

VIGI NVR Open API Document

Directory

1.	OVERVIEW	1
1.1	INTRODUCTION	1
1.2	UPDATE HISTORY	2
2.	OPENAPI TRANSACTION	2
2.1	FORMAT	2
2.1.1	The response format of NVR OpenAPI Event Protocol	2
2.1.2	The request format of NVR OpenAPI control interface	2
2.1.3	The request format of stream interface	3
2.2	AUTHENTICATION	3
3.	NVR OPENAPI EVENT PROTOCOL	4
3.1	EVENT SERVER SETTING	4
3.2	MESSAGE FORMAT	5
4.	OPENAPI INTERFACE	5
4.1	SPECIAL INTERFACE	6
4.1.1	GET /openapi/token	6
4.2	CHANNEL MANAGEMENT	8
4.2.1	GET /openapi/added_devices	8
4.3	VIDEO	9
4.3.1	GET /openapi/resolution	9
4.3.2	GET /openapi/valid_resolutions	9
4.3.3	POST /openapi/resolution	10
4.3.4	GET /openapi/bitrate	11
4.3.5	GET /openapi/bitrate_capability	12
4.3.6	POST /openapi/bitrate	13
4.4	TIME	14
4.4.1	GET /openapi/timing_mode	14
4.4.2	PUT /openapi/timing_mode	14
4.4.3	GET /openapi/ntp	15

4.4.4	PUT /openapi/ntp.....	16
4.5	AUDIO	16
4.5.1	GET /openapi/audio/output/sound	16
4.5.2	GET /openapi/audio/input/sound	17
4.5.3	POST /openapi/audio/output/sound	18
4.5.4	POST /openapi/audio/input/sound	19
4.6	DISK.....	20
4.6.1	GET /openapi/disks	20
4.6.2	GET /openapi/esata_disks.....	21
4.6.3	POST /openapi/smartctl_process.....	22
4.6.4	GET /openapi/smartctl_process/capability	22
4.6.5	POST /openapi/smartctl_process/test.....	23
4.6.6	GET /openapi/smartctl_process/schedule	24
4.6.7	GET /openapi/smartctl_process/attribute.....	25
4.7	POE	25
4.7.1	GET /openapi/poe/info.....	25
4.7.2	POST /openapi/poe/info	27
4.7.3	GET /openapi/poe/link_mode	28
4.7.4	POST /openapi/poe/link_mode.....	30
4.7.5	GET /openapi/poe/status.....	30
4.7.6	GET /openapi/poe/link_status.....	33
4.8	EVENT	33
4.8.1	GET /openapi/event_server	33
4.8.2	POST /openapi/event_server	34
4.8.3	POST /openapi/event_server/ delete_server.....	35
4.9	RECORDING.....	36
4.9.1	GET /openapi/record/days	36
4.9.2	GET /openapi/record/search/free_process.....	40
4.9.3	GET /openapi/record/search/results.....	41
4.9.4.	POST/openapi/record_control	43

4.10	SYSTEM.....	44
4.10.1	POST /openapi/systemctl	44
5.	OPENAPI STREAM INTERFACE	45
5.1	METHODS	45
5.2	URL	45
5.3	DIGEST AUTHENTICATION	46
APPENDIX I	ERROR CODE INFORMATION	47
APPENDIX II	PAYLOAD TYPE	48
APPENDIX III	AREA ABOUT 'SETTIMEZONE'	50
APPENDIX IV	EVENT TYPE AND SUBTYPE	52

1. Overview

1.1 Introduction

VIGI NVR OpenAPI consists of three parts: control protocol, event protocol and stream protocol. The NVR OpenAPI Control Protocol is used for management of NVR. The NVR OpenAPI Event Protocol can be used to receive NVR event message. VIGI NVR uses RTSP as stream protocol to get streams.

TERMS:

Name	Content
Start Line	First line of HTTP message, read RFC 7230 3.1 to know more.
Request Target	Part of Start Line, read RFC 7230 5.3 to know more.
Origin Form	A kind of form of Request Target, read RFC 7230 5.3.1 to know more.
Absolute URI	An URL format, read RFC 3986 4.3 to know more.
Digest Authentication	An authentication scheme, read RFC 7617 to know more.
Bearer Authentication	An authentication scheme, read RFC 6750 to know more.
Header Field	Part of HTTP message, which is after Start Line and before Message Body, read RFC 7230 3.2 to know more.
Multipart	A media type, read RFC 7578 to know more.
Boundary	A parameter of multipart/form-data, read RFC 2046 5.1 to know more.
Percent-encoding	A mechanism to encode in URL, it is sometimes called URL encoding, read RFC 3986 2.1 and 2.4 to know more.
RTSP	Real Time Streaming Protocol, read RFC 2326 to know more.
Digest Authentication	An authentication scheme, read RFC 7616 to know more.

1.2 Update History

Version	Description
V1.0	Initial release
V1.1	Updata 2.2 Authentication : The correction algorithm in response is SHA256 Updata Appendix IV event type and subtype: Add subtype
V1.2	Add 4.9.4: Record Control interface

2. OpenAPI Transaction

2.1 Format

2.1.1 The response format of NVR OpenAPI Event Protocol

The NVR OpenAPI Event Protocol need set event server domain and port, and NVR will push event message to event server when event happen.

The message format is as follows:

```
POST /event_message HTTP/1.1
Host: host
Content-Type: multipart/form-data;boundary=--VIGI-OPENAPI-MSG-PUSH
Content-Length: xxx

----VIGI-OPENAPI-MSG-PUSH
Content-Disposition: form-data;name="event"
Content-Type: application/json; charset=UTF-8
Content-Length: xxx

{"device_id": "id", "device_name": "name", "messages": [ {"type": type, "sub_type":
[ sub_type ], "localtime": "localtime", "port": "port" }, ... ] }

----VIGI-OPENAPI-MSG-PUSH--
```

Read section 3 to know more about event message format

2.1.2 The request format of NVR OpenAPI control interface

The NVR VIGI OpenAPI control interface is a group of RESTful interface with optional payload, the payload is JSON, request format is as follows:

```

METHOD path HTTP/1.1
Content-Type: application/json
Content-Length: xxx
{"key": "value", ...}
  
```

The interface only support HTTPS. The port of OpenAPI Discovery Protocol is 20443.

