

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	3
2.1.3 The request format of stream interface	3
2.2 AUTHENTICATION	3
3. NVR OPENAPI EVENT PROTOCOL.....	5
3.1 EVENT SERVER SETTING.....	5
3.2 MESSAGE FORMAT	5
4. OPENAPI INTERFACE	6
4.1 SPECIAL INTERFACE.....	6
4.1.1 GET /openapi/token.....	6
4.1.2 GET /openapi/module_list.....	8
4.2 CHANNEL MANAGEMENT.....	10
4.2.1 GET /openapi/added_devices.....	10
4.2.2 POST /openapi/add_device.....	11
4.2.3 POST /openapi/remove_device	12
4.2.4 GET /openapi/device_scan	13
4.2.5 GET /openapi/snapshot.....	14
4.2.6 POST /openapi/add_device_rtsp	14
4.3 VIDEO	15
4.3.1 GET /openapi/resolution	15
4.3.2 GET /openapi/valid_resolutions.....	16
4.3.3 POST /openapi/resolution	17
4.3.4 GET /openapi/bitrate	18
4.3.5 GET /openapi/bitrate_capability.....	18
4.3.6 POST /openapi/bitrate	19
4.4 TIME.....	20
4.4.1 GET /openapi/timing_mode	20
4.4.2 PUT /openapi/timing_mode	21
4.4.3 GET /openapi/ntp	22
4.4.4 PUT /openapi/ntp.....	22
4.5 AUDIO.....	23
4.5.1 GET /openapi/audio/output/sound.....	23
4.5.2 GET /openapi/audio/input/sound	24
4.5.3 POST /openapi/audio/output/sound	24
4.5.4 POST /openapi/audio/input/sound	25
4.5.5 GET /openapi/audio/channel_capability	26

4.5.6	GET /openapi/audio/capability	27
4.6	DISK	27
4.6.1	GET /openapi/disks	27
4.6.2	GET /openapi/esata_disks	29
4.6.3	POST /openapi/smartctl_process	30
4.6.4	GET /openapi/smartctl_process/capability	30
4.6.5	POST /openapi/smartctl_process/test	31
4.6.6	GET /openapi/smartctl_process/schedule	32
4.6.7	GET /openapi/smartctl_process/attribute	33
4.7	POE	33
4.7.1	GET /openapi/poe/info	33
4.7.2	POST /openapi/poe/info	35
4.7.3	GET /openapi/poe/link_mode	36
4.7.4	POST /openapi/poe/link_mode	38
4.7.5	GET /openapi/poe/status	38
4.7.6	GET /openapi/poe/link_status	41
4.8	EVENT	41
4.8.1	GET /openapi/event_server	41
4.8.2	POST /openapi/event_server	42
4.8.3	POST /openapi/event_server/delete_server	43
4.9	RECORDING	44
4.9.1	GET /openapi/record/days	44
4.9.2	GET /openapi/record/search/free_process	48
4.9.3	GET /openapi/record/search/results	49
4.9.4	POST /openapi/record_control	51
4.10	SYSTEM	52
4.10.1	POST /openapi/systemctl	52
4.11	PTZ	53
4.11.1	GET /openapi/ptz/capability	53
4.11.2	POST /openapi/ptz/move	54
4.11.3	GET /openapi/ptz/park	55
4.11.4	POST /openapi/ptz/park	56
4.11.5	GET /openapi/ptz/preset	57
4.11.6	GET /openapi/ptz/tour	58
4.11.7	GET /openapi/ptz/target_track	60
4.11.8	POST /openapi/ptz/target_track	60
4.11.9	GET /openapi/ptz/batch_capability	61
4.12	DISARMING	64
4.12.1	GET /openapi/alarm_device/alarm_mode	64
4.12.2	POST /openapi/alarm_device/alarm_mode	65
4.13	ACTIVE DEFENSE	65
4.13.1	GET /openapi/active_defense/arm_status	65
4.13.2	POST /openapi/active_defense/arm_status	66
4.14	ALARM OUTPUT	67
4.14.1	GET /openapi/alarm_output/nvr_alarm	67

4.14.2	POST /openapi/alarm_output/nvr_alarm	68
4.14.3	GET /openapi/alarm_output/ipc_alarm	69
4.14.4	POST /openapi/alarm_output/ipc_alarm	70
4.14.5	POST /openapi/alarm_output/nvr_manual_alarm	71
4.14.6	POST /openapi/alarm_output/ipc_manual_alarm	72
4.14.7	GET /openapi/alarm_output/batch_get_ipc_alarm	72
5.	OPENAPI STREAM INTERFACE.....	74
5.1	METHODS.....	74
5.2	DIGEST AUTHENTICATION	74
5.3	URL	75
5.4	TALK.....	76
APPENDIX I	ERROR CODE INFORMATION	78
APPENDIX II	PAYLOAD TYPE.....	80
APPENDIX III	AREA ABOUT 'SETTIMEZONE'	82
APPENDIX IV	EVENT TYPE AND SUBTYPE	84
APPENDIX V	HOW TO TALK WITH NVR BY OPENAPI.....	86
	The request format of OpenAPI stream interface	86
	The details of the request and response are as follows	86

