



Hik IP Receiver Device Gateway SDK

Developer Guide

Legal Information

© 2020 Hangzhou Hikvision Digital Technology Co., Ltd. All rights reserved.

This Document (hereinafter referred to be “the Document”) is the property of Hangzhou Hikvision Digital Technology Co., Ltd. or its affiliates (hereinafter referred to as “Hikvision”), and it cannot be reproduced, changed, translated, or distributed, partially or wholly, by any means, without the prior written permission of Hikvision. Unless otherwise expressly stated herein, Hikvision does not make any warranties, guarantees or representations, express or implied, regarding to the Document, any information contained herein.

LEGAL DISCLAIMER

TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, THE DOCUMENT IS PROVIDED "AS IS" AND "WITH ALL FAULTS AND ERRORS". HIKVISION MAKES NO REPRESENTATIONS OR WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO, WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NON-INFRINGEMENT. IN NO EVENT WILL HIKVISION BE LIABLE FOR ANY SPECIAL, CONSEQUENTIAL, INCIDENTAL, OR INDIRECT DAMAGES, INCLUDING, AMONG OTHERS, DAMAGES FOR LOSS OF BUSINESS PROFITS, BUSINESS INTERRUPTION OR LOSS OF DATA, CORRUPTION OF SYSTEMS, OR LOSS OF DOCUMENTATION, WHETHER BASED ON BREACH OF CONTRACT, TORT (INCLUDING NEGLIGENCE), OR OTHERWISE, IN CONNECTION WITH THE USE OF THE DOCUMENT, EVEN IF HIKVISION HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES OR LOSS.

Contents

Chapter 1 Overview	1
1.1 Introduction	1
1.2 Product Scopes	1
1.3 Update History	1
Chapter 2 Protocol	2
2.1 Operation Method	2
2.2 URL Format	2
2.3 Message Format	4
2.4 Others	5
Chapter 3 Security	6
3.1 Authentication	6
Chapter 4 Device Gateway Operations	8
4.1 Add Devices to Gateway	8
4.2 Gateway Maintenance	8
Chapter 5 Added Device Operations	9
Chapter 6 Alarm and Event	11
6.1 Subscribe Alarm/Event in Arming Mode	11
Chapter 7 Security Control	16
7.1 Typical Application Based on Device Gateway	17
7.2 Execute Video Verification via Third-Party ARC	19
Chapter 8 Request URIs	21
8.1 /ISAPI/ContentMgmt/	21
8.1.1 /ISAPI/ContentMgmt/DeviceMgmt/addDevice?format=json	21
8.1.2 /ISAPI/ContentMgmt/DeviceMgmt/channelInfo?format=json&devIndex=<uuid>	21
8.1.3 /ISAPI/ContentMgmt/DeviceMgmt/delDevice?format=json	22
8.1.4 /ISAPI/ContentMgmt/DeviceMgmt/delDevice?format=json&devIndex=<uuid>	23

8.1.5 /ISAPI/ContentMgmt/DeviceMgmt/deviceList?format=json	23
8.1.6 /ISAPI/ContentMgmt/DeviceMgmt/refreshDevInfo?format=json	24
8.2 /ISAPI/Event/	24
8.2.1 /ISAPI/Event/notification/subscribeDeviceMgmt/<ID>/devIndex/<uuid>?format=json	24
8.2.2 /ISAPI/Event/notification/subscribeDeviceMgmt/<ID>/queryStatus?format=json	25
8.2.3 /ISAPI/Event/notification/subscribeDeviceMgmt?format=json	25
8.2.4 /ISAPI/Event/notification/unSubscribeDeviceMgmt/<ID>/devIndex/<uuid>?format=json	26
8.2.5 /ISAPI/Event/notification/unSubscribeDeviceMgmt/<ID>?format=json	26
8.3 /ISAPI/Security/	27
8.3.1 /ISAPI/Security/userCheck?format=json	27
8.4 /ISAPI/SecurityCP/	27
8.4.1 /ISAPI/SecurityCP/control/arm/<ID>?ways=<string>&format=json&devIndex=<uuid>	27
8.4.2 /ISAPI/SecurityCP/control/bypass?format=json&devIndex=<uuid>	28
8.4.3 /ISAPI/SecurityCP/control/bypassRecover?format=json&devIndex=<uuid>	28
8.4.4 /ISAPI/SecurityCP/control/clearAlarm/<ID>?format=json&devIndex=<uuid>	29
8.4.5 /ISAPI/SecurityCP/control/disarm/<ID>?format=json&devIndex=<uuid>	29
8.4.6 /ISAPI/SecurityCP/control/outputs?format=json&devIndex=<uuid>	30
8.4.7 /ISAPI/SecurityCP/status/exDevStatus?format=json&devIndex=<uuid>	31
8.4.8 /ISAPI/SecurityCP/status/subSystems?format=json&devIndex=<uuid>	31
8.4.9 /ISAPI/SecurityCP/status/zones?format=json&devIndex=<uuid>	32
8.5 /ISAPI/System/	32
8.5.1 /ISAPI/System/deviceInfo?format=json	32
8.5.2 /ISAPI/System/deviceInfo?format=json&devIndex=<uuid>	33
8.5.3 /ISAPI/System/IO/outputs/<ID>/trigger?format=json&devIndex=<uuid>	34
8.5.4 /ISAPI/System/IO?format=json&devIndex=<uuid>	34

8.5.5 /ISAPI/System/Network/interfaces/<ID>/ipAddress?format=json&devIndex=<uuid>	35
8.5.6 /ISAPI/System/Network/interfaces/<ID>?format=json&devIndex=<uuid>	36
8.5.7 /ISAPI/System/Network/interfaces?format=json&devIndex=<uuid>	37
8.5.8 /ISAPI/System/reboot?format=json	37
8.5.9 /ISAPI/System/reboot?format=json&devIndex=<uuid>	37
8.5.10 /ISAPI/System/time/timeZone?devIndex=<uuid>	38
8.5.11 /ISAPI/System/time?format=json	39
8.5.12 /ISAPI/System/time?format=json&devIndex=<uuid>	40
8.6 Others	40
8.6.1 http://<ipAddress>[:port][videoUrl]	40
Chapter 9 Request and Response Messages	42
9.1 JSON_ChannelInfo	42
9.2 JSON_DelDevList	43
9.3 JSON_DeviceInfo	43
9.4 JSON_DeviceInfoList	44
9.5 JSON_DeviceInList	46
9.6 JSON_DeviceOutList	46
9.7 JSON_DevIndexList	47
9.8 JSON_DevList	47
9.9 JSON_ErrorList	47
9.10 JSON_EventNotificationAlert_AlarmEventInfo	48
9.11 JSON_EventNotificationAlert_CidAlarmMsg	49
9.12 JSON_EventNotificationAlert_DevStatusChangedAlarmMsg	51
9.13 JSON_EventNotificationAlert_HeartbeatInfo	52
9.14 JSON_EventNotificationAlert_VideoVerificationAlarmMsg	53
9.15 JSON_ExDevStatus	55
9.16 JSON_IOPortData	56

9.17 JSON_IOPortList	56
9.18 JSON_IPAddress	57
9.19 JSON_List	59
9.20 JSON_NetworkInterface	59
9.21 JSON_NetworkInterfaceList	59
9.22 JSON_OutputsCtrl	60
9.23 JSON_ResponseStatus	60
9.24 JSON_ResponseStatus_AuthenticationFailed	61
9.25 JSON_SearchDescription	61
9.26 JSON_SearchResult	62
9.27 JSON_SubscribeDevEvent	63
9.28 JSON_SubscribeDeviceMgmt	64
9.29 JSON_SubscribeDeviceMgmtRsp	64
9.30 JSON_SubscribeQueryStatusList	65
9.31 JSON_SubSysList	65
9.32 JSON_Time	66
9.33 JSON_UserCheck	66
9.34 JSON_ZoneList	67
Appendix A. Appendixes	68
A.1 Event Types	68
A.2 Error Codes	68
A.3 Hikvision CID Code	71

Chapter 1 Overview

1.1 Introduction

The Hik IP Receiver Device Gateway SDK based on a text protocol in RESTful style provides a uniform integration scheme for Hikvision products, which makes up for the integration difficulties caused by multiple open protocols, improves integration efficiency and the compatibility of Hikvision security control devices and the third-party platform. This manual mainly introduces the communication and security mechanism, gateway operations, operations (e.g., configuration, maintenance) of devices that added to Device Gateway, alarm/event subscription, security control panel application, and so on.



Note

REST (REpresentational State Transfer) is a protocol design method (named as RESTful) which abstracts all information as the resources. The abstracted resources are marked by the uniform identifies, i.e., URI (Uniform Resource Identifiers), for simple and extendable management.

1.2 Product Scopes

Table 1-1 Supported Product Type and Model

Product Name	Model	Version
Security Control Panel	DS-PWA32-HS	V1.0.4

1.3 Update History

Summary of Changes in Version 1.3_Feb., 2020

New document.

Chapter 2 Protocol

The design of Device Gateway protocol adopts RESTful style, so this part introduces the predefined resource operation methods, protocol API (URL) format, interaction message format, time format, and error processing method.