Notice: The Origin Form Request Target in Start Line is path and optional query, some application or library may request **Absolute URI**, which looks like:

https://<host>:20443<path>[?<key1>=<value1>[&<key2>=<value2>...]]

2.1.3 The request format of stream interface

The stream interface is an implement of **RTSP**, request format is as follow:

```

METHOD URL RTSP/1.0
CSeq: <sequence number>
<headers, like Accept, Transport, Session, ...>
  
```

Read section 5 to know more about METHOD and URL.

2.2 Authentication

Control interface request need a bear token, which should get by a GET request on

Digest Authentication. The format is as follows:

username: The login username

nonce: Uniquely generated string data in the response

realm: TP-LINK NVR

response: SHA-256 algorithm is used to calculate the parameters of Authentication response in request

```

/* Step 1: No Authentication request */
C->S:
GET /openapi/token HTTP/1.1
Authorization: <NULL>

/* reply */
S->C
HTTP/1.1 200 OK
WWW-Authenticate: Digest realm="TP-LINK NVR", nonce="5b032b462e0ad5ea60a0b44eaa871377", algorithm="SHA-256", url="/openapi/token"

/* Step 2: Digest Authentication */
C->S
GET /openapi/token HTTP/1.1
Authorization: Digest username="admin", nonce="50e2ad6177daf7d2e1b6f85b520d6
  
```

```
/* reply */
S->C
HTTP/1.1 200 OK
Content-Type: application/json
{
  "token_type": "bearer",
  "expires_in": 1800,
  "access_token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJhdXNlbgogImFkbWludlwiZmlydycCI6IChyY290liwglmV4cCI6IChyMDI0LTUyMDEyOjQyOjAxliB9.MuKRPjt0P%2fo
xRetkTLAmgG4nTwr3GwQdk%2f6iDn19%2fRSHpM16ZsFa%2f3aS5lsYwZOEkmSxxJVq
m0Xw0AvLGnpZg3zyD37wCmbfPQb7nLPPx%2bDIG%2bxa00Ayhs9di3SN%2bAdFdWTR
rbPt8%2bFJqBIK8U9jPpDYvnkPPIGXViozdRKSfA%3d",
  "refresh_token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJhdXNlbgogImFkbWludlwiZmlydycCI6IChyY290liwglmV4cCI6IChyMDI0LTUyMDE0OjEyOjQyOjAxliB9.N2zH24M%2ftfh
f%2bm9DhGdCEmWMRGWeWQ02bsDVT6nLWZBdJiztLDg9p8AftihRjKVtBkH9yypqIF0%
2bDHP6p9%2bQvXw4UJ7%2bvnuABDmT6MS2U8gttmSOTLxuadlvJQD5R%2fa5QfxtpH
i%2f89yAngMe%2bFxR%2f6CYj7EYOXUzObfkQr%2bZii8A%3d"
}
```

METHOD path HTTP/1.1

Authorization: Bearer access_token

The refresh_token is used for getting new token to reduce authentication, it has a longer lifetime than access_token. The format is as follows:

```
GET /openapi/token?grant_type=refresh_token&refresh_token=refresh_token
```

3. NVR OpenAPI Event Protocol

3.1 Event server setting

-4-

to know more.

3.2 Message format

The message is a **Multipart** and the **Boundary** is __VIGI-OPENAPI_MSG_PUSH, it may have 1 or more parts, JSON part and optional JPEG parts.

The following is JSON keys:

Name	Content
dev_name	String, the device name.
dev_id	String, the device ID.
messages	Object array, every element is a message object.

The message object keys is as follows:

Name	Content
type	Number, the type ID.
sub_type	Number array, every element is the sub_type ID.
localtime	String, local time.
channel	Optional, string, channel number, channel events should take it.
channel_name	Optional, string, channel name, channel events should take it.
port	Optional, string, POE port number, POE events should take it.
disk	Optional, string, disk slot number or None, disk events should take it.
array	Optional, string, RAID ID or None, RAID events should take it.
alarm_output	Optional, string, alarm output port number, alarm events should take it.
alarm_input	Optional, string, alarm input name, alarm events should take it.

Know more about type and sub_type by Appendix IV.

4. OpenAPI interface

Notice:

1. Example request omit some headers and response only retain JSON string.

2. All JSON string in request and response should encoded by **Percent-encoding** as URL,remember encode before send and decode after receive.
3. Every request should take token except getting token.

4.1 Special interface

4.1.1 GET /openapi/token

Name	Content	
Method	GET	
Path	/openapi/token	
Description	This API is used for getting token.	
Parameter	grant_type	Optional, used when getting token by refresh_token, value is refresh_token.
	refresh_token	Optional, used when getting token by refresh_token.
Response JSON keys	token_type	String, value is bearer.
	expires_in	Number, value is seconds until expired.
	access_token	String, value is the token for access.
	refresh_token	String, value is the token for refresh.
Digest Authentication Example	request: GET /openapi/token HTTP/1.1 Authorization: Digest username="admin", nonce="50e2ad6177daf7d2e1b6f85b520d613c", realm="TP-LINK NVR", response="efae5f651d98663461b18dad73453abfeedaad1501b78fe1d8760d1e0b616057"	
	response: <pre>{ "token_type": "bearer", "expires_in": 1800, "access_token": " eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCJ9.eyJhdXNlbnR5bG9jaW50d613c", "refresh_token": " eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCJ9.eyJhdXNlbnR5bG9jaW50d613c" }</pre>	