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	Update 2.2 Authentication : The correction algorithm in response is SHA256 Update Appendix IV event type and subtype: Add subtype
V1.2	Add 4.9.4 Recording interface
V1.3	Add 4.1.2 module_list interface Update 4.2 Channel Management interface 4.2.2 add device 4.2.3 delete device 4.2.4 scan device 4.2.5 snapshot Add 4.11 PTZ interface Update Appendix IV event type and subtype: Add subtype
V1.4	Update 4.2 Channel Management Add 4.2.6: add device using the RTPS protocol. Update 4.5 Audio Add 4.5.5: retrieve audio input and Output capabilities for each NVR channel. Add 4.5.6: get audio input and Output capabilities of the NVR. Update 4.11 PTZ Interface Update 4.11.1: add Pan/Tilt capability of the NVR channel. Add 4.11.9: get the PTZ capabilities of all NVR channels. Add 4.15 Alarm Output Interface Update 4.14 Alarm Output Update 4.14.1 4.14.2: Add alarm_type of alarm output Update Appendix I Error code Information

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>\[&<key2>=<value2>...\]\]](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/1.1 401
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="50e2ad6177daf7d2e1b6f85b520d613c", realm="TP-LINK NVR", response="efae5f651d98663461b18dad73453abfeedaad1501b78fe1d8760d1e0b616057"

/* reply */
S->C
HTTP/1.1 200 OK
Content-Type: application/json
{
  "token_type": "bearer",
  "expires_in": 1800,
  "access_token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJhdXNlbnR5bG9mFkbWlulwglmdyci6IChyYb290liwglmV4cCI6IChyMDI0LTEyLT12IDEyOjQyOjAxiB9.MuKRPjt0P%2foxREtkTLAmgG4nTwr3GwQdk%2f6iDn19%2fRSHpm16ZsFa%2f3aS5IsYwZOEkmSxkJVqm0Xw0AvLGnpZg3zyD37wCmbfPQb7nLPPx%2bDIG%2bxa00Ayhs9di3SN%2bAdFdWTRrbPt8%2bFJqBIK8U9jJpDYvnkPPIGXViozdRKSfA%3d",
  "refresh_token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJhdXNlbnR5bG9mFkbWlulwglmdyci6IChyYb290liwglmV4cCI6IChyMDI0LTEyLT12IDEyOjQyOjAxiB9.N2zH24M%2fthf%2bm9DHgDcEMwMRGWeWQ02bsDVT6nLWZBdJiztLDg9p8AftihRjKVtBkH9yypqIF0%2bDHP6p9%2bQvXw4UJ7%2bvnuABDmT6MS2U8gttmSOTLxuadlvJQD5R%2fa5QfXtpHi%2f89yAhgMe%2bFxR%2f6CYj7EYOXuZObfkQr%2bZii8A%3d"
}

```

The response of this request is a JSON, include token_type, expires_in, access_token and

refresh_token. The token_type will always be bearer, the expires_in is the seconds which token will expire in, the access_token should be added in **Header Field** of every control request for **Bearer Authentication**. The format is as follows:

```
METHOD path HTTP/1.1
Authorization: Bearer access_token
```

Notice: The token in JSON is encoded by **Percent-encoding** as URL, remember decode because **Header Field** need origin string.

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
```

The response is as same as **Digest Authentication**

3. NVR OpenAPI Event Protocol

NVR OpenAPI Event Protocol is used for setting event server and push event message to event server. It support push by HTTP or HTTPS and setting of pushing image or not.

3.1 Event server setting

Using GET /openapi/event_server to get setting and POST /openapi/event_server to set, read 4.9 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 J	token_type	String, value is bearer.

-7-

	response: <pre>{ "token_type": "bearer", "expires_in": 1800, "access_token": "eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCJ9.eyJhdXNlbnR5bG9mFk bWlulwglmdycCI6IChyYb290liwglmV4cCI6IChyMDIzLTExLTlzlDA5OjAxOjE4liB9.qyl6 YjObFJhMb3DfKhonvqKKA%2fisJul4gc6PcE1Je0xKZVySajs3yEjZh3r5mHT26Dnr% 2fK%2bhg39p4PaxN7UyooNiznZ05nl%2fAWA5LAcWGux7uCS1d%2ftyLJkQNQraX vnNeKDe6deJrhclX5krq2Zk6gJ%2fEIN1cHAqHXmLnaS%2fvpk%3d", "refresh_token": "eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCJ9.eyJhdXNlbnR5bG9mFk bWlulwglmdycCI6IChyYb290liwglmV4cCI6IChyMDIzLTExLTlzlDEwOjMxOjE4liB9.him nZua1dfDdEAcAwuWsDcb4nLYD9d8TtMJVDSiv096jle0j%2bss%2bcWrOjElauVFZ JFYD%2fQxu17N0alM9WCicXNE%2br7GUC9ydyhvYmLfJBjKBnesNeigPzEGQCNUe xdmQA91IYsk9Nm7A8hLlpCixd4FUBOfs%2b6EYLmB7v%2fygA%3d" }</pre>
--	--