2.1 Operation Method

The resource operation methods of Device Gateway protocol are same as those of HTTP (Hyper Text Transport Protocol), see details in the following table.



Note

For details about HTTP, and HTTPS, please refer to <https://tools.ietf.org/html/rfc2612> and <https://tools.ietf.org/html/rfc2818>.

Table 2-1 HTTP Operation Method

Method	Description
POST	Create resources. This method is only available for adding resource that does not exist before.
GET	Retrieve resources. This method cannot change the system status, only return data as the response to the requester.
PUT	Update resources. This method is usually for update the resource that already exists, but it can also be used to create the resource if the specific resource does not exist.
DELETE	Delete resources.

2.2 URL Format

URL (Uniform Resource Locator) is a further class of URIs, it can identify a resource and locate the resource by describing its primary access mechanism.

The format of URL is defined as the follows: <protocol>://<host>[:port][abs_path[?query]].

protocol

Protocol types, i.e., HTTP (version 1.1), HTTPS, RTSP (version 1.0).

host

Host name, IP address, or the FQDN (Fully Qualified Domain Name) of network devices.

port

Port number of host service for listening the connection status of TCP (Transmission Control Protocol, see <https://tools.ietf.org/html/rfc793>) or UDP (User Datagram Protocol, see <https://tools.ietf.org/html/rfc768>). If this field is not configured, the default port number will be adopted. For HTTP, the default port number is 80, for HTTPS, the default port number is 443, and for RTSP, the default port number is 554.

abs_path

Resource URI: /ServiceName/ResourceType/resource. Here, the **ServiceName** is ISAPI; the **ResourceType** is predefined with upper camel case according to different functions, see details in the following table; the **resource** is defined with lower camel case and can be extended in actual applications. E.g., /ISAPI/System/deviceInfo/deviceName.

Predefined URI Model	Description
/ISAPI/System/...	System related resources
/ISAPI/Security/...	Security related resources
/ISAPI/Streaming/...	Video streaming and management related resources
/ISAPI/Event...	Event/alarm related resources
/ISAPI/PTZCtrl/...	PTZ control related resources
/ISAPI/Image/...	Video encoding and image related resources
/ISAPI/ContentMgmt/...	Storage management related resources

query

Strings for describing resources information, including related parameters. The parameter names and values must be listed as the following format in this field: ?p1=v1&p2=v2&...&pn=vn.



Note

- To locate the connected device, when operating lower-level device via the URL, the **query** field should be filled as ?devIndex=uuid&p1=v1&p2=v2&...&pn=vn. The uuid (or guid) is a 32-byte (128 bits) random number, which is unique and generated by operating system when adding device, and its format is "xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx".
- For message in JSON format, the **query** field should be filled as ?format=json&p1=v1&p2=v2&...&pn=vn. For details about message format, refer to the next section below. E.g., <http://10.17.132.22/ISAPI/System/time?format=json&devIndex=550e8400e29b41d4a716446655440000>.

2.3 Message Format

For Device Gateway SDK, the request and response messages generated among the interaction of Device Gateway, devices, and system are data in JSON format.

Note

The message format here is only available for URLs based on HTTP.

- The leaf node (without any sub node) in the message is named by lower camel case, while the non-leaf node in the message is named by upper camel case.
 - To communicate by the messages in JSON format, the devices must support the public specifications in <http://www.ecma-international.org/publications/files/ECMA-ST/ECMA-404.pdf> and HTTP with version 1.1.
-

Note

JSON is a lightweight data format which is a subset of JavaScript language and is small, fast, and easy to be parsed.

- Generally, for configuration information, the **Content-Type** of message is "application/json", see the example below:

```
//Request message
GET /ISAPI/System/deviceInfo?format=json HTTP/1.1
...

//Response message
HTTP/1.1 200 OK
...
Content-Type: application/json
...
"DeviceStatus":""
...
```

For data (e.g., firmware, configuration files), the **Content-Type** of message is "application/octet-stream", see the example below:

```
//Request message
POST /ISAPI/System/streamMedia?
format=json&devIndex=550e8400e29b41d4a716446655440000 HTTP/1.1
...
Content-Type: application/octet-stream
...
[proprietary configuration file data content]

//Response message
HTTP/1.1 200 OK
...
Content-Type: application/json
...
```

```
"ResponseStatus":""  
...
```

2.4 Others

Time Format

The time format in the Device Gateway SDK adopts ISO8601 standard (see details in <http://www.w3.org/TR/NOTE-datetime-970915>), that is, YYYY-MM-DDThh:mm:ss.sTZD (e.g., 2017-08-16T20:17:06+08:00, 2017-08-16T20:09:06Z).

Error Processing

During the integration applications of Device Gateway SDK, when the error of URL based on HTTP occurred, the **JSON_ResponseStatus** message which contains error code will be returned. If the error of URL based on RTSP occurs, the corresponding status code will directly be returned, for details, refer to <https://tools.ietf.org/html/rfc2326> .



Note

For batch operations, if some operations are failed, both the **JSON_ResponseStatus** message and failure details message (see details in actual batch operations, such as adding devices in batch, deleting devices in batch, and so on) will be returned to notify which operations are failed and the failure reasons.

Chapter 3 Security

This part mainly introduces the authentication, user permission, and encryption in the integration applications of Device Gateway SDK.

3.1 Authentication

When communicating via Device Gateway SDK, the digest of the session must be authenticated.



Note

- The authentication must be based on *HTTP Authentication: Basic and Digest Access Authentication*, see <https://tools.ietf.org/html/rfc2617> for details.
- The request session must contain authentication information, otherwise, the device will return 401 error code.
- For login authentication, the available URL is GET `/ISAPI/Security/userCheck?format=json`.

The message digest, which contains user name, password, specific nonce value, HTTP or RTSP operation methods, and request URL, is generated by the MD5 algorithm, see the calculation rules below.

qop=Undefined

Digest=MD5(MD5(A1):<nonce>:MD5(A2))

qop="auth:"

Digest=MD5(MD5(A1):<nonce>:<nc>:<cnonce>:<qop>:MD5(A2))

qop="auth-int:"

Digest=MD5(MD5(A1):<nonce>:<nc>:<cnonce>:<qop>:MD5(A2))



Note

- The **qop** is a value for determining whether the authentication is required.
- A1 and A2 are two data blocks required for digest calculation.
A1: Data block about security, which contains user name, password, security domain, random number, and so on. If the digest calculation algorithm is MD5, A1=<user>:<realm>:<password>; if the algorithm is MD5-sess, A1=MD5(<user>:<realm>:<password>):<nonce>:<cnonce>.
A2: Data block about message, such as URL, repeated requests, message body, and so on, it helps to prevent repeated, and realize the resource/message tamper-proof. If the **qop** is not defined or it is "auth:", A2=<request-method>:<uri-directive-value>; if the **qop** is "auth-int:", A2=<request-method>:<uri-directive-value>:MD5(<request-entity-body>).
- The **nonce** is the random number generated by service, the following generation formula is suggested: nonce = BASE64(time-stamp MD5(time-stamp ":" ETag ":" private-key)). The **time-stamp** in the formula is the time stamp generated by service or the unique serial No.; the **ETag** is

the value of HTTP ETag header in the request message; the **private-key** is the data that only known by service.

If authentication failed, the device will return the ***JSON_ResponseStatus_AuthenticationFailed*** message, and the remaining authentication attempts will also be returned. If the remaining attempts is 0, the user will be locked at the next authentication attempt.

Chapter 4 Device Gateway Operations

4.1 Add Devices to Gateway

You can add a device or multiple devices to Device Gateway by the device information (i.e., device ID and device key). After adding devices, you can also get the added device list.

Steps

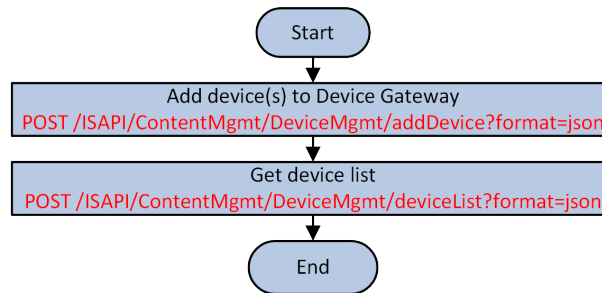


Figure 4-1 Programming Flow of Adding Devices to Gateway

1. Call **`/ISAPI/ContentMgmt/DeviceMgmt/addDevice?format=json`** by POST method to add device(s) to Device Gateway.
2. Call **`/ISAPI/ContentMgmt/DeviceMgmt/deviceList?format=json`** by POST method to get the list of the added devices.