-7-

4.2 Channel Management

4.2.1 GET /openapi/added_devices

Name	Content	
Method	GET	
Path	/openapi/added_devices	
Description	This API is used for getting added devices.	
Response JSON keys	devices	Object array, value is an array of device objects.
Device object keys	id	Number, value is the channel number of device.
	name	String, device name which is as same as channel OSD name.
	alias	String, device name which cannot changed by NVR.
	online	String, value is 0 or 1, 0 means device offline, 1 means device online.
	ip	String, device IPv4 address.
	mac	String, device MAC address.
Example	request: GET /openapi/added_devices HTTP/1.1	
	response: <pre>{ "devices": [{ "id": 1, "name": "IPC%20OSD", "alias": "IPC", "online": "1", "ip": "192.168.0.2", "mac": "00-ff-00-13-31-01" }] }</pre>	

	<pre>], "error_code": 0 }] } </pre>
--	---

4.3 Video

4.3.1 GET /openapi/resolution

Name	Content	
Method	GET	
Path	/openapi/resolution	
Description	This API is used for getting channel resolution.	
Parameter	channel	Channel number.
	stream	Value is main or minor, used for specifying stream.
Response JSON keys	resolution	String, value is width*height.
Example	request: GET /openapi/resolution?channel=1&stream=minor HTTP/1.1	
	response: <pre> { "resolution": "640*480", "error_code": 0 } </pre>	

4.3.2 GET /openapi/valid_resolutions

Name	Content	
Method	GET	
Path	/openapi/valid_resolutions	
Description	This API is used for getting valid resolutions of a channel.	

Parameter	channel	Channel number.
	stream	Value is main or minor, used for specifying stream.
Response JSON keys	resolutions	Object array, value is an array of valid resolution object.
Valid resolutions object keys	resolution	String, value is width*height.
Example	request: GET /openapi/valid_resolutions?channel=1&stream=minor HTTP/1.1	
	response: <pre>{ "resolutions": [{ "resolution": "640*480" }, { "resolution": "352*288" }, { "resolution": "320*240" }], "error_code": 0 }</pre>	

4.3.3 POST /openapi/resolution

Name	Content	
Method	POST	
Path	/openapi/resolution	
Description	This API is used for setting resolution of a channel.	
Request	channel	Channel number.

JSON keys	stream	Value is main or minor, used for specifying stream.
	resolution	String, value is resolution string from valid_resolutions.
Example	request: POST /openapi/resolution HTTP/1.1 <pre>{ "channel": 1, "stream": "minor", "resolution": "320*240" }</pre>	
	response: <pre>{ "error_code": 0 }</pre>	

4.3.4 GET /openapi/bitrate

Name	Content	
Method	GET	
Path	/openapi/bitrate	
Description	This API is used for getting channel bitrate.	
Parameter	channel	Channel number.
	stream	Value is main or minor, used for specifying stream.
Response JSON keys	maximun_bitrate	String, unit is Kbps.
	type	String, cbr or vbr.
	quality	String, the level of quality. It's effective when type is VBR.
Example	request: GET /openapi/bitrate?channel=1&stream=minor HTTP/1.1	

	response: <pre>{ "maximun_bitrate": "1024", "type": "vbr", "quality": "5", "error_code": 0 }</pre>
--	--

4.3.5 GET /openapi/bitrate_capability

Name	Content	
Method	GET	
Path	/openapi/bitrate_capability	
Description	This API is used for getting channel bitrate capability.	
Parameter	channel	Channel number.
	stream	Value is main or minor, used for specifying stream.
Response JSON keys	valid_bitrates	String array, every element is valid bitrate.
	valid_qualities	String array, every element is valid quality.
Example	request: GET /openapi/bitrate_capability?channel=1&stream=minor HTTP/1.1	
	response: <pre>{ "valid_bitrates": ["256", "384", "512", "768", "1024", "1536", "2048", "2560"] }</pre>	

	<pre>], "valid_qualitys": ["5", "3", "1"], "error_code": 0 } </pre>
--	--

4.3.6 POST /openapi/bitrate

Name	Content	
Method	POST	
Path	/openapi/bitrate	
Description	This API is used for setting channel bitrate.	
Request JSON keys	channel	Channel number.
	stream	Value is main or minor, used for specifying stream.
	maximun_b itrate	String, value is in valid bitrates, unit is Kbps.
	type	String, cbr or vbr.
	quality	String, value is in valid qualitys, the level of quality. It's effective when type is VBR.
Example	request: POST /openapi/bitrate HTTP/1.1 <pre> { "channel": 1, "stream": "main", "maximun_bitrate": "1024", "type": "vbr", "quality": "5" } </pre>	

	response: <pre>{ "sub_code": 0 "error_code": 0 }</pre>
--	--

4.4 Time

4.4.1 GET /openapi/timing_mode

Name	Content	
Method	GET	
Path	/openapi/timing_mode	
Description	This API is used for getting timing mode.	
Response JSON keys	mode	String, value is ntp or manual.
Example	request: GET /openapi/timing_mode HTTP/1.1	
	response: <pre>{ "mode": "manual", "error_code": 0 }</pre>	

4.4.2 PUT /openapi/timing_mode

Name	Content	
Method	PUT	
Path	/openapi/timing_mode	
Description	This API is used for setting timing mode.	
Request JSON keys	mode	String, value is ntp or manual.

Example	request: PUT /openapi/timing_mode HTTP/1.1 <pre>{ "mode": "ntp" }</pre>
	response: <pre>{ "error_code": 0 }</pre>

4.4.3 GET /openapi/ntp

Name	Content	
Method	GET	
Path	/openapi/ntp	
Description	This API is used for getting ntp configuration.	
Response JSON keys	server	String, NTP server IPv4 address.
	port	Number, NTP server port.
Example	request: GET /openapi/ntp HTTP/1.1	
	response: <pre>{ "server": "0.0.0.0", "port": 999, "error_code": 0 }</pre>	

4.4.4 PUT /openapi/ntp

Name	Content	
Method	PUT	
Path	/openapi/ntp	
Description	This API is used for setting ntp configuration.	
Request JSON keys	server	String, NTP server IPv4 address.
	port	Number, NTP server port.
Example	request: PUT /openapi/ntp HTTP/1.1 <pre>{ "server": "0.0.0.0", "port": 999, "error_code": 0 }</pre>	
	response: <pre>{ "error_code": 0 }</pre>	

4.5 Audio

4.5.1 GET /openapi/audio/output/sound

Name	Content	
Method	GET	
Path	/openapi/audio/output/sound	
Description	This API is used for getting output sound information.	
Parameter	channel	Channel number.