4.1.2 GET /openapi/module_list

Name	Content	
Method	GET	
Path	/openapi/module_list	
Description	This API is used for getting supported modules and versions.	
Response JSON keys	name	String, name of supported module.
	version	Number, version of module.
Example	request: GET /openapi/module_list HTTP/1.1	
	response: <pre>{ "module_list": [{ "name": "channel_management", "version": 1 }] }</pre>	

```

    },
    {
      "name": "video",
      "version": 1
    },
    {
      "name": "time",
      "version": 1
    },
    {
      "name": "audio",
      "version": 1
    },
    {
      "name": "disk",
      "version": 1
    },
    {
      "name": "poe",
      "version": 1
    },
    {
      "name": "event",
      "version": 1
    },
    {
      "name": "recording",
      "version": 1
    },
    {
      "name": "system",
      "version": 1
    },
    {
      "name": "ptz",
      "version": 1
    },
    {

```

	<pre> "name": "alarm_device", "version": 1 }, { "name": "active_defense", "version": 1 }, { "name": "alarm_output ", "version": 1 }], "error_code": 0 }</pre>
--	--

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

4.2.2 POST /openapi/add_device

Name	Content	
Method	POST	
Path	/openapi/add_device	
Description	This API is used for adding a new device.	
Version	1	
Request JSON keys	username	String, device username.
	password	String, device password.
	connect_prot	String, value is TP-LINK, ONVIF. Used for specifying control protocol.

	ip	String, device IPv4 address.
	port	String, device management port.
Example	request: POST /openapi/add_device HTTP/1.1 <pre>{ "username": "admin", "password": "Admin11..", "connect_prot": "TP-LINK", "ip": "192.168.137.7", "port": "443" }</pre>	
	response: <pre>{ "error_code": 0 }</pre>	

4.2.3 POST /openapi/remove_device

Name	Content	
Method	POST	
Path	/openapi/remove_device	
Description	This API is used for removing a device.	
Version	1	
Request JSON keys	channel	Number, channel number of device.
Example	request: POST /openapi/remove_device HTTP/1.1 <pre>{ "channel": 2 }</pre>	

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

4.2.4 GET /openapi/device_scan

Name	Content	
Method	GET	
Path	/openapi/device_scan	
Description	This API is used for scanning for available devices in the network.	
Version	1	
Response JSON keys	devices	Object array, value is an array of device objects.
Device object keys	ip	String, device IPv4 address.
	name	String, device name which is as same as channel OSD name.
	connect_prot	String, value is TP-LINK, ONVIF, or RTSP. Used for specifying control protocol.
	port	String, device management port.
	mac	String, device MAC address.
	model	String, device model.
Example	request: GET /openapi/device_scan HTTP/1.1	
	response: <pre>{ "devices": [{</pre>	

	<pre> "ip": "192.168.137.195", "name": "VIGI%20C450%201.0_D875", "connect_prot": "TP-LINK", "port": "443", "mac": "00-ff-00-37-d8-75", "model": "VIGI%20C450" }], "error_code": 0 }</pre>
--	--

4.2.5 GET /openapi/snapshot

Name	Content	
Method	GET	
Path	/openapi/snapshot	
Description	This API is used for getting a snapshot of a channel.	
Version	1	
Parameter	channel	Channel number.
Response		JPEG file, snapshot of the channel.
Example	request: GET /openapi/snapshot?channel=1 HTTP/1.1	
	response: response.jpeg	

4.2.6 POST /openapi/add_device_rtsp

Name	Content
Method	POST
Path	/openapi/add_device_rtsp
Description	This API is used for adding a new device by RTSP protocol.

Version	1	
Request JSON keys	username	String, device username.
	password	String, device password.
	rtsp_url_main	String. Example: If the RTSP URL is rtsp://192.168.10.31:554/stream1, then the IP address is 192.168.10.31, the RTSP port is 554, and the resource path is stream1
Example	request: POST /openapi/add_device_rtsp HTTP/1.1 <pre>{ "username": "admin", "password": "Tplink.124", "rtsp_url_main": "rtsp://192.168.137.100:554/stream1" }</pre>	
	response: <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" }] }</pre>	

	<pre> }, { "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 JSON keys	channel	Channel number.
	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"], "valid_qualities": ["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 irate	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>{</pre>	

	<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>{ "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>	

4.5.5 GET /openapi/audio/channel_capability

Name	Content	
Method	GET	
Path	/openapi/audio/channel_capability	
Description	This API is used for getting audio input/output capability of IPC.	
Parameter	channel	Channel number.
Response JSON keys	speaker_enable	Number, value is 0 or 1, 0 indicates not supporting audio output, 1 indicates supporting.
	microphone_enable	Number, value is 0 or 1, 0 indicates not supporting audio input, 1 indicates supporting.
Example	request:	

