FORTINET

High Performance Network Security

FortiRecorder™ REST API Reference

Version 6.4.0

July 22, 2021

TABLE OF CONTENTS

Introduction

This document provides the REST API information supported in FortiRecorder version 6.4.0 release.
It covers several APIs available on FortiRecorder:

- **FortiRecorder video clip service REST API.**
  This API allows access to recorded video clips and snapshots from recordings.
- **FortiRecorder system level resources REST API.**
  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.
- **FortiRecorder Face Recognition REST API.**
  These APIs provides eservices related to the built-in face recognition of FortiRecorder. Event logs and the user database can be accessed.

FortiRecorder REST API HTTP methods and response codes

When using the APIs, the following conventions are followed:

HTTP Method	Usage
HTTP GET	To retrieve all resources or particular resource
HTTP POST	To create a new resource or perform certain administrative actions
HTTP PUT	To update an existing resource
HTTP DELETE	To delete an existing resource

FortiRecorder REST APIs use well-defined HTTP status codes to indicate query results to the API. Following are some of the HTTP status codes used:

HTTP Response Code	Description
200 - OK	API request successful.
400 - Bad Request	Bad request.
403 - Forbidden	Request is missing authentication token or administrator is missing access profile permissions.
404 - Not Found	Unable to find the specified resource.
405 - Method Not Allowed	Specified HTTP method is not allowed for this resource
500 – Server Error	Internal Server Error

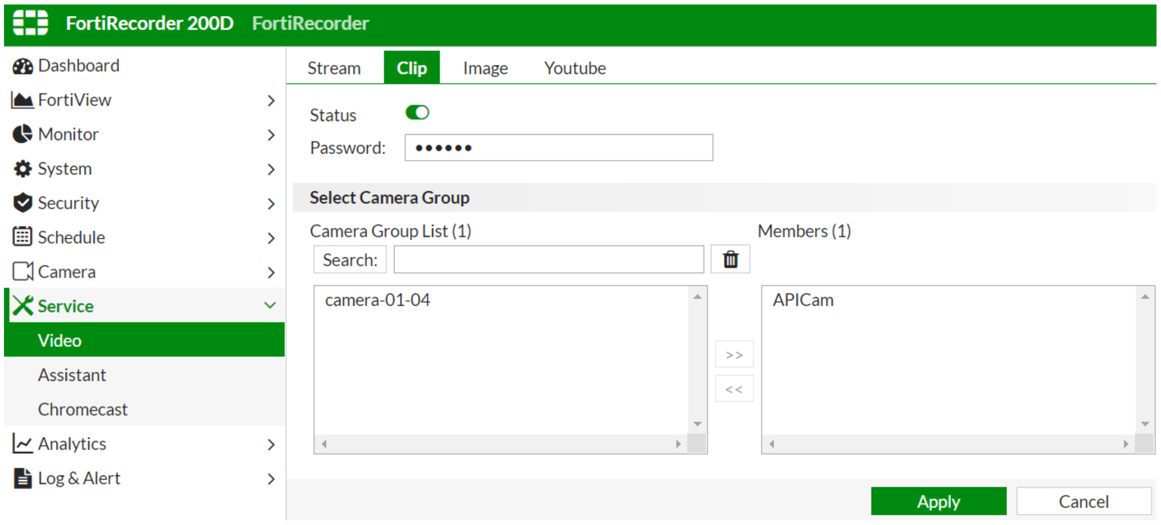
REST API for video clip service

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

Enabling Video clip service REST API support

This feature can be enabled in FortiRecorder under *Service > Video > Clip*.

You can choose a password and select a list of cameras that should be accessible through this service.



Status
Password: ••••••
Select Camera Group
Camera Group List (1)
Search:
camera-01-04
Members (1)
APICam

Authentication

To establish a valid authentication session, you must make a POST request to the FortiRecorder login handler with the name "service-clip" and the password defined in the FortiRecorder service section. The POST 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 value must be included in any further requests.

Example: Login to clip service API

```
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

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

JSON:

```
{"reqAction":21,
 "camera":"camera_name",
 "begin":1584734700,
 "end":1584734900}
```

Where

reqAction:	21 -- required, fixed
camera:	camera name as defined in FRC
begin:	timestamp for beginning of clip in UTC
end:	timestamp for end of clip in UTC. If equal to begin it indicates download of snapshot in jpg, otherwise clip in mp4.

Note: Time stamps are in UTC, so the local time on the recorder has to be converted with the right timezone. OSD of 1pm PDT needs timestamp of 8pm GMT.

Example: Download a 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/

--file video.mp4 will contain clip
```

Example: Download a snaphot jpg

```
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/

--file snapshot.jpg will contain snapshot
```

REST API for system level resources

FortiRecorder supports retrieval and modification of system level CMDB configuration settings as well as system level statistics. The API can be accessed using the following general 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 sub / child resources, use this to query 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/`	---	return 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, refer to the System Resources List.