Response JSON keys	mute	String, value is on or off, on means mute, off means not.
	volume	Number, value range from 0 to 100.
	system_volume	Number, value range from 0 to 100.
Example	request: GET /openapi/audio/output/sound?channel=1 HTTP/1.1	
	response: <pre>{ "mute": "on", "volume": 60, "system_volume": 60, "error_code": 0 }</pre>	

4.5.2 GET /openapi/audio/input/sound

Name	Content	
Method	GET	
Path	/openapi/audio/input/sound	
Description	This API is used for getting input sound information.	
Parameter	channel	Channel number.
Response JSON keys	mute	String, value is on or off, on means mute, off means not.
	volume	Number, value range from 0 to 100.
	noise_cancelling	String, value is on or off.
Example	request: GET /openapi/audio/input/sound?channel=1 HTTP/1.1	

	response: <pre>{ "mute": "off", "volume": 50, "noise_cancelling": "off", "error_code": 0 }</pre>
--	--

4.5.3 POST /openapi/audio/output/sound

Name	Content	
Method	POST	
Path	/openapi/audio/output/sound	
Description	This API is used for setting output sound.	
Response JSON keys	channel	Channel number.
	mute	String, value is on or off, on means mute, off means not.
	volume	Number, value range from 0 to 100.
	system_volume	Number, value range from 0 to 100.
Example	request: POST /openapi/audio/output/sound HTTP/1.1 <pre>{ "mute": "off", "volume": 100, "system_volume": 100, "channel": 1 }</pre>	
	response: <pre>{</pre>	

	<pre> "sub_code": 0 "error_code": 0 } </pre>
--	--

4.5.4 POST /openapi/audio/input/sound

Name	Content	
Method	POST	
Path	/openapi/audio/input/sound	
Description	This API is used for setting input sound.	
Response JSON keys	channel	Channel number.
	mute	String, value is on or off, on means mute, off means not.
	volume	Number, value range from 0 to 100.
	noise_cancelling	String, value is on or off.
Example	request: POST /openapi/audio/input/sound HTTP/1.1 <pre> { "mute": "off", "volume": 100, "noise_cancelling": "on", "channel": 1 } </pre>	
	response: <pre> { "sub_code": 0 "error_code": 0 } </pre>	

4.6 Disk

4.6.1 GET /openapi/disks

Name	Content	
Method	GET	
Path	/openapi/disks	
Description	This API is used for getting disk information.	
Response JSON keys	disks	Object array, every element is a disk object.
Disk object keys	total_space	String, format is %d GB.
	free_space	String, format is %d GB.
	status	String, value is unformatted, full, normal, miss, formatting or abnormal.
	slot	Number, indicate hardware interface on NVR of a disk.
	id	Number, an ID of disk.
	rw_attr	String, r means readable, w means writable.
Example	request: GET /openapi/disks HTTP/1.1	
	response: <pre>{ "disks": [{ "total_space": "698%20GB", "free_space": "120%20GB", "status": "normal", "slot": 0, "id": 1, "rw_attr": "rw" }] }</pre>	

	<pre> }], "error_code": 0 } </pre>
--	---

4.6.2 GET /openapi/esata_disks

Name	Content	
Method	GET	
Path	/openapi/esata_disks	
Description	This API is used for getting esata disk information.	
Response JSON keys	disks	Object array, every element is a disk object.
Disk object keys	total_space	String, format is %d GB.
	free_space	String, format is %d GB.
	status	String, value is unformatted, full, normal, miss, formatting or abnormal.
	slot	Number, indicate hardware interface on NVR of a disk.
	id	Number, an ID of disk.
	rw_attr	String, r means readable, w means writable.
Example	request: GET /openapi/disks HTTP/1.1	
	response: <pre> { "disks": [{ "total_space": "698%20GB", "free_space": "120%20GB", "status": "normal", "id": 1, </pre>	

	<pre> "rw_attr": "rw" }], "error_code": 0 } </pre>
--	---

4.6.3 POST /openapi/smartctl_process

Name	Content	
Method	POST	
Path	/openapi/smartctl_process	
Description	This API is used for controlling smartctl process, all SMART API need smartctl.	
Request JSON keys	action	String, value is start or stop. Start before use other SMART API and stop finally.
Example	request: POST /openapi/smartctl_process HTTP/1.1 <pre> { "action": "start" } </pre>	
	response: <pre> { "error_code": 0 } </pre>	

4.6.4 GET /openapi/smartctl_process/capability

Name	Content
------	---------

Method	GET	
Path	/openapi/smartctl_process/capability	
Description	This API is used for getting SMART capability, use after smartctl process start.	
Parameter	disk	The ID of disk, get from GET /openapi/disks.
Response JSON keys	short	String, value is on or off, on means support, off means not.
	extend	String, value is on or off, on means support, off means not.
	conveyance	String, value is on or off, on means support, off means not.
Example	request: GET /openapi/smartctl_process/capability?disk=1 HTTP/1.1	
	response: <pre>{ "short": "on", "extend": "on", "conveyance": "off", "error_code": 0 }</pre>	

4.6.5 POST /openapi/smartctl_process/test

Name	Content
Method	POST
Path	/openapi/smartctl_process/test

Description	This API is used for starting SMART test, use after smartctl process start.	
Request JSON keys	disk	Number, the ID of disk, get from GET /openapi/disks.
	type	String, value is short, extended or conveyance, the type of test.
Example	request: POST /openapi/smartctl_process/test HTTP/1.1 <pre>{ "disk": 1, "type": "extended" }</pre>	
	response: <pre>{ "error_code": 0 }</pre>	

4.6.6 GET /openapi/smartctl_process/schedule

Name	Content	
Method	GET	
Path	/openapi/smartctl_process/schedule	
Description	This API is used for getting SMART test schedule, use after smartctl process start.	
Parameter	disk	The ID of disk, get from GET /openapi/disks.
Response JSON keys	schedule	String, test schedule.