4.2 Gateway Maintenance

Table 4-1 Information, Reboot, and Time



Function	URI
Get or set Device Gateway information	GET or PUT <code>/ISAPI/System/deviceInfo?format=json</code>
Reboot Device Gateway	PUT <code>/ISAPI/System/reboot?format=json</code>
Get or set time synchronization parameters of Device Gateway	GET or PUT <code>/ISAPI/System/time?format=json</code>

Chapter 5 Added Device Operations

Device Management

Function	URI
Get a device's information	GET <i>/ISAPI/System/deviceInfo?format=json&devIndex=<uuid></i>
Get multiple devices' information in batch	POST <i>/ISAPI/ContentMgmt/DeviceMgmt/refreshDevInfo?format=json</i>
Delete a device	DELETE <i>/ISAPI/ContentMgmt/DeviceMgmt/delDevice?format=json&devIndex=<uuid></i>
Delete multiple devices in batch	POST <i>/ISAPI/ContentMgmt/DeviceMgmt/delDevice?format=json</i>
Edit a device's information	PUT <i>/ISAPI/System/deviceInfo?format=json&devIndex=<uuid></i>
Get channel information of a device	GET <i>/ISAPI/ContentMgmt/DeviceMgmt/channelInfo?format=json&devIndex=<uuid></i>

Network Configuration

Function	URI
Get all network interfaces' information	GET <i>/ISAPI/System/Network/interfaces?format=json&devIndex=<uuid></i>  Note As the network interface is the hardware resource, so it cannot be created or deleted.
Get or set a specific network interface's information (including IP address)	GET or PUT <i>/ISAPI/System/Network/interfaces/<ID>?format=json&devIndex=<uuid></i>  Note By default, the device connects to the network, and it contains more than one network interface.
Get/set a specific network interface's IP address	GET or PUT <i>/ISAPI/System/Network/interfaces/<ID>/ipAddress?format=json&devIndex=<uuid></i>

System Maintenance

Function	URI
Reboot device added to Device Gateway	PUT <i>/ISAPI/System/reboot?format=json&devIndex=<uuid></i>
Set time zone parameters of devices added to Device Gateway	PUT <i>/ISAPI/System/time/timeZone?devIndex=<uuid></i>
Get time synchronization parameters of devices added to Device Gateway	GET <i>/ISAPI/System/time?format=json&devIndex=<uuid></i>

Alarm Input and Output Configuration

Function	URI
Manually trigger alarm output of device	PUT <i>/ISAPI/System/IO/outputs/<ID>/trigger?format=json&devIndex=<uuid></i>
Get alarm input and output information of device	GET <i>/ISAPI/System/IO?format=json&devIndex=<uuid></i>

Chapter 6 Alarm and Event

You can arm the devices via the Device Gateway in arming mode, and the devices will upload the alarm or event information to Device Gateway if the alarm or event is triggered or occurred. And then the third-party system can send the subscription conditions to multiple devices for subscribing different alarms or events via Device Gateway by setting up only one connection.

6.1 Subscribe Alarm/Event in Arming Mode

For arming mode, the system will connect to the devices automatically and send commands to the devices for uploading alarm/event information when the alarm is triggered or event occurred. In this case, multiple connections are built among the third-party system, Device Gateway, and devices, so the arming parameters (including alarm/event types to be subscribed, linkage actions, arming schedule, and so on) or alarm/event information will be sent or uploaded via the Device Gateway.

Before You Start

Make sure you have configured the alarm parameters, such as arming schedule, linkage actions (set to "center"), and so on, to arm the device.

Steps

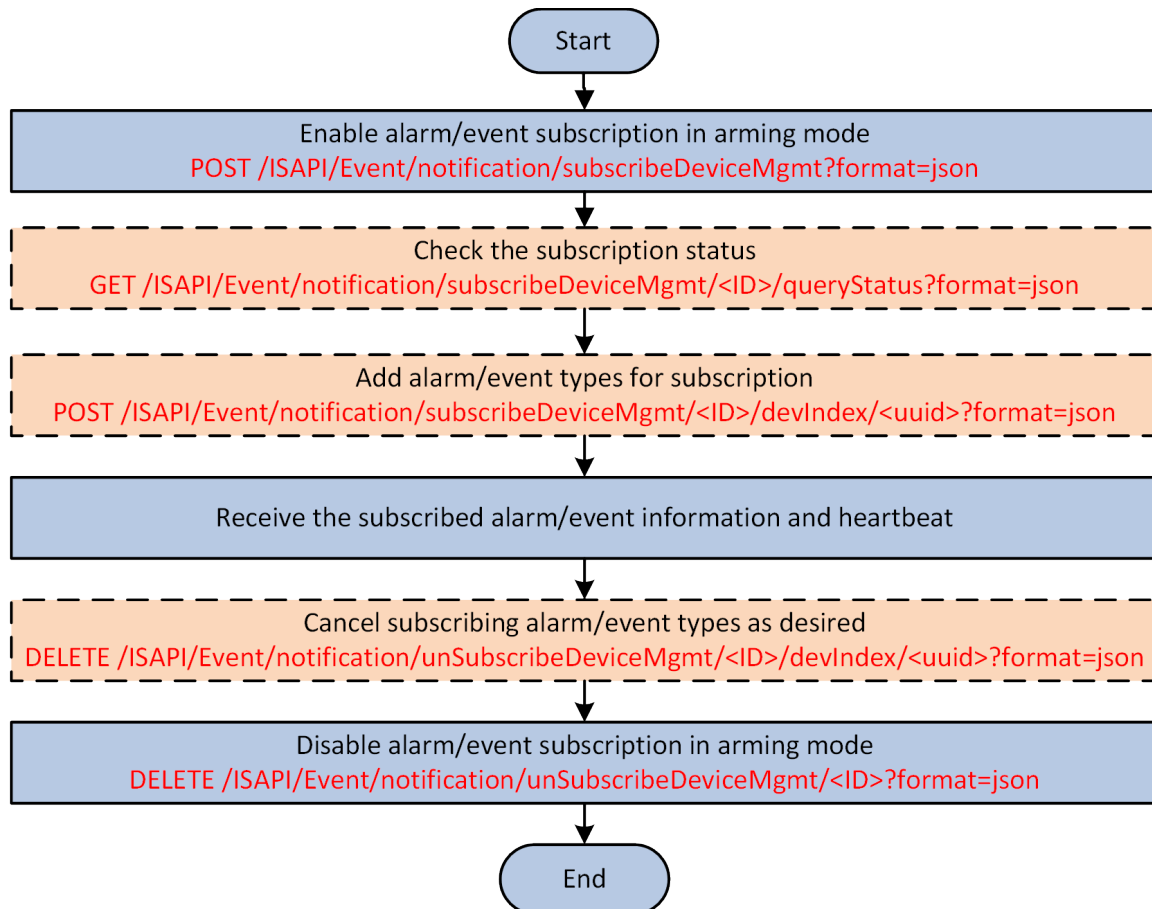


Figure 6-1 Programming Flow of Subscribing Alarm/Event in Arming Mode

Note

We have provided a sample code file (SubscribeAlarm.csproj) of alarm or event subscription in the current directory of this document for reference.

1. Call **`/ISAPI/Event/notification/subscribeDeviceMgmt?format=json`** by POST method to enable alarm/event subscription in arming mode.

Note

- You can set the alarm/event types to be subscribed in the request message of this URL.
- Refer to **Event Types** for the supported alarm or event types.

2. **Optional:** Call **`/ISAPI/Event/notification/subscribeDeviceMgmt/<ID>/queryStatus?format=json`** by GET method to check the subscription status.

Note

This step is not supported when the subscription mode (**eventMode**) is "all".

3. **Optional:** Call `/ISAPI/Event/notification/subscribeDeviceMgmt/<ID>/devIndex/<uuid>?format=json` by POST method to add alarm/event types for subscription.



This step is not supported when the subscription mode (**eventMode**) is "all".

4. When the alarm is triggered or event occurred, receive the subscribed alarm/event information and heartbeat from the messages **`JSON_EventNotificationAlert_AlarmEventInfo`** and **`JSON_EventNotificationAlert_HeartbeatInfo`** uploaded by device via Device Gateway.



- If receiving alarm/event information exception or receiving heartbeat timed out (the default heartbeat interval is 10 seconds, and the suggested timeout is 30 seconds), you should call `/ISAPI/Event/notification/subscribeDeviceMgmt?format=json` by POST to rebuild the link.
 - The alarm or event information messages vary with different alarm or event types, refer to the corresponding alarm or event configuration chapters or get the message examples in **`JSON_EventNotificationAlert_AlarmEventInfo`**.
-

5. **Optional:** Call `/ISAPI/Event/notification/unSubscribeDeviceMgmt/<ID>/devIndex/<uuid>?format=json` by DELETE method to cancel subscribing alarm/event types as desired.



This step is not supported when the subscription mode (**eventMode**) is "all".

6. Call `/ISAPI/Event/notification/unSubscribeDeviceMgmt/<ID>?format=json` by DELETE to disable alarm/event subscription in arming mode.