	GET /openapi/audio/channel_capability?channel=1 HTTP/1.1
	response: <pre>{ "speaker_enable": 1, "microphone_enable": 1, "error_code": 0 }</pre>

4.5.6 GET /openapi/audio/capability

Name	Content	
Method	GET	
Path	/openapi/audio/capability	
Description	This API is used for getting audio input/output capability of NVR.	
Response JSON keys	speaker_enable	Number, value is 0 or 1, 0 indicates not supporting audio output, 1 indicates supporting.
	microphone_enable	Number, value is 0 or 1, 0 indicates not supporting audio input, 1 indicates supporting.
Example	request: GET /openapi/audio/capability HTTP/1.1	
	response: <pre>{ "speaker_enable": 1, "microphone_enable": 1, "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" }], "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, "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	disk	Number, the ID of disk, get from GET /openapi/disks.

JSON keys	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>{</pre>	

	<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 }, { "port": 4, "enable": "on", "priority": "0", </pre>	

	<pre> "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 } </pre>
--	---

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 JSON keys	port	Number, POE port number.
	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:

```
{
  "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,
      "link_mode": "full_100"
    }
  ],
  "error_code": 0
}
```

	}
--	---

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	status	Object array, every element is a global status object or a

JSON keys		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, "status": 0 } }], }</pre>	

```
{
  "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
            }
          }
        }
      ],
      "error_code": 0
    }
  }
```

	}
--	---

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"
    },
    {
      "day": "20231023"
    }
  ]
}
```

	<pre> }, { "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" }, { "day": "20231126" }, </pre>
--	---

	<pre> { "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>{</pre>	

	<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 JSON keys	start_time	String, the timestamp of start time.
	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>

	<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 } </pre>
--	--

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>	

4.11 PTZ

4.11.1 GET /openapi/ptz/capability

Name	Content	
Method	GET	
Path	/openapi/ptz/capability	
Description	This API is used for getting ptz capability of channel.	
Version	1	
Parameter	channel	Channel number.
Response JSON keys	pan_tilt_supported	String, value is 0 or 1.
	zoom_supported	String, value is 0 or 1.
	preset_supported	String, value is 0 or 1.
	preset_number_max	String, max preset number.
	tour_number_max	String, max tour number.
	pattern_number_max	String, max pattern number.
	aperture_supported	String, value is 0 or 1.
	focus_supported	String, value is 0 or 1.
	calibrate_supported	String, value is 0 or 1.
	diagonal_motion_supported	String, value is 0 or 1.
Example	request: GET openapi/ptz/capability?channel=1 HTTP/1.1	
	response: <pre>{ "pan_tilt_supported": "1",</pre>	

```

    "zoom_supported": "0",
    "preset_supported": "1",
    "preset_number_max": "300",
    "tour_number_max": "8",
    "pattern_number_max": "4",
    "tour_spots_number_max": "8",
    "tour_supported": "1",
    "pattern_supported": "0",
    "aperture_supported": "0",
    "focus_supported": "0",
    "calibrate_supported": "1",
    "diagonal_motion_supported": "0",
    "x_min": "-1.000000",
    "x_max": "1.000000",
    "y_min": "-1.000000",
    "y_max": "1.000000",
    "z_min": "0.000000",
    "z_max": "0.000000",
    "move_min": "0.000000",
    "move_max": "0.000000",
    "tour_stay_time_max": "3600000",
    "tour_stay_time_min": "60000",
    "error_code": 0
  }

```

4.11.2 POST /openapi/ptz/move

Name	Content	
Method	POST	
Path	/openapi/ptz/move	
Description	This API is used for operating PTZ of channel.	
Version	1	
Request JSON keys	channel	Channel number.
	direction	Number, operation of PTZ.

	speed	String, moving speed.
Example	request: POST /openapi/ptz/move HTTP/1.1 <pre>{ "channel": 1, "direction": 0, "speed": "0.6" }</pre>	
	response: <pre>{ "error_code": 0 }</pre>	

4.11.3 GET /openapi/ptz/park

Name	Content	
Method	GET	
Path	/openapi/ptz/park	
Description	This API is used for getting park information of channel.	
Version	1	
Parameter	channel	Channel number
Response JSON keys	action_mode	String, value is preset or tour.
	park_time	Number, park time.
	action_id	Number, action id.
	enabled	Number, value is 0 or 1.
Example	request: GET /openapi/ptz/park?channel=1 HTTP/1.1	

	response: <pre>{ "action_mode": "preset", "park_time": 5, "action_id": 2, "enabled": 0, "error_code": 0 }</pre>

4.11.4 POST /openapi/ptz/park

Name	Content	
Method	POST	
Path	/openapi/ptz/park	
Description	This API is used for setting park of channel.	
Version	1	
Request JSON keys	channel	Channel number.
	action_mode	String, value is preset or tour.
	park_time	Number, park time.
	action_id	Number, action id.
	enabled	Number, value is 0 or 1.
Example	request: POST /openapi/ptz/park HTTP/1.1 <pre>{ "action_mode": "tour", "park_time": 6,</pre>	

	<pre> "action_id": 2, "enabled": 1, "channel": 1 } </pre>
	response: <pre> { "error_code": 0 } </pre>

4.11.5 GET /openapi/ptz/preset

Name	Content	
Method	GET	
Path	/openapi/ptz/preset	
Description	This API is used for getting presets of channel.	
Version	1	
Parameter	channel	Channel number
Response JSON keys	preset_id	Number, preset id, range from 1 to 300.
	read_only	Number, value is 0 or 1.
	name	String.
Example	request: GET /openapi/ptz/preset?channel=1 HTTP/1.1	
	response: <pre> { "preset": [{ "preset_id": 1, "read_only": 0, "name": "Preset1" }] } </pre>	

	<pre> }, { "preset_id": 2, "read_only": 0, "name": "Preset2" }], "error_code": 0 } </pre>
--	--

4.11.6 GET /openapi/ptz/tour

Name	Content	
Method	GET	
Path	/openapi/ptz/tour	
Description	This API is used for getting tours of channel.	
Version	1	
Parameter	channel	Channel number
Response JSON keys	tour_id	Number, tour id, range from 1 to 8.
	preset_count	Number, preset number.
	name	String.
	preset_id	Object array, every element is a preset id.
	time	Object array, every element is retention period of preset.
	speed	Object array.
Example	request: GET /openapi/ptz/tour?channel=1 HTTP/1.1	

response:

```
{
  "tour": [
    {
      "tour_id": 1,
      "preset_count": 3,
      "name": "Patrol%20Path1",
      "preset_id": [
        "1",
        "2",
        "3"
      ],
      "time": [
        "60",
        "60",
        "60"
      ],
      "speed": [
        "0",
        "0",
        "0"
      ]
    },
    {
      "tour_id": 2,
      "preset_count": 0
    },
    {
      "tour_id": 3,
      "preset_count": 0
    },
    {
      "tour_id": 4,
```

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

4.11.7 GET /openapi/ptz/target_track

Name	Content	
Method	GET	
Path	/openapi/ptz/target_track	
Description	This API is used for getting traget track of channel.	
Version	1	
Parameter	channel	Channel number
Response JSON keys	enabled	String, value is on or off.
	people_enabled	String, value is on or off, enable human detection.
Example	request: GET /openapi/ptz/target_track?channel=1 HTTP/1.1	
	response: <pre> { "enabled": "off", "people_enabled": "on", "error_code": 0 } </pre>	

4.11.8 POST /openapi/ptz/target_track

Name	Content
Method	POST

Path	/openapi/ptz/target_track	
Description	This API is used for setting target track of channel.	
Version	1	
Request JSON keys	channel	Channel number.
	enabled	String, value is on or off.
	people_enabled	String, value is on or off, enable human detection.
Example	request: POST /openapi/ptz/target_track HTTP/1.1 <pre>{ "channel": 1, "enabled": "on", "people_enabled": "on" }</pre>	
	response: <pre>{ "error_code": 0 }</pre>	

4.11.9 GET /openapi/ptz/batch_capability

Name	Content	
Method	GET	
Path	/openapi/ptz/batch_capability	
Description	This API is used for getting ptz capability of all channel.	
Version	1	
Response JSON keys	id	Number, channel numbe.
	pan_tilt_supported	String, value is 0 or 1.

	zoom_supported	String, value is 0 or 1.
	preset_supported	String, value is 0 or 1.
	preset_number_max	String, max preset number.
	tour_number_max	String, max tour number.
	pattern_number_max	String, max pattern number.
	aperture_supported	String, value is 0 or 1.
	focus_supported	String, value is 0 or 1.
	calibrate_supported	String, value is 0 or 1.
	diagonal_motion_supported	String, value is 0 or 1.
Example	request: GET /openapi/ptz/batch_capability HTTP/1.1	
	response: <pre>{ "capability": [{ "id": 1, "pan_tilt_supported": "1", "zoom_supported": "0", "preset_supported": "1", "preset_number_max": "300", "tour_number_max": "8", "pattern_number_max": "4", "tour_spots_number_max": "8", "tour_supported": "1", "pattern_supported": "0", "aperture_supported": "0",</pre>	