Example	request: GET /openapi/smartctl_process/schedule?disk=1 HTTP/1.1
	response: <pre>{ "schedule": "30%25", "error_code": 0 }</pre>

4.6.7 GET /openapi/smartctl_process/attribute

Name	Content	
Method	GET	
Path	/openapi/smartctl_process/attribute	
Description	This API is used for getting SMART attribute, use after smartctl process start.	
Parameter	disk	The ID of disk, get from GET /openapi/disks.
Response JSON keys	smart	Object, SMART information.
Example	request: GET /openapi/smartctl_process/attribute?disk=1 HTTP/1.1	
	response: <pre>{ "smart": {...}, "error_code": 0 }</pre>	

4.7 POE

4.7.1 GET /openapi/poe/info

Name	Content
Method	GET

Path	/openapi/poe/info	
Description	This API is used for getting POE information.	
Response JSON keys	info	Object array, every element is POE information object.
POE information object keys	port	Number, POE port number.
	enable	String, value is on or off, POE port switch.
	priority	String, value is 0, 1 or 2, port power on priority, 0 means high, 1 means normal, 2 means low.
	max_port_power	Number, range from 0 to 300, max port power, unit is 0.1W.
Example	request: GET /openapi/poe/info HTTP/1.1	
	response: <pre>{ "info": [{ "port": 1, "enable": "on", "priority": "0", "max_port_power": 300 }, { "port": 2, "enable": "on", "priority": "0", "max_port_power": 300 }, { "port": 3, "enable": "on", "priority": "0", "max_port_power": 300 }] }</pre>	

```
{
  {
    "port": 4,
    "enable": "on",
    "priority": "0",
    "max_port_power": 300
  },
  {
    "port": 5,
    "enable": "on",
    "priority": "0",
    "max_port_power": 300
  },
  {
    "port": 6,
    "enable": "on",
    "priority": "0",
    "max_port_power": 300
  },
  {
    "port": 7,
    "enable": "on",
    "priority": "0",
    "max_port_power": 300
  },
  {
    "port": 8,
    "enable": "on",
    "priority": "0",
    "max_port_power": 300
  }
},
{
  "error_code": 0
}
}
```

4.7.2 POST /openapi/poe/info

Name	Content
------	---------

Method	POST	
Path	/openapi/poe/info	
Description	This API is used for setting POE information.	
Request JSON keys	port	Number, indicate POE port.
	enable	String, value is on or off, POE port switch.
	priority	String, value is 0 or 1 port power on priority, 0 means high, 1 means normal
	max_port_power	Number, range from 0 to 300, max port power, unit is 0.1W.
Example	request: POST /openapi/poe/info HTTP/1.1 <pre>{ "port": 1, "enable": "off", "priority": "1", "max_port_power": 200 }</pre>	
	response: <pre>{ "error_code": 0 }</pre>	

4.7.3 GET /openapi/poe/link_mode

Name	Content	
Method	GET	
Path	/openapi/poe/link_mode	
Description	This API is used for getting POE port link mode.	
Response	port	Number, POE port number.

JSON keys	link_mode	String, value is half_10, full_10, half_100 or full_100, means duplex and speed.
Example	request: GET /openapi/poe/link_mode HTTP/1.1	
	response: <pre>{ "link_mode": [{ "port": 1, "link_mode": "full_100" }, { "port": 2, "link_mode": "full_100" }, { "port": 3, "link_mode": "full_100" }, { "port": 4, "link_mode": "full_100" }, { "port": 5, "link_mode": "full_100" }, { "port": 6, "link_mode": "full_100" }, { "port": 7, "link_mode": "full_100" }, { "port": 8,</pre>	

	<pre> "link_mode": "full_100" }], "error_code": 0 } </pre>
--	---

4.7.4 POST /openapi/poe/link_mode

Name	Content	
Method	POST	
Path	/openapi/poe/link_mode	
Description	This API is used for setting POE port link mode.	
Request JSON keys	port	Number, indicate POE port.
	link_mode	String, value is half_10, full_10, half_100 or full_100, means duplex and speed.
Example	request: POST /openapi/poe/link_mode HTTP/1.1 <pre> { "port": 1, "link_mode": "full_10" } </pre>	
	response: <pre> { "error_code": 0 } </pre>	

4.7.5 GET /openapi/poe/status

Name	Content
Method	GET

Path	/openapi/poe/status	
Description	This API is used for getting POE port status.	
Response JSON keys	status	Object array, every element is a global status object or a port status object.
Global JSON keys	sys_power	Number, total power of NVR, the unit is 0.1W.
	cost_power	Number, cost power of NVR, the unit is 0.1W.
Port JSON keys	cost	Number, cost power of port, the unit is 0.1W.
	status	Number, value range from 0 to 6, 0 means no power device, 1 means power good, 2 means power down, 3 means illegal power device, 4 means short circuit, 5 means overload, 6 over temperature.
Example	request: GET /openapi/poe/status HTTP/1.1	
	response: <pre>{ "status": [{ "global": { "sys_power": 1130, "cost_power": 0 } }, { "port1": { "cost": 0, "status": 0 } }, { "port2": { "cost": 0,</pre>	

```

        "status": 0
    },
    {
        "port3": {
            "cost": 0,
            "status": 0
        }
    },
    {
        "port4": {
            "cost": 0,
            "status": 0
        }
    },
    {
        "port5": {
            "cost": 0,
            "status": 0
        }
    },
    {
        "port6": {
            "cost": 0,
            "status": 0
        }
    },
    {
        "port7": {
            "cost": 0,
            "status": 0
        }
    },
    {
        "port8": {
            "cost": 0,
            "status": 0
        }
    }