Example

Message Example of Subscribing Alarm/Event in Arming Mode

Request Message

```
POST /ISAPI/Event/notification/subscribeDeviceMgmt?format=json HTTP/1.1
Host: 127.0.0.1:80
X-Forwarded-For: 127.0.0.1
X-Real-Port: 57946
Connection: Keep-Alive
Content-Length: 63
Content-Type: application/x-www-form-urlencoded
Authorization: Digest username="admin",
realm="DS-GWAS0101(6419)",
nonce="4e546c6c596a55355a5749364e5752684f4445344d57593d",
uri="/ISAPI/Event/notification/subscribeDeviceMgmt",
cnonce="7e867c78d9874aa904b489a6791aaeaf",
nc=00000001,
qop="auth",
response="fa7d3d95762b6e10c8a18f09f1f61e81"

{
  "SubscribeDeviceMgmt": {
```

```
"eventMode":"all",
"defenceMode":"all"
}
}
```

Response Message

```
HTTP/1.1 200 OK
Server: nginx
Date: Wed, 26 Dec 2018 11:38:38 GMT
Content-Type: multipart/mixed; boundary=boundary
Connection: close
MIME-Version: 1.0

--boundary
Content-Type: application/json; charset="UTF-8"
Content-Length: 39

{"SubscribeDeviceMgmtRsp":{"id":5939}}
--boundary
Content-Type: application/json; charset="UTF-8"
Content-Length: 415

{
  "EventNotificationAlert":{
    "channelID": "1",
    "dateTime": "2018-01-21T12:50:39+08:00",
    "activePostCount": 1,
    "eventType": "devStatusChanged",
    "eventState": "active",
    "eventDescription": "device status about online or offline changed",
    "devIndex":"2cd6716d-767f-4756-ac55-50276a5e3b4a",
    "channelName": "Device1",
    "status":"online",
  }
}
--boundary
Content-Type: application/json; charset="UTF-8"
Content-Length: 183

{
  "EventNotificationAlert":{
    "dateTime":"2018-12-26T19:39:21+08:00",
    "activePostCount":1,
    "eventType":"heartBeat",
    "eventState":"active",
    "eventDescription":"Heart Beat"
  }
}
```

Note

We have also provided a sample code file (SubscribeAlarm.csproj) of alarm/event subscription in the current directory of this document for reference.

Chapter 7 Security Control

A security control device detects people, vehicles, etc., entering a predefined region, triggers events and alarms, and reports events/alarms information (such as location) to security personnel. The region can be classified as zone and partition: for zone, it refers to a protection area and is regarded as the maximum recognizable unit to distinguish the alarm event; for partition, it is an independent control system of a security control device, allows you to batch arm or disarm all zones in it.

Control Alarm Device

Function	URI
Arm partition	PUT /ISAPI/SecurityCP/control/arm/<ID>?ways=<string>&format=json&devIndex=<uuid>
Disarm partition	PUT /ISAPI/SecurityCP/control/disarm/<ID>?format=json&devIndex=<uuid>
Clear partition's alarms	PUT /ISAPI/SecurityCP/control/clearAlarm/<ID>?format=json&devIndex=<uuid>
Bypass zone in a batch	PUT /ISAPI/SecurityCP/control/bypass?format=json&devIndex=<uuid>
Recover zone bypass in a batch	PUT /ISAPI/SecurityCP/control/bypassRecover?format=json&devIndex=<uuid>
Control relay	POST /ISAPI/SecurityCP/control/outputs?format=json&devIndex=<uuid>

Get Real-Time Status of Alarm Device

Function	URI
Get all partitions' statuses	GET /ISAPI/SecurityCP/status/subSystems?format=json&devIndex=<uuid>
Get all zones' statuses	GET /ISAPI/SecurityCP/status/zones?format=json&devIndex=<uuid>
Get all peripherals' statuses	GET /ISAPI/SecurityCP/status/exDevStatus?format=json&devIndex=<uuid>

7.1 Typical Application Based on Device Gateway

The Device Gateway provides protocol conversion function for the third-party Alarm Receiving Center (ARC) to connect to and manage the AX security control panels as the supporting protocol types of center and panel are different. And then the third-party ARC can receive the alarms or events of Hikvision security control panel.

The Device Gateway connects to AX security control panel based on ISUP (Intelligent Security Uplink Protocol) 5.0, and connects to ARC via Device Gateway protocol. And then the communication and interaction between AX security control panel and ARC will be built on the protocol conversion function of Device Gateway, refer to the application framework below.

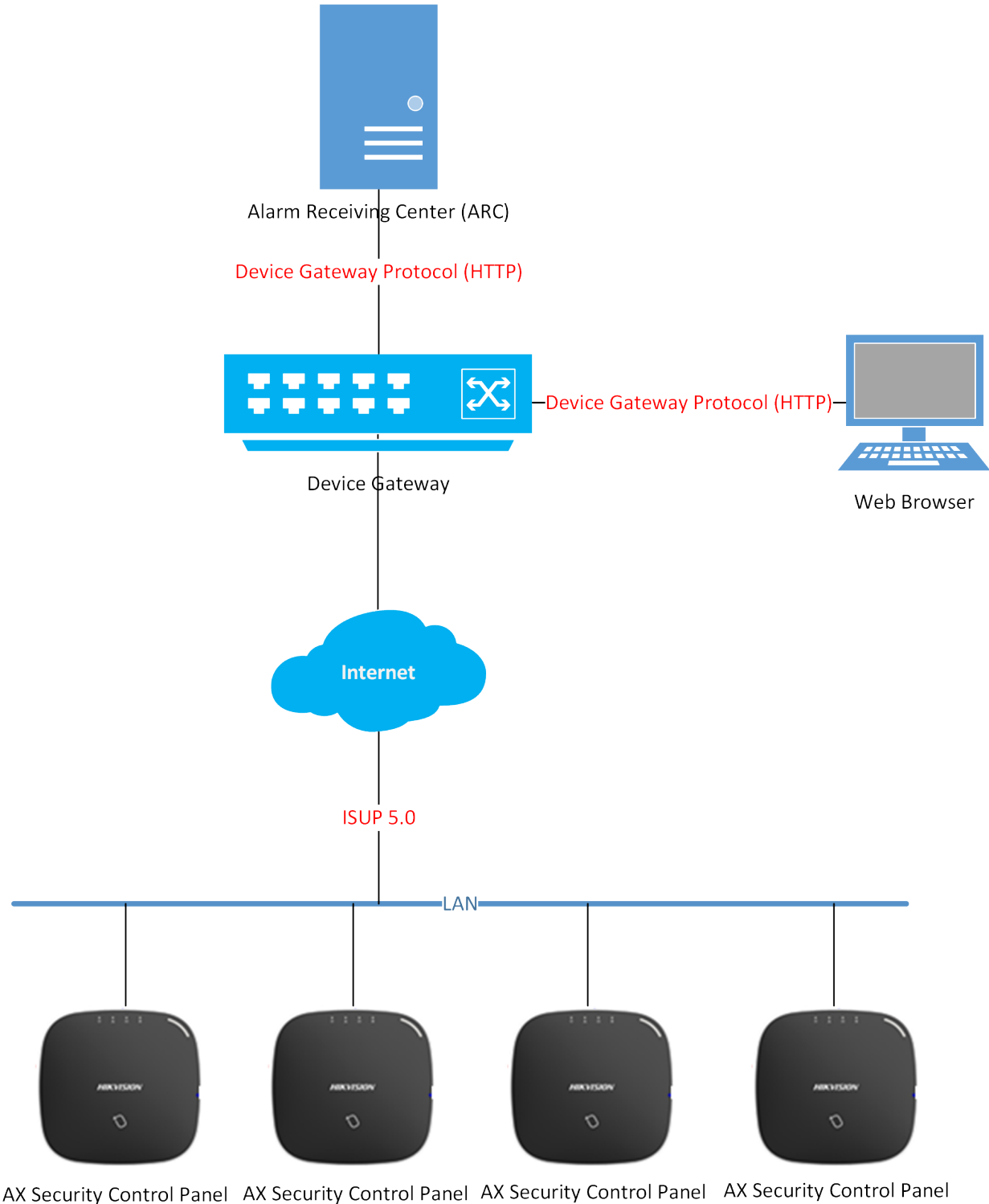


Figure 7-1 Typical Application Based on Device Gateway

7.2 Execute Video Verification via Third-Party ARC

The video verification is a main application between AX security control panel and third-party Alarm Receiving Center (ARC), which can compare the videos of a channel before and after triggering alarm, to restore the alarm scene for fault locating and alarm analysis.