This enables intgration 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

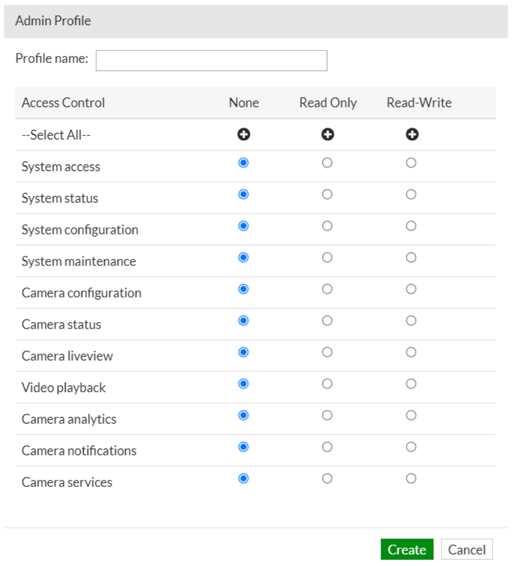
When making requests to FortiRecorder appliance using the REST API, you will need to pass the authentication. Currently the following authentication option is available:

- Local user password-based authentication

You also need the appropriate admin profile to access the FortiRecorder resources. See ‘System resource list and URLs’ to find out which profiles are needed.

For Method GET, Read Only is required as a minimum.

For the other methods, Read-Write is required.



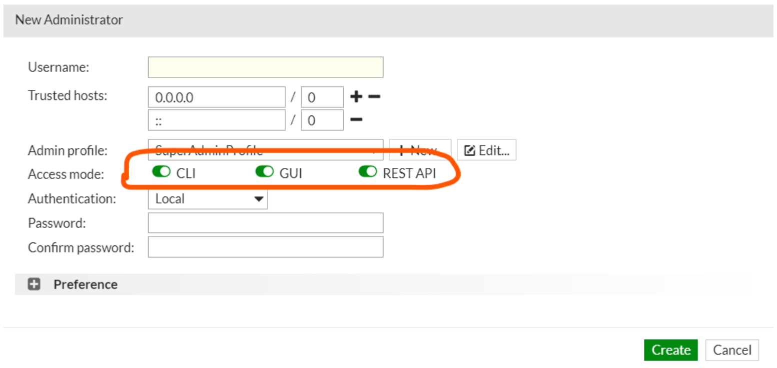
Admin Profile

Profile name:

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	●	○	○

Create Cancel

Also the admin account used for authentication has to have the REST API Access mode enabled.



New Administrator

Username:	
Trusted hosts:	0.0.0.0 / 0
	:: / 0
Admin profile:	SuperAdminProfile
Access mode:	CLI GUI REST API
Authentication:	Local
Password:	
Confirm password:	

Preference

Create Cancel

Password-based authentication

To establish a valid authentication session, you must make a POST request to the FortiRecorder login handler with your admin username and password. The POST 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: Admin login with password-based authentication
```
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 save to cookie.txt, which will be used in the following commands.

System resource list and URLs

URL	HTTP Method	Admin Profiles	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
/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 is present in JSON, it takes precedence over HTTP method header (i.e. HTTP 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 below 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
video_codec	0	Codec type

video_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: 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 FRC 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"}

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	interface is enabled
	false	interface is disabled
allowaccess	151	
discover	true	camera discovery enabled
webaccess	1	GUI access enabled
link_status	true	interface is up
	false	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 e.g. FK-SVM0000000000
port_http	80	Interface http port
port_https	443	Interface https port
port_ssh	22	Interface ssh port
port_telnet	23	Interface ssh port
port_frc_central	8550	Interface FortiCentral port
frc_central_secure		true: force SSL connection for FortiCentral
port_rtsp	554	Interface rtsp port
public_address		public IP address as seen from outside gateway
public_https_port	443	public https port
public_http_port	80	public http port
public_rtsp_port	554	public rtsp port
public_ftp_port	21	public ftp port
public_frc_central_port	8550	public FortiCentral port
public_notify_tcp_port	3010	public event notification TCP port

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
```

REST API for Face Recognition

FortiRecorder has a built-in face recognition module that allows detecting and recognizing faces. This API allows access to the logged events and manage the user database.

Authentication

When making requests to FortiRecorder appliance using the REST API, you will need to pass the authentication.

Password-based authentication

To establish a valid authentication session, you must make a POST request to the FortiRecorder login handler with your admin username and password. The POST 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: Admin login with password-based authentication

```
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 save to cookiefile, which will be used in the following commands.

Note: The permissions for the administrative account you use will affect which objects and operations you'll 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	all known persons are shown
	UNKNOWN	all unknown persons are shown
	person_name	
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

High Performance Network Security