3: unwriteable
mailbox_disk_info		video disk information e.g. "Capacity 27 GB, Used 26 GB (94.29%), Free 1632 MB",
mailbox_disk_capacity		video disk capacity in MB e.g. 28590
mailbox_disk_used		video disk usage in MB e.g 26957
mailbox_disk_status		0: available 1: not available
local_storage_status		0: Ok 1: Not mounted 2: inaccessible 3: unwriteable
remote_storage_status		0: Ok 1: Not mounted 2: inaccessible 3: unwriteable
actual_retention_local	175103	Number of seconds spanning video files on local storage
actual_retention_remote	5103	Number of seconds spanning video files on remote storage
retention_status		0: calculation completed 1: calculation in progress
remote_video_disk_info		remote storage disk info e.g. "n/a", see `mailbox_disk_info`
remote_video_disk_status		0: available 1: not available

SysStatusCommand – Administrative actions

Apart from resources, FortiRecorder REST API supports basic administrative actions such as restarting /

shutting down a device. Use the following URL to send action request:

URL: `http(s)://host_ip/api/v1/SysStatusCommand`

Method: `POST`

JSON: `{“action”: action_value}`

Where action_value is one of the following integers:

1 --- Restart

2 --- Shut down

3 --- Reload

CameraEvent

Enumeration of camera and system related events.

URL:

```
http(s)://host_ip/api/v1/CameraEvent?device_name=<name>&
start_time=<start_time>&end_time=<end_time>
[&evt_filter=filter][&startIndex=<start>&pageSize=<num>]
```

Method: `GET`

Where:

device_name: required, comma separated list of camera names

start_time: required, start time in the format YYYY-MM-DD-HH-MM-SS

end_time: required, end time in the format YYYY-MM-DD-HH-MM-SS

filter: optional value, see event type section in document.

startIndex: optional, used to retrieve a subset of the total list of events

pageSize: optional, max number of events to return

Response:

totalRemoteCount number of entries in collection

SubCount number of entries returned

start_time: start time in the format YYYY-MM-DD-HH-MM-SS

end_time:	end time in the format YYYY-MM-DD-HH-MM-SS

Collection: []

mkey:	unique event key
device_name:	camera name
start_time:	Start of event in format: YYYY-MM-DD HH:MM:SS
end_time:	End of event in format: YYYY-MM-DD HH:MM:SS

type:	see Event type
subtype:	see Event subtype
state:	see Event state

Event Type:

Events are coded as event and subtype. Only one bit is active, but when used as filter multiple bits can be set.

0:	Evt_None	
1 << 0:	Evt_Detect_Generic	
1 <<1:	Evt_Detect_Motion	See subtype Motion
1 <<2:	Evt_Detect_Audio	
1 <<3:	Evt_Detect_DI	
1 <<4:	Evt_Detect_PIR	
1 <<5:	Evt_Detect_Tamper	See subtype Tamper
1 <<6:	Evt_Detect_Face_Detection	See subtype Face Detection
1 <<7:	Evt_Detect_Physical_Access	
1 <<8:	Evt_Detect_Object_Detection	
1 <<16:	Evt_Camera	See subtype Camera
1 <<17:	Evt_Recording	See subtype Recording
1 <<18:	Evt_Schedule	See subtype Camera Event
1 <<19:	Evt_Annotate	
1 <<20:	Evt_System	See subtype System Event
1 <<21:	Evt_Notification	

Event Subtype:

The sub-type is in the context of the event type. Only one bit is active.

Motion

0:	SubEvt_Motion_None
1 <<0:	SubEvt_Motion_Motion
1 <<1:	SubEvt_Motion_MotionAlarm
1 <<2:	SubEvt_Motion_ObjectInside
1 <<3:	SubEvt_Motion_Crossed

Tamper

0:	SubEvt_Tamper_None = 0,
1 <<0:	SubEvt_Tamper_Realtime
1 <<1:	SubEvt_Tamper_Tamper
1 <<2:	SubEvt_Tamper_Scene_Changed

Face Detection

0:	SubEvt_Face_None
1 <<0:	SubEvt_Face_Blocked
1 <<1:	SubEvt_Face_VIP
1 <<2:	SubEvt_Face_Expired
1 <<3:	SubEvt_Face_Unknown
1 <<4:	SubEvt_Face_Generic
1 <<5:	SubEvt_Face_Masked
1 <<6:	SubEvt_Face_Unmasked