```

	<pre> }], "error_code": 0 } </pre>
--	---

4.7.6 GET /openapi/poe/link_status

Name	Content	
Method	GET	
Path	/openapi/poe/link_status	
Description	This API is used for getting POE port link status.	
Response JSON keys	link_status	String, every character indicate a port, 1 means linked, 0 means not.
Example	request: GET /openapi/poe/link_status HTTP/1.1	
	response: <pre> { "link_status": "00001010", "error_code": 0 } </pre>	

4.8 Event

4.8.1 GET /openapi/event_server

Name	Content	
Method	GET	
Path	/openapi/event_server	
Description	This API is used for getting event server information.	
Response JSON keys	id	Number, value range from 1 to 3, provide an ID and avoid conflict.
	ip_or_domain	String, event server ip or domain.

	port	Number, event server port.
	url	String, event server url.
	protocol	String, protocol,value is HTTP or HTTPS.
	picture_switch	String, value is on or off.
Example	request: GET /openapi/event_server HTTP/1.1	
	<pre>{ "event_server": [{ "id": 1, "ip_or_domain": "192.168.137.1", "port": 3001, "url": "XX", "protocol": "HTTPS", "picture_switch": "on" }, { "id": 2, "ip_or_domain": "192.168.137.5", "port": 3004, "url": "XX", "protocol": "HTTPS", "picture_switch": "on" }], "error_code": 0 }</pre>	

4.8.2 POST /openapi/event_server

Name	Content	
Method	POST	
Path	/openapi/event_server	
Description	This API is used for adding an event server.	
Request JSON keys	id	Number, value range from 1 to 3, provide an ID and avoid conflict.

	ip_or_domain	String, event server ip or domain.
	port	Number, event server port.
	protocol	String, protocol,value is HTTP or HTTPS.
	url	String, event server url.
	picture_switch	String, value is on or off.
Example	request: POST /openapi/event_server/new_server HTTP/1.1 <pre>{ "id": "1", "ip_or_domain": "192.168.0.110", "port": 3000, "url": "XX", "protocol": "HTTPS", "picture_switch": "off" }</pre>	
	response: <pre>{ "error_code": 0 }</pre>	

4.8.3 POST /openapi/event_server/delete_server

Name	Content	
Method	POST	
Path	/openapi/event_server/delete_server	
Description	This API is used for deleting an event server.	
Request JSON keys	id	Number, value range from 1 to 3, the ID of the event server's.

Example	request: POST /openapi/event_server/delete_server HTTP/1.1 <pre>{ "id": 1 }</pre>
	response: <pre>{ "error_code": 0 }</pre>

4.9 Recording

These interfaces provide information of recording which can be replayed on stream interface.

4.9.1 GET /openapi/record/days

Name	Content	
Method	GET	
Path	/openapi/record/days	
Description	This API is used for getting days having recording by a range of month.	
Parameter	channel	Number, channel number.
	start	String, the format is %Y%m, like 202310, the first month to find.
	end	String, the format is %Y%m, like 202312, the last month to find.
Response JSON keys	days	Object array, every element is a day object.
Day JSON keys	day	String, the format is %Y%m%d, like 20231221, the day having recording.
Example	request: GET /openapi/record/days?channel=1&start=202310&end=202312 HTTP/1.1	

response:

```
{
  "days": [
    {
      "day": "20231001"
    },
    {
      "day": "20231009"
    },
    {
      "day": "20231010"
    },
    {
      "day": "20231011"
    },
    {
      "day": "20231012"
    },
    {
      "day": "20231013"
    },
    {
      "day": "20231014"
    },
    {
      "day": "20231019"
    },
    {
      "day": "20231020"
    },
    {
      "day": "20231021"
    },
    {
      "day": "20231022"
    }
  ],
}
```

	<pre> { "day": "20231023" }, { "day": "20231024" }, { "day": "20231025" }, { "day": "20231026" }, { "day": "20231027" }, { "day": "20231102" }, { "day": "20231103" }, { "day": "20231104" }, { "day": "20231105" }, { "day": "20231106" }, { "day": "20231107" }, { "day": "20231108" }, { "day": "20231109" </pre>
--	--

	<pre> }, { "day": "20231110" }, { "day": "20231111" }, { "day": "20231112" }, { "day": "20231113" }, { "day": "20231114" }, { "day": "20231115" }, { "day": "20231120" }, { "day": "20231121" }, { "day": "20231122" }, { "day": "20231123" }, { "day": "20231124" }, { "day": "20231125" }, { </pre>
--	---

	<pre> "day": "20231126" }, { "day": "20231127" }, { "day": "20231128" }, { "day": "20231129" }, { "day": "20231130" }, { "day": "20231221" }, { "day": "20231222" }], "error_code": 0 } </pre>
--	---

4.9.2 GET /openapi/record/search/free_process

Name	Content	
Method	GET	
Path	/openapi/record/search/free_process	
Description	This API is used for getting an ID of free search process.	
Response JSON keys	process	Number, ID of a free search process.
Example	request: GET /openapi/record/search/free_process HTTP/1.1	

	response: <pre>{ "process": 6, "error_code": 0 }</pre>
--	--

4.9.3 GET /openapi/record/search/results

Name	Content	
Method	GET	
Path	/openapi/record/search/results	
Description	This API is used for getting days having recording by a range of month.	
Parameter	channel	Number, channel number.
	process	Number, search process ID, a new search need get a free process ID from /openapi/record/search/free_process and reuse it when update start_index and end_index.
	day	String, the format is %Y%m%d, like 20231221, the day to search.
	start_index	Number, index of the first result to get, a search process may have too many results to show, use start_index and end_index to get part of them, if the number of results of a search process is n, the index range from 0 to n - 1, if cannot get any results by this API, it means the start_index is too big. And the number of results of a search process should not be more than 100, otherwise cannot get any results by this API.
	end_index	Number, index of last result to get.
Response JSON keys	results	Object array, every element is a time range object, which means a smallest unit to play, the continuous units can played by the start time of first unit and the end time of last unit.
Time range	start_time	String, the timestamp of start time.