Steps

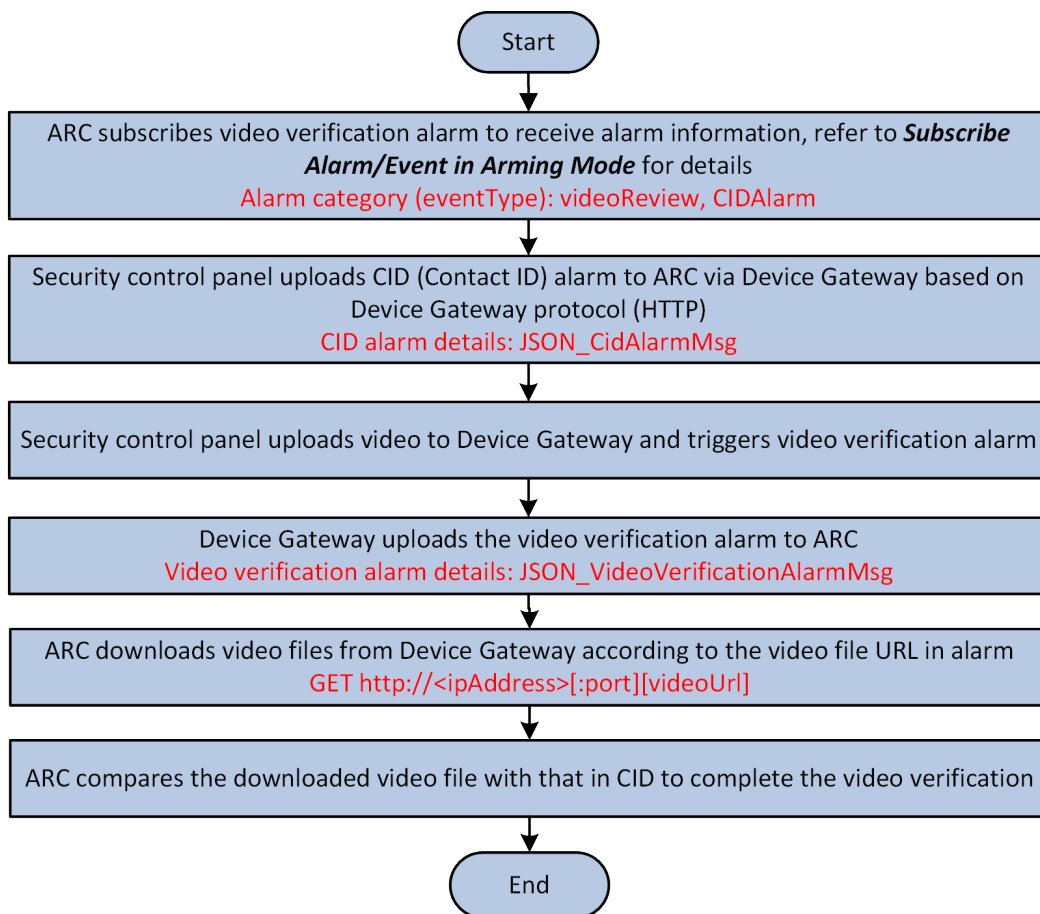


Figure 7-2 Progress of Executing Video Verification via Third-Party ARC

Note

We have provided a sample code file (VideoVerification.csproj) of video verification in the current directory of this document for reference.

1. The ARC sets alarm category (**eventType**) to "videoReview" or "CIDAlarm" for subscribing video verification alarm or CID alarm, refer to **Subscribe Alarm/Event in Arming Mode** for details.
2. The security control panel uploads CID (Contact ID) alarm to ARC via Device Gateway based on Device Gateway protocol (or HTTP).

The CID alarm details is uploaded in the message **JSON_EventNotificationAlert_CidAlarmMsg**.

3. The security control panel uploads alarm video to Device Gateway if any alarm is triggered, and then triggers the video verification alarm.
4. The Device Gateway uploads the triggered video verification alarm to ARC.
The video verification alarm details is uploaded in the message ***JSON_EventNotificationAlert_VideoVerificationAlarmMsg*** .
5. The ARC calls ***http://<ipAddress>[:port][videoUrl]*** by GET method to download alarm video files from Device Gateway according to the file URL in alarm.
6. The ARC compares the downloaded video file with that in CID alarm to complete the video verification.

Chapter 8 Request URIs

8.1 /ISAPI/ContentMgmt/

8.1.1 /ISAPI/ContentMgmt/DeviceMgmt/addDevice?format=json

Add device(s).

Request URL Definition

Table 8-1 POST /ISAPI/ContentMgmt/DeviceMgmt/addDevice?format=json

Method	POST
Description	Add device(s).
Query	security : the version No. of encryption scheme. When security does not exist, it indicates that the data is not encrypted; when security is 1, it indicates that the nodes of sensitive information in the message are encrypted in AES128 CBC mode. iv : the initialization vector, and it is required when security is 1. format : determine the format of request or response message.
Request	<i>JSON_DeviceInList</i>
Response	Succeeded: <i>JSON_DeviceOutList</i> Failed: <i>JSON_ResponseStatus</i> and <i>JSON_DeviceOutList</i>

Remarks

For bulk operations, "succeeded" only indicates that one, multiple, or all operations are succeeded, refer to the **Device** field in the *JSON_DeviceOutList* message for the detailed successful operation(s); "failed" indicates that all operations are failed, you can also refer to the **Device** field in the *JSON_DeviceOutList* message for the error code (i.e., 0x20000009 or 0x400000a0) or description (see *Error Codes* for details).

8.1.2 /ISAPI/ContentMgmt/DeviceMgmt/channelInfo? format=json&devIndex=<uuid>

Get the channel information of the specific device.

Request URL Definition

Table 8-2 GET /ISAPI/ContentMgmt/DeviceMgmt/channelInfo?format=json&devIndex=<uuid>

Method	GET
Description	Get the channel information of the specific device.
Query	format: determine the format of request or response message. devIndex: a uuid or guid (a 32-byte or 128-bit random number) for locating the connected or lower-level devices, which is unique and generated by operating system when adding device, and its format is "xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx".
Request	None.
Response	Succeeded: <i>JSON_ChannelInfo</i> Failed: <i>JSON_ResponseStatus</i>

Remarks

The <uuid> in the URI refers to the device ID generated by operating system, see details in **URL Format**.

8.1.3 /ISAPI/ContentMgmt/DeviceMgmt/delDevice?format=json

Delete devices in batch.

Request URL Definition

Table 8-3 POST /ISAPI/ContentMgmt/DeviceMgmt/delDevice?format=json

Method	POST
Description	Delete devices in batch.
Query	format: determine the format of request or response message.
Request	<i>JSON_DevIndexList</i>
Response	Succeeded: <i>JSON_DelDevList</i> Failed: <i>JSON_ResponseStatus</i> and <i>JSON_DelDevList</i>

Remarks

For bulk operations, "succeeded" only indicates that one, multiple, or all operations are succeeded, refer to the **Device** field in the *JSON_DelDevList* message for the detailed successful operation(s); "failed" indicates that all operations are failed, you can also refer to the **Device** field in the *JSON_DelDevList* message for the error code or description (see **Error Codes** for details).

8.1.4 /ISAPI/ContentMgmt/DeviceMgmt/delDevice? format=json&devIndex=<uuid>

Delete a device.

Request URI Definition

Table 8-4 DELETE /ISAPI/ContentMgmt/DeviceMgmt/delDevice?format=json&devIndex=<uuid>

Method	DELETE
Description	Delete a device.
Query	format: determine the format of request or response message. devIndex: a uuid or guid (a 32-byte or 128-bit random number) for locating the connected or lower-level devices, which is unique and generated by operating system when adding device, and its format is "xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx".
Request	None
Response	<i>JSON_ResponseStatus</i>

Remarks

The <uuid> in the URI refers to the device ID generated by operating system, see details in **URL Format**.

8.1.5 /ISAPI/ContentMgmt/DeviceMgmt/deviceList?format=json

Search for the added device list.

Request URL Definition

Table 8-5 POST /ISAPI/ContentMgmt/DeviceMgmt/deviceList?format=json

Method	POST
Description	Search for the added device list.
Query	format: determine the format of request or response message.
Request	<i>JSON_SearchDescription</i>
Response	<i>JSON_SearchResult</i>

8.1.6 /ISAPI/ContentMgmt/DeviceMgmt/refreshDevInfo?format=json

Get device basic information in batch.

Request URL Definition

Table 8-6 POST /ISAPI/ContentMgmt/DeviceMgmt/refreshDevInfo?format=json

Method	POST
Description	Get device basic information in batch.
Query	format : determine the format of request or response message.
Request	<i>JSON_DevList</i>
Response	Succeeded: <i>JSON_DeviceInfoList</i> Failed: <i>JSON_ResponseStatus</i>

8.2 /ISAPI/Event/

8.2.1 /ISAPI/Event/notification/subscribeDeviceMgmt/<ID>/devIndex/<uuid>?format=json

Operations about alarm/event subscription of a specific device.

Request URL Definition

Table 8-7 GET /ISAPI/Event/notification/subscribeDeviceMgmt/<ID>/devIndex/<uuid>?format=json

Method	GET
Description	Get the subscribed alarm/event types of the specific device.
Query	format : determine the format of request or response message.
Request	None.
Response	<i>JSON_SubscribeDevEvent</i>

Table 8-8 POST /ISAPI/Event/notification/subscribeDeviceMgmt/<ID>/devIndex/<uuid>?format=json

Method	POST
Description	Add alarm/event types of the specific device to be subscribed.
Query	format : determine the format of request or response message.

Request	<i>JSON_SubscribeDevEvent</i>
Response	<i>JSON_ResponseStatus</i>

Remarks

The **ID** in the URL is the subscription ID, and the **uuid** is the device ID.

8.2.2 /ISAPI/Event/notification/subscribeDeviceMgmt/<ID>/queryStatus?format=json

Get the alarm/event subscription status.

Request URL Definition

Table 8-9 GET /ISAPI/Event/notification/subscribeDeviceMgmt/<ID>/queryStatus?format=json

Method	GET
Description	Get the alarm/event subscription status.
Query	format: determine the format of request or response message.
Request	None.
Response	<i>JSON_SubscribeQueryStatusList</i>

Remarks

The **<ID>** in the request URL refers to the arming connection ID.

8.2.3 /ISAPI/Event/notification/subscribeDeviceMgmt?format=json

Enable subscribing alarm/event in arming mode.