Object Detection

0:	SubEvt_Object_None
1 <<0:	SubEvt_Object_Person
1 <<1:	SubEvt_Object_Motion
1 <<2:	SubEvt_Object_Weapon
1 <<3:	SubEvt_Object_Vehicle
1 <<4:	SubEvt_Object_Animal
1 <<5:	SubEvt_Object_Item
1 <<6:	SubEvt_Object_Sports

Camera Event

0:	SubEvt_Camera_None

1 <<0:	SubEvt_Camera_Reset
1 <<1:	SubEvt_Camera_Reboot
1 <<2:	SubEvt_Camera_Power_Up
1 <<3:	SubEvt_Camera_Restart
1 <<4:	SubEvt_Camera_Disable
1 <<5:	SubEvt_Camera_Enable
1 <<6:	SubEvt_Camera_SD_Format
1 <<7:	SubEvt_Camera_Upgrade
1 <<8:	SubEvt_Camera_Suspend
1 <<9:	SubEvt_Camera_Resume
1 <<10:	SubEvt_Camera_Interruption

Recording

0:	SubEvt_Rec_None
1 <<0:	SubEvt_Rec_Continuous
1 <<1:	SubEvt_Rec_Detection
1 <<2:	SubEvt_Rec_Manual
1 <<3:	SubEvt_Rec_Temp

System Event

0:	SubEvt_System_None
1 <<0:	SubEvt_System_Startup
1 <<1:	SubEvt_System_Halt
1 <<2:	SubEvt_System_Reboot
1 <<3:	SubEvt_System_Reload
1 <<4:	SubEvt_System_Disk
1 <<5:	SubEvt_System_Upgrade
1 <<6:	SubEvt_System_Downgrade
1 <<7:	SubEvt_System_Loadgui
1 <<8:	SubEvt_System_Update

Event State:

Multiple bits can be active at the same time

0:	State_None = 0,	
1 <<0:	State_Active	Active recording
1 <<1:	State_Inactive	
1 <<1:	State_Edge	On SD card
1 <<1:	State_NonEdge	

1 <<1:	State_Locked	Locked recording file
1 <<1:	State_UnLocked	

Example: Enumerate events for 2 cameras (cam1 and cam2) for a given time period.

```
curl -k -v -b cookie.txt https://192.168.1.99/api/v1/
Timeline?device_name=cam1,cam2&start_time=2021-01-01-01-00-
00&end_time=2021-01-02-14-00-00
```

Face recognition

FortiRecorder has a built-in face recognition module that can detect and recognize faces. This API allows access to the logged events and manages the user database.

Authentication

When making requests to FortiRecorder appliance using the REST API, you must authenticate.

To establish a valid authentication session, you must make an HTTP POST request to the FortiRecorder login handler with the name `admin` and its password. The request should contain JSON data with `name` and `password` fields:

URL: `http(s)://host_or_ip/api/v1/AdminLogin/`

Method: `POST`

JSON:

```
{"name":"admin",
 "password":"****"}
```

If login is successful, the response will contain the authentication token in the APSCOOKIE cookie value. This cookie value must be included in any further requests.

Example

```
curl -v -H "Content-Type: application/json" -X POST -d
'{"name":"admin","password":"*****"}' https://ip_or_host/api/v1/AdminLogin
-c cookiefile
```

If login is successful, the cookies will be saved to a cookie file, which will be used in the following commands.

Note: Administrator account permissions affect which objects and operations you have access to. ***Admin profile: System configuration is required***.

REST API reference

Face_recognitionUser

Create a user

URL: `http(s)://host_ip/api/v1/Face_recognitionUser/{user_id}`

Method: `POST`

where

user_id:	unique ID for new user

Get the user image list

URL: `http(s)://host_ip/api/v1/Face_recognitionUser/{user_id}`

Method: `GET`

where

user_id:	unique ID of user

Response:

mkey	user_id
totalRemoteCount	number of entries in collection

Collection: [group1_summary->images**]** The image list of each user

is_local	false	This picture is pushed by REST API

Example: Create a user

```
curl -k -v -b cookiefile -X POST
https://ip_or_host/api/v1/Face_recognitionUser/user001
```

Example: Get the user image list

```
curl -k -v -b cookiefile -X GET
https://ip_or_host/api/v1/Face_recognitionUser/pt001
```