```

    "focus_supported": "0",
    "calibrate_supported": "1",
    "x_min": "-1.000000",
    "x_max": "1.000000",
    "y_min": "-1.000000",
    "y_max": "1.000000",
    "z_min": "0.000000",
    "z_max": "0.000000",
    "move_min": "0.000000",
    "move_max": "0.000000",
    "tour_stay_time_max": "3600000",
    "tour_stay_time_min": "60000"
  },
  {
    "id": 2,
    "pan_tilt_supported": "1",
    "zoom_supported": "0",
    "preset_supported": "1",
    "preset_number_max": "300",
    "tour_number_max": "8",
    "pattern_number_max": "4",
    "tour_spots_number_max": "8",
    "tour_supported": "1",
    "pattern_supported": "0",
    "aperture_supported": "0",
    "focus_supported": "0",
    "calibrate_supported": "1",
    "x_min": "-1.000000",
    "x_max": "1.000000",
    "y_min": "-1.000000",
    "y_max": "1.000000",
    "z_min": "0.000000",
    "z_max": "0.000000",
  }

```

	<pre> "move_min": "0.000000", "move_max": "0.000000", "tour_stay_time_max": "3600000", "tour_stay_time_min": "60000" }], "error_code": 0 }</pre>
--	---

4.12 Disarming

4.12.1 GET /openapi/alarm_device/alarm_mode

Name	Content	
Method	GET	
Path	/openapi/alarm_device/alarm_mode	
Description	This API is used for getting One-Key Disarming mode.	
Version	1	
Parameter	device_id	Number, device id. Only supports device_id=1.
Response JSON keys	alarm_mode	String, value is alarm or disarming.
Example	request: GET /openapi/alarm_device/alarm_mode?device_id=1 HTTP/1.1	
	response: <pre> { "alarm_mode": "alarm", "error_code": 0 }</pre>	

4.12.2 POST /openapi/alarm_device/alarm_mode

Name	Content	
Method	POST	
Path	/openapi/alarm_device/alarm_mode	
Description	This API is used for setting One-Key Disarming mode.	
Version	1	
Request JSON keys	device_id	Number, device id. Only supports device_id=1.
	alarm_mode	String, value is alarm or disarming.
Example	request: POST /openapi/alarm_device/alarm_mode HTTP/1.1 <pre>{ "device_id": 1, "alarm_mode": "alarm" }</pre>	
	response: <pre>{ "error_code": 0 }</pre>	

4.13 Active Defense

4.13.1 GET /openapi/active_defense/arm_status

Name	Content
Method	GET
Path	/openapi/active_defense/arm_status
Description	This API is used for getting disarming state of channel.
Version	1

Parameter	channel	Number, channel number.
Response JSON keys	arm_status	String, value is armed, disarmed or disabled.
Example	request: GET /openapi/active_defense/arm_status?channel=1 HTTP/1.1	
	response: <pre>{ "arm_status": "armed", "error_code": 0 }</pre>	

4.13.2 POST /openapi/active_defense/arm_status

Name	Content	
Method	POST	
Path	/openapi/active_defense/arm_status	
Description	This API is used for setting disarming state of channel.	
Version	1	
Request JSON keys	channel	Number, channel number.
	arm_status	String, value is armed or disarmed.
Example	request: POST /openapi/active_defense/arm_status HTTP/1.1 <pre>{ "channel": 1, "arm_status": "armed" }</pre>	
	response:	

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

4.14 Alarm Output

4.14.1 GET /openapi/alarm_output/nvr_alarm

Name	Content	
Method	GET	
Path	/openapi/alarm_output/nvr_alarm	
Description	This API is used to getting Alarm Output device settings of nvr	
Version	1	
Response JSON keys	alarm_name	String, alarm output name.
	delay_time	String, alarm output duration. Value is [5, 10, 30, 60, 120, 300, 600] seconds.
	enabled	String, value is on or off.
	alarm_type	String, value is NO or NC. NO: Normally Open NC: Normally Closed
Example	request: GET /openapi/alarm_output/nvr_alarm HTTP/1.1	
	response: <pre>{ "alarm_output_info": [{ "device_id": 1, "delay_time": "10", "enabled": "off", "alarm_name": "AO1",</pre>	