Request URL Definition

Table 8-10 POST /ISAPI/Event/notification/subscribeDeviceMgmt?format=json

Method	POST
Description	Enable subscribing alarm/event in arming mode.
Query	format: determine the format of request or response message.
Request	<i>JSON_SubscribeDeviceMgmt</i>
Response	Succeeded: <i>JSON_SubscribeDeviceMgmtRsp</i> + <i>JSON_EventNotificationAlert_AlarmEventInfo</i> or <i>JSON_EventNotificationAlert_HeartbeatInfo</i>

	Failed: <i>JSON_ResponseStatus</i>
--	------------------------------------

Remarks

The *JSON_EventNotificationAlert_AlarmEventInfo* and *JSON_EventNotificationAlert_HeartbeatInfo* are uploaded repeatedly.

8.2.4 /ISAPI/Event/notification/unSubscribeDeviceMgmt/<ID>/devIndex/<uuid>?format=json

Cancel subscribing the alarm/event types of a specific device.

Request URL Definition

Table 8-11 DELETE /ISAPI/Event/notification/unSubscribeDeviceMgmt/<ID>/devIndex/<uuid>?format=json

Method	DELETE
Description	Cancel subscribing the alarm/event types of a specific device.
Query	format: determine the format of request or response message.
Request	None.
Response	<i>JSON_ResponseStatus</i>

Remarks

The **ID** in the URL is the subscription ID, which is returned by device; the **uuid** is the device ID.

8.2.5 /ISAPI/Event/notification/unSubscribeDeviceMgmt/<ID>?format=json

Disable alarm/event subscription in arming mode.

Request URL Definition

Table 8-12 DELETE /ISAPI/Event/notification/unSubscribeDeviceMgmt/<ID>?format=json

Method	DELETE
Description	Disable alarm/event subscription in arming mode.
Query	format: determine the format of request or response message.
Request	None.
Response	<i>JSON_ResponseStatus</i>

Remarks

The **ID** in the URL is the subscription ID, which is returned by device.

8.3 /ISAPI/Security/

8.3.1 /ISAPI/Security/userCheck?format=json

Verify the user after logging in to the device.

Request URL Definition

Table 8-13 GET /ISAPI/Security/userCheck?format=json

Method	GET
Description	Verify the user after logging in to the device.
Query	format : determine the format of request or response message.
Request	None.
Response	<i>JSON_UserCheck</i>

Remarks

If login succeeded, *HTTP 200/OK* will be returned; if login failed, *HTTP 401/Unauthorized* will be returned.

8.4 /ISAPI/SecurityCP/

8.4.1 /ISAPI/SecurityCP/control/arm/<ID>? ways=<string>&format=json&devIndex=<uuid>

Arm the partition.

Request URL Definition

Table 8-14 PUT /ISAPI/SecurityCP/control/arm/<ID>?
ways=<string>&format=json&devIndex=<uuid>

Method	PUT
Description	Arm the partition.

Query	ways: arming mode, "stay"-stay arming and "away"-away arming are available. format: determine the format of request or response message. devIndex: a uuid or guid (a 32-byte or 128-bit random number) for locating the connected or lower-level devices, which is unique and generated by operating system when adding device, and its format is "xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx".
Request	None.
Response	<i>JSON_ResponseStatus</i>

Remarks

The <ID> in the request URL refers to the partition No., which starts from 1. If ID equals to 0xffffffff, all partitions are selected.

8.4.2 /ISAPI/SecurityCP/control/bypass?format=json&devIndex=<uuid>

Bypass zones in batch.

Request URL Definition

Table 8-15 PUT /ISAPI/SecurityCP/control/bypass?format=json&devIndex=<uuid>

Method	PUT
Description	Bypass zones in batch.
Query	format: determine the format of request or response message. devIndex: a uuid or guid (a 32-byte or 128-bit random number) for locating the connected or lower-level devices, which is unique and generated by operating system when adding device, and its format is "xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx".
Request	<i>JSON_List</i>
Response	<i>JSON_ResponseStatus</i>

8.4.3 /ISAPI/SecurityCP/control/bypassRecover?format=json&devIndex=<uuid>

Recover bypass of zones in batch.

Request URL Definition

Table 8-16 PUT /ISAPI/SecurityCP/control/bypassRecover?format=json&devIndex=<uuid>

Method	PUT
Description	Recover bypass of zones in batch.
Query	format: determine the format of request or response message. devIndex: a uuid or guid (a 32-byte or 128-bit random number) for locating the connected or lower-level devices, which is unique and generated by operating system when adding device, and its format is "xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx".
Request	<i>JSON_List</i>
Response	<i>JSON_ResponseStatus</i>

8.4.4 /ISAPI/SecurityCP/control/clearAlarm/<ID>?format=json&devIndex=<uuid>

Clear the alarms.

Request URI Definition

Table 8-17 PUT /ISAPI/SecurityCP/control/clearAlarm/<ID>?format=json&devIndex=<uuid>

Method	PUT
Description	Clear the alarms.
Query	format: determine the format of request or response message. devIndex: a uuid or guid (a 32-byte or 128-bit random number) for locating the connected or lower-level devices, which is unique and generated by operating system when adding device, and its format is "xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx".
Request	None
Response	<i>JSON_ResponseStatus</i>

Remarks

The <ID> in the request URI refers to the partition No., if the value of ID is 0xffffffff, it indicates all partitions.

8.4.5 /ISAPI/SecurityCP/control/disarm/<ID>?format=json&devIndex=<uuid>

Disarm the partition.

Request URL Definition

Table 8-18 PUT /ISAPI/SecurityCP/control/disarm/<ID>?format=json&devIndex=<uuid>

Method	PUT
Description	Disarm the partition.
Query	format: determine the format of request or response message. devIndex: a uuid or guid (a 32-byte or 128-bit random number) for locating the connected or lower-level devices, which is unique and generated by operating system when adding device, and its format is "xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx".
Request	None.
Response	<i>JSON_ResponseStatus</i>

Remarks

The <ID> in the request URL refers to the partition No., which starts from 1. If ID equals to 0xffffffff, all partitions are selected.

8.4.6 /ISAPI/SecurityCP/control/outputs?format=json&devIndex=<uuid>

Control relays in batch.

Request URL Definition

Table 8-19 POST /ISAPI/SecurityCP/control/outputs?format=json&devIndex=<uuid>

Method	POST
Description	Control relays in batch.
Query	format: determine the format of request or response message. devIndex: a uuid or guid (a 32-byte or 128-bit random number) for locating the connected or lower-level devices, which is unique and generated by operating system when adding device, and its format is "xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx".
Request	<i>JSON_OutputsCtrl</i>
Response	Succeeded: <i>JSON_ResponseStatus</i> Failed: <i>JSON_ErrorList</i>

8.4.7 /ISAPI/SecurityCP/status/exDevStatus?format=json&devIndex=<uuid>

Get statuses of all peripherals.

Request URL Definition

Table 8-20 GET /ISAPI/SecurityCP/status/exDevStatus?format=json&devIndex=<uuid>

Method	GET
Description	Get statuses of all peripherals.
Query	format: determine the format of request or response message. devIndex: a uuid or guid (a 32-byte or 128-bit random number) for locating the connected or lower-level devices, which is unique and generated by operating system when adding device, and its format is "xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx".
Request	None.
Response	Succeeded: <i>JSON_ExDevStatus</i> Failed: <i>JSON_ResponseStatus</i>

8.4.8 /ISAPI/SecurityCP/status/subSystems?format=json&devIndex=<uuid>

Get statuses of all partitions.

Request URL Definition

Table 8-21 GET /ISAPI/SecurityCP/status/subSystems?format=json&devIndex=<uuid>

Method	GET
Description	Get statuses of all partitions.
Query	format: determine the format of request or response message. devIndex: a uuid or guid (a 32-byte or 128-bit random number) for locating the connected or lower-level devices, which is unique and generated by operating system when adding device, and its format is "xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx".
Request	None.
Response	Succeeded: <i>JSON_SubSysList</i> Failed: <i>JSON_ResponseStatus</i>

8.4.9 /ISAPI/SecurityCP/status/zones?format=json&devIndex=<uuid>

Get statuses of all zones.

Request URL Definition

Table 8-22 GET /ISAPI/SecurityCP/status/zones?format=json&devIndex=<uuid>

Method	GET
Description	Get statuses of all zones.
Query	format: determine the format of request or response message. devIndex: a uuid or guid (a 32-byte or 128-bit random number) for locating the connected or lower-level devices, which is unique and generated by operating system when adding device, and its format is "xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx".
Request	None.
Response	Succeeded: <i>JSON_ZoneList</i> Failed: <i>JSON_ResponseStatus</i>

8.5 /ISAPI/System/

8.5.1 /ISAPI/System/deviceInfo?format=json

Operations about Device Gateway configurations.

Request URL Definition

Table 8-23 GET /ISAPI/System/deviceInfo?format=json

Method	GET
Description	Get the Device Gateway configurations.
Query	format: determine the format of request or response message.
Request	None.
Response	<i>JSON_DeviceInfo</i>

Table 8-24 PUT /ISAPI/System/deviceInfo?format=json

Method	PUT
Description	Set the Device Gateway configurations.