Response example:

```
{
"objectID": "Face_recognitionUser:pt001",
"reqAction": 1,
"nodePermission": 3,
"mkey": "pt001",
"department": "default-department",
"role": "default-role",
"display_name": "",
"image_content": "face-recognition/employees/pt001/default_image.jpg",
"group1_summary": {
"images": [
{
"image_path": "face-recognition/employees/pt001/group1/
      image/68_1584718836.jpg",
"is_local": true,
"mkey": "68_1584718836"
},
{
"image_path": "face-recognition/employees/pt001/group1/
     image/73_1584718823.jpg",
"is_local": true,
"mkey": "73_1584718823"
},
{
"image_path": "face-recognition/employees/pt001/group1/
     image/74_1584718818.jpg",
"is_local": true,
"mkey": "74_1584718818"
}
]
},
"department_display": "default-department",
"role_display": "default-role",
"default_images": "[]",
"percent": {}
}
```

EmployeeFaceRecord

Push image in base64 format

URL: `http(s)://host_ip/api/v1/EmployeeFaceRecord`

Method: `POST`

Where

employeeUsername	user_id	
fileindex	file_name	
option	raw	
	mtcnn	If the picture has been processed by mtcnn
content	picture in base64 string	

Delete an existing image

URL: `http(s)://host_ip/api/v1/EmployeeFaceRecord`

Method: `DEL`

Where

employeeUsername	user_id	
fileindex	file_name	'mkey' from the get user image list API

AiLog

Download the face-recognition log based on time range

URL:

```
http://{ip}/api/v1/AiLog?type=activity&subtype=activity&
range=TIMESTAMP&startIndex=0&pageSize=10&
start_time={start_timestamp}&end_time={end_timestamp}&
person={user_id}&camera={camera_id}
```

Method: `GET`

Where

type	activity	fixed
subtype	activity	fixed
range	TIMESTAMP	fixed
startIndex	0	0 based offset
page Size	10	number of records returned
start_time		start timestamp
end_time		end timestamp
person	KNOWN UNKNOWN person_name	all known persons are shown all unknown persons are shown
camera	cam_id	the camera id in setting

Response:

nextPage	true	keep updating `startIndex` to fetch all records
	false	indicates collection has been finished and no further data can be requested

RTSP playback

You can use RTSP (RFC 2326) to connect to FortiRecorder and play previously recorded video.

URL

Video players can connect to FortiRecorder using the following URL schemes:

```
rtsp://<FRC>:8554/cam=<CAMERA>
```

```
rtsp://<FRC>:8554/cam=<CAMERA>?time=<STARTTIME>
```

where:

- <FRC> -- host name or IP address of FortiRecorder
- <CAMERA> -- name of a camera in the FortiRecorder configuration
- <STARTTIME> -- optional; video recording timestamp to start playback at, in seconds since January 1, 1970, UTC; if not specified, the default is 60 seconds before the current time

Authentication

FortiRecorder uses HTTP digest authentication, with the same username and password as the video service. See Enabling video clip service.

RTSP commands

PLAY

Recorded time of frames

During playback, gaps between videos will be skipped. To communicate the recorded frame time to the RTSP client, each frame's NTP timestamp is embedded as an RTP extension. See section 6.4 of the ONVIF Streaming Specification.