```

    "alarm_type": "NO"
  },
  {
    "device_id": 2,
    "delay_time": "10",
    "enabled": "off",
    "alarm_name": "AO2",
    "alarm_type": "NO"
  },
  {
    "device_id": 3,
    "delay_time": "10",
    "enabled": "off",
    "alarm_name": "AO3",
    "alarm_type": "NO"
  },
  {
    "device_id": 4,
    "delay_time": "10",
    "enabled": "off",
    "alarm_name": "AO4",
    "alarm_type": "NO"
  }
],
"error_code": 0
}

```

4.14.2 POST /openapi/alarm_output/nvr_alarm

Name	Content
Method	POST
Path	/openapi/alarm_output/nvr_alarm
Description	This API is used to configure Alarm Output device settings of NVR.

Version	1	
Request JSON keys	device_id	Number, alarm device id.
	alarm_name	String, alarm output name.
	delay_time	String, alarm output duration. Value is [5, 10, 30, 60, 120, 300, 600] seconds.
	enabled	String, value is on or off.
	alarm_type	String, value is NO or NC. NO: Normally Open NC: Normally Closed
Example	request: POST /openapi/alarm_output/nvr_alarm HTTP/1.1 <pre>{ "device_id": 1, "alarm_name": "A1", "delay_time": "600", "enabled": "on", "alarm_type": "NC" }</pre>	
	response: <pre>{ "error_code": 0 }</pre>	

4.14.3 GET /openapi/alarm_output/ipc_alarm

Name	Content
Method	GET
Path	/openapi/alarm_output/ipc_alarm
Description	This API is used for getting Alarm Output device settings of channel.
Version	1

Parameter	channel	Number, Channel number.
Response JSON keys	alarm_name	String, alarm output name.
	delay_time	String, alarm output duration. Value is [5, 10, 30, 60, 120, 300, 600] seconds.
	enabled	String, value is on or off.
Example	request: GET /openapi/alarm_output/ipc_alarm?channel=1 HTTP/1.1	
	response: <pre>{ "alarm_name": "A1", "delay_time": "600", "enabled": "on", "error_code": 0 }</pre>	

4.14.4 POST /openapi/alarm_output/ipc_alarm

Name	Content	
Method	POST	
Path	/openapi/alarm_output/ipc_alarm	
Description	This API is used to configure Alarm Output device settings of channel.	
Version	1	
Request JSON keys	channel	Number, Channel number.
	alarm_name	String, alarm output name.
	delay_time	String, alarm output duration. Value is [5, 10, 30, 60, 120, 300, 600] seconds.
	enabled	String, value is on or off.

Example	request: POST /openapi/alarm_output/ipc_alarm HTTP/1.1 <pre>{ "channel": 1, "alarm_name": "A1", "delay_time": "600", "enabled": "on" }</pre>
	response: <pre>{ "error_code": 0 }</pre>

4.14.5 POST /openapi/alarm_output/nvr_manual_alarm

Name	Content	
Method	POST	
Path	/openapi/alarm_output/nvr_manual_alarm	
Description	This API is used to control Alarm Output of nvr.	
Version	1	
Request JSON keys	device_id	Number, alarm device id.
	action	String, value is start or stop.
Response JSON keys	timer	Number. Indicates the duration of this alarm output, in seconds.
Example	request: POST /openapi/alarm_output/nvr_manual_alarm HTTP/1.1 <pre>{ "device_id": 1, "action": "start" }</pre>	
	response:	

	<pre>{ "timer": 10, "error_code": 0 }</pre>
--	---

4.14.6 POST /openapi/alarm_output/ipc_manual_alarm

Name	Content	
Method	POST	
Path	/openapi/alarm_output/ipc_manual_alarm	
Description	This API is used to control Alarm Output of Channel	
Version	1	
Request JSON keys	channel	Number, Channel number.
	action	String, value is start or stop.
Response JSON keys	timer	Number. Indicates the duration of this alarm output, in seconds.
Example	request: POST /openapi/alarm_output/ipc_manual_alarm HTTP/1.1 <pre>{ "channel": 1, "action": "start" }</pre>	
	response: <pre>{ "timer": 30, "error_code": 0 }</pre>	

4.14.7 GET /openapi/alarm_output/batch_get_ipc_alarm

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

Method	GET	
Path	/openapi/alarm_output/batch_get_ipc_alarm	
Description	This API is used for getting Alarm Output device settings of channel.	
Version	1	
Response JSON keys	channel	Number, Channel number.
	alarm_name	String, alarm output name.
	delay_time	String, alarm output duration. Value is [5, 10, 30, 60, 120, 300, 600] seconds.
	enabled	String, value is on or off.
	manual_alarm_out_supported	String , value is 0 or 1. Indicates whether the channel supports the "ipc_manual_alarm" method
Example	request: GET /openapi/alarm_output/batch_get_ipc_alarm	
	response: <pre>{ "alarm_output": [{ "channel": 1, "delay_time": "600", "enabled": "off", "alarm_name": "AO111", "manual_alarm_out_supported": "1" }, { "channel": 2, "delay_time": "5", "enabled": "on",</pre>	

	<pre> "alarm_name": "A1", "manual_alarm_out_supported": "0" }], "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 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>
```

5.3 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

5.4 Talk

Name	Content	
Command	talk	
Description	This API is used by the client to initiate an audio session. Then, the client can send audio data to the NVR.	
Request	mode	String, talk mode, mandatory field. Value: half_duplex: Half-duplex mode aec: AEC Full-duplex mode
	channel	Number, channel number.
Response	null	
Example	request: talk to NVR: <pre>{ "method": "do", "talk": { } }</pre> talk to channel: <pre>{ "method": "do", "talk": { "mode": "half_duplex", "channel": 5 } }</pre>	

response:

```
{
  "error_code":0,
  "session_id":"xxx"
}
```

Once the client receives a successful response to the Talk request, it can start sending audio data to the NVR.

Currently, NVR Speaker supports G.711 μ -law audio with a sample rate of 16 kHz. Other audio formats or sample rates are not guaranteed to work correctly.

Audio data is transmitted through the RTSP interleaved channel over TCP. The audio stream should be sent as raw G.711 μ -law payload without an RTP header.

The audio packet format is as follows:

\$ (1B)	Chn ID (1B)	Length (2B)
Raw G.711 μ -law Audio Payload		

Field descriptions:

\$ (0x24): RTSP interleaved frame identifier.

Chn ID: RTSP interleaved channel ID.

Length: Length of the audio payload in bytes (excluding the 4-byte RTSP interleaved header).

Audio Payload: Raw G.711 μ -law encoded audio data.