JSON keys	end_time	String, the timestamp of end time.
Example	request: GET /openapi/record/search/results?channel=1&process=1&day=20231221 &start_index=0&end_index=9 HTTP/1.1	
	response: <pre>{ "results": [{ "start_time": "1703101145", "end_time": "1703101169" }, { "start_time": "1703101169", "end_time": "1703101181" }, { "start_time": "1703101181", "end_time": "1703101315" }, { "start_time": "1703101315", "end_time": "1703101337" }, { "start_time": "1703101337", "end_time": "1703101877" }, { "start_time": "1703101877", "end_time": "1703101894" }] }</pre>	

```

    },
    {
      "start_time": "1703101894",
      "end_time": "1703102458"
    },
    {
      "start_time": "1703102458",
      "end_time": "1703102482"
    },
    {
      "start_time": "1703102482",
      "end_time": "1703102614"
    },
    {
      "start_time": "1703102614",
      "end_time": "1703102632"
    }
  ],
  "error_code": 0
}

```

4.9.4. POST/openapi/record_control

Name	Content	
Method	POST	
Path	/openapi/record_control	
Description	This API is used for setting record control.	
Request JSON keys	channel	Channel number.
	enable	String, auto or off.
Example	request: POST /openapi/record_control HTTP/1.1	

	<pre>{ "channel": 1, "enable": "off" }</pre>
	response: <pre>{ "error_code": 0 }</pre>

4.10 System

4.10.1 POST /openapi/systemctl

Name	Content	
Method	POST	
Path	/openapi/systemctl	
Description	This API is used for some system settings, include reset and reboot.	
Request JSON keys	action	String, reset or reboot.
Example	request: POST /openapi/systemctl HTTP/1.1 <pre>{ "action": "reboot" }</pre>	
	response: <pre>{ "error_code": 0 }</pre>	

5. OpenAPI Stream interface

VIGI NVR OpenAPI implement RTSP as stream interface, both support over TCP and over UDP.

5.1 Methods

Method	Live supported	Replay supported
OPTIONS	yes	yes
DESCRIBE	yes	yes
SETUP	yes	yes
TEARDOWN	yes	yes
PLAY	yes	yes
PAUSE	no	yes
GET_PARAMETER	yes	yes
SET_PARAMETER	yes	yes

5.2 URL

Name	Content
Live URL	rtsp://<IP>/live/<channel>/<stream>/avm
Replay URL	rtsp://<IP>/replay/<channel>/<stream>/avm?starttime=<starttime>&endtime=<endtime>
IP	NVR IP.
channel	Channel number.
stream	1 or 2, 1 means main stream and 2 means minor stream.
starttime & endtime	UTC time, format is YYYYMMDDtHHMMSSz, example, 19701111t123059z means 1970-11-11 12:30:59 UTC. 'z' is the zone designator for the zero UTC offset
Example	Live URL:

	rtsp://192.168.0.240/live/1/1/avm Replay URL: rtsp://192.168.0.240/replay/1/1/avm?starttime=20231111t120000z&endtime=20231111t123000z
--	---

5.3 Digest Authentication

Only support Digest authentication, example as follow,

NVR unauthorized response:

RTSP/1.0 401 Unauthorized<\r\n> CSeq: 1<\r\n> Date: Thu, 14 Dec 2023 08:18:32 GMT<\r\n> User-Agent: vigi<\r\n> WWW-Authenticate: Digest realm="vigi", nonce="2440a40", stale="FALSE"<\r\n> Content-Length: 0<\r\n\r\n>

Play request:

PLAY rtsp://192.168.0.240/live/1/1/avm RTSP/1.0<\r\n> CSeq: 2<\r\n> Authorization: Digest username="admin", realm="vigi", nonce="2440a40", uri="rtsp://192.168.0.240/live/1/1/avm", response="92bbdf372d026ba0de933b37ec427da7"<\r\n> User-Agent: LibVLC/2.2.4 (LIVE555 Streaming Media v2016.02.22)<\r\n> Session: 0x508448<\r\n> Range: npt=0.000-<\r\n\r\n>

NVR OK response:

RTSP/1.0 200 OK<\r\n> CSeq: 2<\r\n> Date: Thu, 14 Dec 2023 08:18:38 GMT<\r\n> User-Agent: vigi<\r\n> Session: 0x508448;timeout=60<\r\n> Range: npt=0.000-<\r\n> RTP-Info: <\r\n> Content-Length: 0<\r\n\r\n>

Appendix I Error code information

Error code information about OpenAPI interface

errCode	errMsg
-40101	System error
-40102	Abnormal Situation
-40103	Memory is not enough
-40104	Unsupported event
-40105	Unsupported operator
-40106	Unsupported command
-40107	Operation prohibited
-40108	Timeout
-40109	System is busy
-40110	Device is not found
-40201	Data overflow
-40202	Data is too long
-40203	The entry already exists
-40204	The entry has been associated
-40205	The entry does not exist
-40206	The entry conflict
-40207	The entry list is full
-40208	The entry list is empty
-40209	Parameter error
-40210	Format error
-40211	Missing necessary parameters
-40212	Boolean-type value taking error
-40213	The length of the string is illegal
-40301	The IP address is incorrect
-40302	Multicast IP address
-40303	Invalid IP address format
-40304	Loopback IP address
-40309	Illegal network segment
-40401	Authentication failed
-40402	Verification Code authentication failed
-40403	Session timeout
-40404	The client is locked
-40405	Currently in the factory reset state
-40406	Authentication failed. Exceed the number of supported clients
-40408	The system is locked
-70301	The username is empty
-70302	The username does not exist

-70303	The current user has no permission
-70304	Failed to obtain user information
-70312	Password error
-70901	Abnormal frame rate
-70902	Frame rate out of range
-70903	Abnormal Image quality
-70904	Image quality out of range
-70905	Abnormal Bit rate
-70906	Bit rare out of range
-70907	Resolution formate error
-70908	Resolution out of range
-70909	stream_type illegal
-70910	bitrate_type illegal
-70911	encode_type illegal
-71101	The number of clients is full, and currently there are up to 4 clients
-71102	The client ID is already occupied
-71103	The client ID is invalid
-71104	If there are no events on the day, or click play before you have searched for an event
-71105	Failed to search for events or get calendars
-71106	Year fails to get calendar
-71107	The time to search for an event or play is illegal
-71108	The channel is illegal when playing
-71109	No channel was selected during playback
-71110	Search events are not allowed while playing
-71111	The client has not established a socket connection and cannot be played
-71112	The parameter is incorrect
-71113	The file used to save the list of events failed to open
-71114	Unknown error
-71115	This error code indicates that the device has received a search request, but the search takes a long time and cannot be returned immediately
-76209	Illegal ID

Appendix II Payload Type

For the value of PT in RTP packets, see RFC3551.