Query	format: determine the format of request or response message.
Request	<i>JSON_DeviceInfo</i>
Response	<i>JSON_ResponseStatus</i>

Remarks

For PUT operation method, the read-only fields in the *JSON_DeviceInfo* message are invalid.

8.5.2 /ISAPI/System/deviceInfo?format=json&devIndex=<uuid>

Operations about configurations of devices connected to Device Gateway.

Request URL Definition

Table 8-25 GET /ISAPI/System/deviceInfo?format=json&devIndex=<uuid>

Method	GET
Description	Get the configurations of devices connected to Device Gateway.
Query	format: determine the format of request or response message. devIndex: a uuid or guid (a 32-byte or 128-bit random number) for locating the connected or lower-level devices, which is unique and generated by operating system when adding device, and its format is "xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx".
Request	None.
Response	<i>JSON_DeviceInfo</i>

Table 8-26 PUT /ISAPI/System/deviceInfo?format=json&devIndex=<uuid>

Method	PUT
Description	Set the configurations of devices connected to Device Gateway.
Query	format: determine the format of request or response message. devIndex: a uuid or guid (a 32-byte or 128-bit random number) for locating the connected or lower-level devices, which is unique and generated by operating system when adding device, and its format is "xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx".
Request	<i>JSON_DeviceInfo</i>
Response	<i>JSON_ResponseStatus</i>

Remarks

- In PUT operation, the read-only fields in the *JSON_DeviceInfo* message is invalid.
- The <uuid> in the URI refers to the device ID generated by operating system, see details in *URL Format* .

8.5.3 /ISAPI/System/IO/outputs/<ID>/trigger?format=json&devIndex=<uuid>

Trigger alarm output manually.

Request URI Definition

Table 8-27 PUT /ISAPI/System/IO/outputs/<ID>/trigger?format=json&devIndex=<uuid>

Method	PUT
Description	Trigger alarm output manually.
Query	format: determine the format of request or response message. devIndex: a uuid or guid (a 32-byte or 128-bit random number) for locating the connected or lower-level devices, which is unique and generated by operating system when adding device, and its format is "xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx".
Request	<i>JSON_IOPortData</i>
Response	<i>JSON_ResponseStatus</i>

Remarks

- The alarm output is toggled to a high or low signal.
- The **ID** in the URI is the alarm output No.

8.5.4 /ISAPI/System/IO?format=json&devIndex=<uuid>

Get alarm input and output information of device.

Request URI Definition

Table 8-28 GET /ISAPI/System/IO?format=json&devIndex=<uuid>

Method	GET
Description	Get alarm input and output information of device.
Query	format: determine the format of request or response message. devIndex: a uuid or guid (a 32-byte or 128-bit random number) for locating the connected or lower-level devices, which is unique and

	generated by operating system when adding device, and its format is "xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx".
Request	None
Response	Succeeded: <i>JSON_IOPortList</i> Failed: <i>JSON_ResponseStatus</i>

8.5.5 /ISAPI/System/Network/interfaces/<ID>/ipAddress? format=json&devIndex=<uuid>

Operations about IP address of a specific network interface.

Request URL Definition

**Table 8-29 GET /ISAPI/System/Network/interfaces/<ID>/ipAddress?
format=json&devIndex=<uuid>**

Method	GET
Description	Get the IP address of a specific network interface.
Query	format: determine the format of request or response message. devIndex: a uuid or guid (a 32-byte or 128-bit random number) for locating the connected or lower-level devices, which is unique and generated by operating system when adding device, and its format is "xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx".
Request	None.
Response	<i>JSON_IPAddress</i>

**Table 8-30 PUT /ISAPI/System/Network/interfaces/<ID>/ipAddress?
format=json&devIndex=<uuid>**

Method	PUT
Description	Set the IP address of a specific network interface.
Query	format: determine the format of request or response message. devIndex: a uuid or guid (a 32-byte or 128-bit random number) for locating the connected or lower-level devices, which is unique and generated by operating system when adding device, and its format is "xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx".
Request	<i>JSON_IPAddress</i>
Response	<i>JSON_ResponseStatus</i>

Remarks

During the above operations, the following error codes may be returned: 0x60000004, 0x60000005, 0x60000006, 0x60000007, and 0x60000010, see the corresponding error descriptions in **Error Codes** .

8.5.6 /ISAPI/System/Network/interfaces/<ID>?format=json&devIndex=<uuid>

Operations about a specific network interface information.

Request URL Definition

Table 8-31 GET /ISAPI/System/Network/interfaces/<ID>?format=json&devIndex=<uuid>

Method	GET
Description	Get the information of a specific network interface.
Query	format: determine the format of request or response message. devIndex: a uuid or guid (a 32-byte or 128-bit random number) for locating the connected or lower-level devices, which is unique and generated by operating system when adding device, and its format is "xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx".
Request	None.
Response	<i>JSON_NetworkInterface</i>

Table 8-32 PUT /ISAPI/System/Network/interfaces/<ID>?format=json&devIndex=<uuid>

Method	PUT
Description	Set the information of a specific network interface.
Query	format: determine the format of request or response message. devIndex: a uuid or guid (a 32-byte or 128-bit random number) for locating the connected or lower-level devices, which is unique and generated by operating system when adding device, and its format is "xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx".
Request	<i>JSON_NetworkInterface</i>
Response	<i>JSON_ResponseStatus</i>

Remarks

If the above operations are failed, the following error code may be returned: 0x60000004, 0x60000005, 0x60000006, 0x60000007, and 0x60000010, see the corresponding error descriptions in **Error Codes** .

8.5.7 /ISAPI/System/Network/interfaces?format=json&devIndex=<uuid>

Get all network interfaces' information of the device.

Request URL Definition

Table 8-33 GET /ISAPI/System/Network/interfaces?format=json&devIndex=<uuid>

Method	GET
Description	Get all network interfaces' information of the device.
Query	format: determine the format of request or response message. devIndex: a uuid or guid (a 32-byte or 128-bit random number) for locating the connected or lower-level devices, which is unique and generated by operating system when adding device, and its format is "xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx".
Request	None.
Response	<i>JSON_NetworkInterfaceList</i>

8.5.8 /ISAPI/System/reboot?format=json

Reboot Device Gateway.

Request URL Definition

Table 8-34 PUT /ISAPI/System/reboot?format=json

Method	PUT
Description	Reboot Device Gateway.
Query	format: determine the format of request or response message.
Request	None.
Response	<i>JSON_ResponseStatus</i>

8.5.9 /ISAPI/System/reboot?format=json&devIndex=<uuid>

Reboot the device.

Request URL Definition

Table 8-35 PUT /ISAPI/System/reboot?format=json&devIndex=<uuid>

Method	PUT
Description	Reboot the device.
Query	format: determine the format of request or response message. devIndex: a uuid or guid (a 32-byte or 128-bit random number) for locating the connected or lower-level devices, which is unique and generated by operating system when adding device, and its format is "xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx".
Request	None.
Response	<i>JSON_ResponseStatus</i>

8.5.10 /ISAPI/System/time/timeZone?devIndex=<uuid>

Get or set time zone parameters of devices that have been added to Device Gateway.

Request URI Definition

Table 8-36 GET /ISAPI/System/time/timeZone?devIndex=<uuid>

Method	GET
Description	Get time zone parameters of devices that have been added to Device Gateway.
Query	devIndex: a uuid or guid (a 32-byte or 128-bit random number) for locating the connected or lower-level devices, which is unique and generated by operating system when adding device, and its format is "xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx".
Request	None.
Response	Succeeded: Time string (e.g., GST-8:00:00 and CST-8:00:00DST00:30:00,M4.1.0/02:00:00,M10.5.0/02:00:00, the later one is with daylight saving time) Failed: <i>JSON_ResponseStatus</i>

Table 8-37 PUT /ISAPI/System/time/timeZone?devIndex=<uuid>

Method	PUT
Description	Set time zone parameters of devices that have been added to Device Gateway.

Query	devIndex: a uuid or guid (a 32-byte or 128-bit random number) for locating the connected or lower-level devices, which is unique and generated by operating system when adding device, and its format is "xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx".
Request	Time string (e.g., GST-8:00:00 and CST-8:00:00DST00:30:00,M4.1.0/02:00:00,M10.5.0/02:00:00, the later one is with daylight saving time)
Response	<i>JSON_ResponseStatus</i>

Remarks

In the time string example with daylight saving time, the "DST00:30:00" refers to the offset of daylight saving time, the "M4.1.0/02:00:00" refers to the start daylight saving time (i.e., 02:00:00 on Sunday of the first week in April), and the "M10.5.0/02:00:00" is the end daylight saving time (i.e., 02:00:00 on Sunday of the fifth week in October).

8.5.11 /ISAPI/System/time?format=json

Get or set time synchronization parameters of Device Gateway.

Request URL Definition

Table 8-38 GET /ISAPI/System/time?format=json

Method	GET
Description	Get time synchronization parameters of Device Gateway.
Query	format: determine the format of request or response message.
Request	None.
Response	Succeeded: <i>JSON_Time</i> Failed: <i>JSON_ResponseStatus</i>

Table 8-39 PUT /ISAPI/System/time?format=json

Method	PUT
Description	Set time synchronization parameters of Device Gateway.
Query	format: determine the format of request or response message.
Request	<i>JSON_Time</i>
Response	<i>JSON_ResponseStatus</i>

8.5.12 /ISAPI/System/time?format=json&devIndex=<uuid>

Get or set time synchronization parameters of devices that have been added to Device Gateway.