Example

```
OPTIONS rtsp://172.20.131.56:8554/cam=cam1 RTSP/1.0
```

```
CSeq: 0


RTSP/1.0 200 OK

CSeq: 0

Public: OPTIONS, DESCRIBE, SETUP, TEARDOWN, PLAY, PAUSE


DESCRIBE rtsp://172.20.131.56:8554/cam=cam1 RTSP/1.0

CSeq: 1


RTSP/1.0 401 Unauthorized

CSeq: 1

WWW-Authenticate: Digest realm="FortiRecorder",
nonce="427e06651ff20c8c4edb0d2b7425062c"


DESCRIBE rtsp://172.20.131.56:8554/cam=cam1 RTSP/1.0

CSeq: 2

Authorization: Digest username="test", realm="FortiRecorder",
nonce="427e06651ff20c8c4edb0d2b7425062c",
uri="rtsp://172.20.131.56:8554/cam=cam1",
response="a085f7fac48b09f98a59bd6978d914ae"


RTSP/1.0 200 OK

CSeq: 2

Content-Base: rtsp://172.20.131.56:8554/cam=cam1

Content-Length: 249


v=0

o=- 0 0 IN IP4 127.0.0.1

t=0 0
```

```
a=tool:FortiRecorder

m=video 0 RTP/AVP 96

a=rtpmap:96 H264/90000

a=fmtp:96 packetization-mode=1; sprop-parameter-
sets=Z0IAKOkAoAQNcSAA27oAM3+YA2IEJQ==,aM4xUg==; profile-level-id=420028

a=control:streamid=0


SETUP rtsp://172.20.131.56:8554/cam=cam1 RTSP/1.0

CSeq: 3

Authorization: Digest username="test", realm="FortiRecorder",
nonce="427e06651ff20c8c4edb0d2b7425062c",
uri="rtsp://172.20.131.56:8554/cam=cam1",
response="7bc23478b463d1e3e130eae2f6155ec4"

Transport: RTP/AVP;unicast;client_port=8382-8383


RTSP/1.0 200 OK

CSeq: 3

Transport: RTP/AVP;unicast;client_port=8382-8383;server_port=20942-
20943;ssrc=19d26fa8

Session: rQ4NTXL27tda9abD


PLAY rtsp://172.20.131.56:8554/cam=cam1 RTSP/1.0

CSeq: 4

Session: rQ4NTXL27tda9abD

Authorization: Digest username="test", realm="FortiRecorder",
nonce="427e06651ff20c8c4edb0d2b7425062c",
uri="rtsp://172.20.131.56:8554/cam=cam1",
response="91298a6df73132a544c85adb9082c3d0"

Range: clock=20230712T191144-


RTSP/1.0 200 OK
```

```
Range: clock=20230712T191144-

CSeq: 4

RTP-Info:
url=rtsp://172.20.131.56:8554/cam=cam1/streamid=0;seq=1;rtptime=2519022532

Session: rQ4NTXL27tda9abD


PLAY rtsp://172.20.131.56:8554/cam=cam1 RTSP/1.0

CSeq: 5

Session: rQ4NTXL27tda9abD

Authorization: Digest username="test", realm="FortiRecorder",
nonce="427e06651ff20c8c4edb0d2b7425062c",
uri="rtsp://172.20.131.56:8554/cam=cam1",
response="91298a6df73132a544c85adb9082c3d0"

Range: clock=20230712T191154-


RTSP/1.0 200 OK

Range: clock=20230712T191154-

CSeq: 5

RTP-Info:
url=rtsp://172.20.131.56:8554/cam=cam1/streamid=0;seq=92;rtptime=2519072025

Session: rQ4NTXL27tda9abD


TEARDOWN rtsp://172.20.131.56:8554/cam=cam1 RTSP/1.0

CSeq: 6

Session: rQ4NTXL27tda9abD

Authorization: Digest username="test", realm="FortiRecorder",
nonce="427e06651ff20c8c4edb0d2b7425062c",
uri="rtsp://172.20.131.56:8554/cam=cam1",
response="00dd6110ab53bb2400b4601700eb286b"

RTSP/1.0 200 OK

CSeq: 6
```

```
Session: rQ4NTXL27tda9abD
```

Using VLC

VLC does not currently support absolute times, so use the URL scheme which specifies the start time.

VLC will set the playback time using the normal playback time (`npt`) units in the time range header:

```
Range: npt=0-
```

On the first request with a `PLAY` command, this will be the offset from the start time (usually zero). To resume after pausing, VLC issues a `PLAY` command without the `Range:` specified. To seek, VLC issues a PLAY command again, but with the `Range:`.

To allow seeking outside of the default range, FortiRecorder tries to re-center the slider by adjusting the returned range:

```
Range: npt=1800-3600
```