For optimal playback quality on the NVR Speaker, the following settings are recommended:

Audio Codec: G.711 μ -law (PCMU)

Sample Rate: 16 kHz

Channel: Mono

Packet Size: 1056 bytes

Transmission Interval: 66 ms per packet

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
-52402	Invalid parameter
-52406	Internal error

-52411	Audio device busy
-52412	Unsupported audio mode
-52413	Channel offline
-52420	Channel does not support speaker
-70107	Interface parameter error
-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 rate out of range
-70907	Resolution format 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
-71301	Channel ID illegal
-71309	Action ID illegal
-71505	The IP address of the IPC is illegal
-71508	The port of the IPC is illegal
-71531	The domain name is illegal
-71533	The IPC has been managed by another nvr
-71534	The URL length is illegal
-71558	The channel is offline and cannot be operated.
-71559	The channel is not enabled and cannot be operated.
-71562	Device does not exist
-71563	The channel does not support this function
-73701	Alarm device enabled param illegal
-73702	Alarm device name length illegal
-73704	Alarm Output device delay time illegal
-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	IPC IP conflict.
	18	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

Appendix V How to talk with NVR by OpenAPI

Users can access via the OpenAPI interface using their own programs, with the request format based on the following:

The request format of OpenAPI stream interface

The VIGI OpenAPI stream request format is as follows:

```
MULTITRANS rtsp://ip/multitrans RTSP/1.0
CSeq : 1
Content-Type : application/json
Content-Length : xxx
{"type" : "request", "seq" : "1", "params" : {"method" : "xx"}}
```

- The VIGI OpenAPI stream information must be a json string.
- The port corresponding to the OpenAPI stream interface is the port corresponding to the rtsp.
- The default RTSP port is 554, and the default authentication algorithm is MD5. Can be configured through the NVR's web or GUI.
- The default audio format for the device speaker is G.711 A-law, with a sampling rate of 16000 Hz.

The details of the request and response are as follows

Client -> NVR

```
MULTITRANS rtsp://192.168.137.242/multitrans RTSP/1.0\r\n
CSeq: 1\r\n
Content-Type: application/json\r\n
Content-Length: 0\r\n
\r\n
```

NVR -> Client

```
RTSP/1.0 401 Unauthorized\r\n
CSeq: 1\r\n
Date: Thu, 25 Sep 2025 11:05:37 GMT\r\n
User-Agent: vigi\r\n
WWW-Authenticate: Digest realm="vigi", nonce="e135c0b0", stale="FALSE",
qop="auth"\r\n
WWW-Authenticate: Digest realm="vigi", nonce="e135c0b0", algorithm="SHA-256",
stale="FALSE", qop="auth"\r\n
Content-Length: 0\r\n
\r\n
```

Client -> NVR

```
MULTITRANS rtsp://192.168.137.242/multitrans RTSP/1.0\r\n
CSeq: 2\r\n
Authorization: Digest username="admin", realm="vigi", nonce="e135c0b0",
uri="rtsp://192.168.137.242/multitrans",
cnonce="1ee4caaaec697945e1a95d659af9d157", qop="auth", algorithm="MD5",
nc="00000001", response="14b0788bfc62e401ec13868c2e1d3022"\r\n
Content-Type: application/json\r\n
Content-Length: 0\r\n
\r\n
```

NVR -> Client

```
RTSP/1.0 200 OK\r\n
CSeq: 2\r\n
Date: Thu, 25 Sep 2025 11:05:37 GMT\r\n
User-Agent: vigi\r\n
Session: 0x7a59dc\r\n
Content-Type: application/json\r\n
Content-Length: 80\r\n
\r\n
{"type":"response","seq":2,"params":{"error_code":0,"session_id":"12103344"}}
```

Client -> NVR

```
MULTITRANS rtsp://192.168.137.242/multitrans RTSP/1.0\r\n
CSeq: 3\r\n
Authorization: Digest username="admin", realm="vigi", nonce="e135c0b0",
uri="rtsp://192.168.137.242/multitrans",
cnonce="1ee4caaaec697945e1a95d659af9d157", qop="auth", algorithm="MD5",
nc="00000001", response="14b0788bfc62e401ec13868c2e1d3022"\r\n
User-Agent: LibVLC/3.0.2 (LIVE555 Streaming Media v2016.11.28)\r\n
Session: 0x7a59dc\r\n
Content-Type: application/json\r\n
Content-Length: 94\r\n
\r\n
{"type":"request","seq":1,"params":{"method":"do","talk":{"mode":"half_duplex",
"channel":5}}}
```

NVR -> Client

```
RTSP/1.0 200 OK\r\n
CSeq: 3\r\n
Date: Thu, 25 Sep 2025 11:05:38 GMT\r\n
User-Agent: vigi\r\n
Session: 0x7a59dc\r\n
```

Content-Type: application/json\r\n

Content-Length: 80\r\n

\r\n

{"type": "response", "seq": 3, "params": {"error_code": 0, "session_id": "12103344"}}

[Client -> NVR](#)

Audio data