PT of RTP, 96-127 is dynamic. There is no universal definition for dynamic PT except that 96 is

conventionally used for H264/H265. Therefore, in this document, some dynamic PT types are used as auxiliary data types for events transmission and other functions. GB28181 provides suggestions for the Payload Type of RTP. The value of the audio part is the same as that of RFC3551. This document adopts the recommended dynamic PT value of the video part, as shown in the following table.

Table 4 - 1 GB28181 Video section Payload Type

Type value	Type of payload	Reference format
97	MPEG-4, Video	RFC 3016
98	H.264, Video	RFC 3984
99	SVAC, Video	RFC 3984
8	G711A/PCMA, Audio	RFC 3551
20	SVACA, Audio	RFC 3551
4	G723, Audio	RFC 3551
18	G729, Audio	RFC 3551
9	G722, Audio	RFC 3551

Dynamic PT=96, still reserved for H.264 for compatibility. That is, the PT value of H.264 can be 96 or 98.

Appendix III area about 'setTimeZone'

AREA
Pacific/Wake
Pacific/Midway
Pacific/Honolulu
America/Anchorage
America/Los_Angeles
America/Phoenix
America/Chihuahua
America/Denver
America/Tegucigalpa
America/Chicago
America/Mexico_City
Canada/Saskatchewan
America/Bogota
America/New_York
America/Indiana/Indianapolis
America/Caracas
America/Asuncion
America/Halifax
America/Cuiaba
America/La_Paz
America/Santiago
Canada/Newfoundland
America/Sao_Paulo
America/Buenos_Aires
America/Cayenne
America/Godthab
America/Montevideo
Atlantic/South_Georgia
Atlantic/Azores
Atlantic/Cape_Verde
Africa/Casablanca
UTC
Europe/London
Atlantic/Reykjavik
Europe/Amsterdam
Europe/Belgrade
Europe/Brussels
Europe/Sarajevo
Africa/Algiers

Europe/Athens
Asia/Beirut
Africa/Cairo
Asia/Damascus
Africa/Harare
Europe/Vilnius
Asia/Jerusalem
Asia/Amman
Asia/Baghdad
Europe/Minsk
Asia/Kuwait
Africa/Nairobi
Asia/Istanbul
Europe/Moscow
Asia/Tehran
Asia/Muscat
Asia/Baku
Asia/Tbilisi
Asia/Yerevan
Asia/Kabul
Asia/Karachi
Asia/Yekaterinburg
Asia/Tashkent
Asia/Kolkata
Asia/Colombo
Asia/Katmandu
Asia/Dhaka
Asia/Rangoon
Asia/Bangkok
Asia/Novosibirsk
Asia/Krasnoyarsk
Asia/Hong_Kong
Asia/Kuala_Lumpur
Australia/Perth
Asia/Taipei
Asia/Ulaanbaatar
Asia/Irkutsk
Asia/Tokyo
Asia/Seoul
Asia/Yakutsk
Australia/Adelaide
Australia/Darwin
Australia/Brisbane

Australia/Canberra
Pacific/Guam
Australia/Hobart
Asia/Vladivostok
Pacific/Noumea
Asia/Magadan
Pacific/Auckland
Pacific/Fiji
Asia/Kamchatka
Pacific/Tongatapu

Appendix IV event type and subtype

Type table

type ID	type string
1	Channel event.
2	Device exception.
7	Store exception.

Subtype table

type ID	subtype ID	string
1	1	Time.
	2	Motion detection.
	2, 101	Motion detection/Person
	2, 102	Motion detection/Vehicle
	3	Camera tampering.
	4	Line Crossing detection.
	4, 101	Line Crossing detection/Person
	4, 102	Line Crossing detection/Vehicle
	5	Intrusion detection.
	5, 101	Intrusion detection/Person
	5, 102	Intrusion detection/Vehicle
	6	Region entering detection.
	6, 101	Region entering detection/Person
	6, 102	Region entering detection/Vehicle
	7	Region exiting detection.
	7, 101	Region exiting detection/Person
	7, 102	Region exiting detection/Vehicle
	8	Loitering detection.
	8, 101	Loitering detection/Person
	8, 102	Loitering detection/Vehicle
	9	Crowd detection.
	10	Fast motion detection.

	11	Park detection.
	12	Object abandoned detection.
	13	Object removal detection.
	14	Audio error.
	15	Out of focus.
	16	Scene Change detection.
	17	Face detection.
	18	Alarm singal.
	19	Video loss.
	20	Video message.
	21	Human detection.
	22	Vehicle detection.
	23	Abandon or take.
	24	Doorbell.
	25	Cry detection.
	26	Emergency call.
	36	Push I frame.
	38	IPC Alarm Input
2	1	SD card missing.
	2	SD card full.
	3	Disk missing.
	4	Disk full.
	5	Device offline.
	6	SD card error.
	7	Disk error.
	8	Log in error.
	9	POE single port shortcut.
	10	POE single port overload.
	11	POE chip temperature error.
	12	POE total overload.
	13	Alarm input.
	14	Fan abnormal.
	15	IP conflict.
	17	Disk disable.
7	1	RAID offline.
	2	RAID degraded.
	3	RAID disk abnormal.
	4	RAID spare abnormal.
	5	No array.
	6	Array full.
	7	No HD.
	8	HD password error.
	9	RAID password error

	10	Expansion HD password error
--	----	-----------------------------