Example

```
OPTIONS rtsp://172.20.131.56:8554/cam=cam1 RTSP/1.0
CSeq: 2
User-Agent: LibVLC/3.0.18 (LIVE555 Streaming Media v2016.11.28)

RTSP/1.0 200 OK
CSeq: 2
Public: OPTIONS, DESCRIBE, SETUP, TEARDOWN, PLAY, PAUSE

DESCRIBE rtsp://172.20.131.56:8554/cam=cam1 RTSP/1.0
CSeq: 3
User-Agent: LibVLC/3.0.18 (LIVE555 Streaming Media v2016.11.28)
Accept: application/sdp

RTSP/1.0 401 Unauthorized
CSeq: 3
```

```
WWW-Authenticate: Digest realm="FortiRecorder",
nonce="bda3ff5dd24b4b7175cd7420af971b16"


DESCRIBE rtsp://172.20.131.56:8554/cam=cam1 RTSP/1.0

CSeq: 4

Authorization: Digest username="test", realm="FortiRecorder",
nonce="bda3ff5dd24b4b7175cd7420af971b16",
uri="rtsp://172.20.131.56:8554/cam=cam1",
response="2d602b41a77a93e0c606f17d79638faf"

User-Agent: LibVLC/3.0.18 (LIVE555 Streaming Media v2016.11.28)

Accept: application/sdp


RTSP/1.0 200 OK

CSeq: 4

Content-Base: rtsp://172.20.131.56:8554/cam=cam1

Content-Length: 249


v=0

o=- 0 0 IN IP4 127.0.0.1

t=0 0

a=tool:FortiRecorder

m=video 0 RTP/AVP 96

a=rtpmap:96 H264/90000

a=fmtp:96 packetization-mode=1; sprop-parameter-
sets=Z0IAKOkAoAQNcSAA27oAM3+YA2IEJQ==,aM4xUg==; profile-level-id=420028

a=control:streamid=0


SETUP rtsp://172.20.131.56:8554/cam=cam1/streamid=0 RTSP/1.0

CSeq: 5

Authorization: Digest username="test", realm="FortiRecorder",
nonce="bda3ff5dd24b4b7175cd7420af971b16",
uri="rtsp://172.20.131.56:8554/cam=cam1",
response="e8c10f302cafb5d7133766ad088deb72"

User-Agent: LibVLC/3.0.18 (LIVE555 Streaming Media v2016.11.28)

Transport: RTP/AVP;unicast;client_port=65070-65071



RTSP/1.0 200 OK

CSeq: 5

Transport: RTP/AVP;unicast;client_port=65070-65071;server_port=20028-
20029;ssrc=19d26fa8

Session: kQ16P8x1hF0qgrGp



PLAY rtsp://172.20.131.56:8554/cam=cam1 RTSP/1.0

CSeq: 6

Authorization: Digest username="test", realm="FortiRecorder",
nonce="bda3ff5dd24b4b7175cd7420af971b16",
uri="rtsp://172.20.131.56:8554/cam=cam1",
response="9e62f34bde86046e32e7894476458635"

User-Agent: LibVLC/3.0.18 (LIVE555 Streaming Media v2016.11.28)

Session: kQ16P8x1hF0qgrGp

Range: npt=0.000-



RTSP/1.0 200 OK

Range: npt=3540-3600

CSeq: 6

RTP-Info:
url=rtsp://172.20.131.56:8554/cam=cam1/streamid=0;seq=1;rtptime=2616060667

Session: kQ16P8x1hF0qgrGp



PAUSE rtsp://172.20.131.56:8554/cam=cam1 RTSP/1.0
```

```
CSeq: 7

Authorization: Digest username="test", realm="FortiRecorder",
nonce="bda3ff5dd24b4b7175cd7420af971b16",
uri="rtsp://172.20.131.56:8554/cam=cam1",
response="907fe83beeb1ed15301f9dd8b79f049b"

User-Agent: LibVLC/3.0.18 (LIVE555 Streaming Media v2016.11.28)

Session: kQ16P8x1hF0qgrGp



RTSP/1.0 200 OK

CSeq: 7

Session: kQ16P8x1hF0qgrGp



PLAY rtsp://172.20.131.56:8554/cam=cam1 RTSP/1.0

CSeq: 8

Authorization: Digest username="test", realm="FortiRecorder",
nonce="bda3ff5dd24b4b7175cd7420af971b16",
uri="rtsp://172.20.131.56:8554/cam=cam1",
response="9e62f34bde86046e32e7894476458635"

User-Agent: LibVLC/3.0.18 (LIVE555 Streaming Media v2016.11.28)

Session: kQ16P8x1hF0qgrGp

Range: npt=2818.440-



RTSP/1.0 200 OK

Range: npt=2812-3600

CSeq: 8

RTP-Info:
url=rtsp://172.20.131.56:8554/cam=cam1/streamid=0;seq=534;rtptime=2616574891

Session: kQ16P8x1hF0qgrGp



TEARDOWN rtsp://172.20.131.56:8554/cam=cam1 RTSP/1.0
```

```
CSeq: 9

Authorization: Digest username="test", realm="FortiRecorder",
nonce="bda3ff5dd24b4b7175cd7420af971b16",
uri="rtsp://172.20.131.56:8554/cam=cam1",
response="c3b4afc480000393849b5a552fbd3722"

User-Agent: LibVLC/3.0.18 (LIVE555 Streaming Media v2016.11.28)

Session: kQ16P8x1hF0qgrGp



RTSP/1.0 200 OK

CSeq: 9

Session: kQ16P8x1hF0qgrGp
```

High Performance Network Security