Request URL Definition

Table 8-40 GET /ISAPI/System/time?format=json&devIndex=<uuid>

Method	GET
Description	Get time synchronization parameters of devices that have been added to Device Gateway.
Query	format: determine the format of request or response message. devIndex: a uuid or guid (a 32-byte or 128-bit random number) for locating the connected or lower-level devices, which is unique and generated by operating system when adding device, and its format is "xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx".
Request	None.
Response	Succeeded: <i>JSON_Time</i> Failed: <i>JSON_ResponseStatus</i>

Table 8-41 PUT /ISAPI/System/time?format=json&devIndex=<uuid>

Method	PUT
Description	Set time synchronization parameters of devices that have been added to Device Gateway.
Query	format: determine the format of request or response message. devIndex: a uuid or guid (a 32-byte or 128-bit random number) for locating the connected or lower-level devices, which is unique and generated by operating system when adding device, and its format is "xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx".
Request	<i>JSON_Time</i>
Response	<i>JSON_ResponseStatus</i>

8.6 Others

8.6.1 http://<ipAddress>[:port][videoUrl]

Download video files.

Request URL Definition

Table 8-42 GET http://<ipAddress>[:port][videoUrl]

Method	GET
Description	Download video files.
Query	None.
Request	None.
Response	Video data

Remarks

The **ipAddress** and **port** in the request URL refer to the IP address and port number for logging in to HikGateway; the **videoUrl** in the URL is the video file URL (**videoURI**) uploaded in the alarm message.

Chapter 9 Request and Response Messages

9.1 JSON_ChannelInfo

ChannelInfo message in JSON format

```
{
  "ChannelInfo":{
    "VideoChannel":[
/*video channel information*/
    {
      "Channel":{
        "id": ,
/*channel No., integer*/
        "name": "",
/*channel name, string*/
        "type": "",
/*string, "analog, digital, zero"*/
        "status": ""
/*string, "online, offline, disable"*/
      }
    },
    "AlarmInChannel":[
/*alarm input information*/
    {
      "Channel":{
        "id": ,
/*channel No., integer*/
        "name": "",
/*channel name, string*/
        "type": "",
/*string, "analog, digital, zero"*/
      }
    },
    "AlarmOutChannel":[
/*alarm output channel information*/
    {
      "Channel":{
        "id": ,
/*channel No., integer*/
        "name": "",
/*channel name, string*/
        "type": "",
/*string, "analog, digital, zero"*/
      }
    }
  ]
}
```



```
    "AudioChannel":[
/*audio channel information*/
    {
        "Channel":{
            "id": ,
/*channel No., integer*/
            "name": "",
/*channel name, string*/
            "type": "",
/*string, "analog, digital, zero"*/
        }
    }
]
```

9.2 JSON_DelDevList

DelDevList message in JSON format

```
{
  "DelDevList":[
    {
      "Dev":{
        "devIndex": "",
/*required, device ID (uuid),string*/
        "status": "",
/*required, deleting result, "success, fail"*/
        "subStatusCode": ""
/*optional, string, sub status code, it is returned when adding failed:
"badParameters"-incorrect parameters, "theDeviceIdDoesNotExist"-the device ID
does not exist, "noMemory"-insufficient device memory*/
      }
    }
  ]
}
```

9.3 JSON_DeviceInfo

DeviceInfo message in JSON format

```
{
  "DeviceInfo":{
    "deviceName": "",
/*required, device name*/
    "deviceId": "",
/*required, device ID (uuid/guid), read only*/
    "deviceDescription": "",
/*optional, device description*/
  }
```

```
"systemContact": "",
/*system, read only, string*/
"model": "",
/*required, device model, read only*/
"serialNumber": "",
/*required, serial No., read only*/
"macAddress": "",
/*required, MAC address, read only*/
"firmwareVersion": "",
/*required, firmware version, read only*/
"firmwareReleasedDate": "",
/*optional, firmware release data, read only*/
"bootVersion": "",
/*required, drive version, read only*/
"bootReleasedDate": "",
/*optional, drive release data, read only*/
"hardwareVersion": "",
/*optional, hardware version, read only*/
"encoderVersion": "",
/*optional, encoder version, read only*/
"encoderReleasedDate": "",
/*optional, encoder release data, read only*/
"decoderVersion": "",
/*optional, decoder version, read only*/
"decoderReleasedDate": "",
/*optional, decoder release data, read only*/
"softwareVersion": "",
/*optional, software version, read only*/
"deviceType": "",
/*required, device type, "IPCamera, IPDome, DVR, HybirdNVR, NVR, DVS, IPZoom,
CVR, Gateway, ACS"*/
"telecontrolID": ,
/*optional, digital device type, read only*/
"operationSystem": ""
/*optional, operating system information, string, read only*/
"devType":
/*optional, integer, read only, device type values*/
}
}
```

Remarks

The read-only nodes are invalid when calling URL by PUT method.

9.4 JSON_DeviceInfoList

JSON Message about Device Information List

```
{
  "DeviceInfoList": {
    "devIndex": "",
```

```
/*optional, string type, device UUID, it cannot be edited and it is returned
when adding device, e.g., "2cd6716d-767f-4756-ac55-50276a5e3b4a"*/
    "deviceName": "",
/*required, device name*/
    "deviceId": "",
/*required, device ID (uuid/guid), read only*/
    "deviceDescription": "",
/*optional, device description*/
    "systemContact": "",
/*system, read only, string*/
    "model": "",
/*required, device model, read only*/
    "serialNumber": "",
/*required, serial No., read only*/
    "macAddress": "",
/*required, MAC address, read only*/
    "firmwareVersion": "",
/*required, firmware version, read only*/
    "firmwareReleasedDate": "",
/*optional, firmware release data, read only*/
    "bootVersion": "",
/*required, drive version, read only*/
    "bootReleasedDate": "",
/*optional, drive release data, read only*/
    "hardwareVersion": "",
/*optional, hardware version, read only*/
    "encoderVersion": "",
/*optional, encoder version, read only*/
    "encoderReleasedDate": "",
/*optional, encoder release data, read only*/
    "decoderVersion": "",
/*optional, decoder version, read only*/
    "decoderReleasedDate": "",
/*optional, decoder release data, read only*/
    "softwareVersion": "",
/*optional, software version, read only*/
    "deviceType": "",
/*required, device type, "IPCamera, IPDome, DVR, HybirdNVR, NVR, DVS, IPZoom,
CVR, Gateway, SecurityCP, ACS"*/
    "telecontrolID": ,
/*optional, digital device type, read only*/
    "operationSystem": ""
/*optional, operating system information, string, read only*/
    "devType":
/*optional, integer, read only, device type values*/
    }
}
```

9.5 JSON_DeviceInList

JSON Message about Input Parameter List of Adding Devices

```
{
  "DeviceInList": [{
    "Device":{
      "protocolType": "",
      /*required, protocol type, string, "ehome"-ISUP (Intelligent Security Uplink
      Protocol), "ehomeV5"-ISUP 5.0*/
      "EhomeParams":{
        /*it is valid only when protocolType are "ehome" or "ehomeV5"*/
        "EhomeID": "",
        /*required, ISUP (Intelligent Security Uplink Protocol) ID, string, up to 31
        characters are allowed*/
        "EhomeKey": ""
        /*string type, it is valid only when protocolType is "ehomeV5", and the maximum
        key length is 32 bytes*/
        "heartInterval": ,
        /*optional, int, heartbeat interval, unit: second, the default value is 10*/
        "heartExceededCount":
        /*optional, int, timeout attempts of heartbeat, the default value is 3*/
      },
      "devName": "",
      /*optional, device name, string, up to 32 characters are allowed*/
      "devType": "",
      /*optional, string, device type, "encodingDev"-encoding device, "SecurityCP"-
      security control panel, "AccessControl"-access control device*/
      "visualVerification":
      /*optional, boolean, whether to enable visual verification, the default status
      is enabled; if enabled, the uploaded alarm video file will be verified by
      Device Gateway when alarm is triggered; this node is only available for the
      alarm device that supports ISUP 5.0*/
    }
  },
  {...}
]
```

9.6 JSON_DeviceOutList

DeviceOutList message in JSON format

```
{
  "DeviceOutList": [
    {
      "Device":{
        "devIndex": "",
        /*optional, device ID (uuid/guid), string type, it will be returned when the
```

```
device is added*/
    "devName": "",
    /*optional, device name, string type, up to 31 characters are allowed*/
    "protocolType": "",
    /*required, protocol type, string, "ehome"-ISUP (Intelligent Security Uplink
    Protocol), "ehomeV5"-ISUP 5.0*/
    "EhomeParams":{
    /*it is valid only when protocolType are "ehome" or "ehomeV5"*/
        "EhomeID":""
    /*required, ISUP ID, string type, up to 32 characters are allowed*/
    },
    "status": "",
    /*required, adding result: "success"-added, "fail"-adding failed*/
    "subStatusCode":""
    /*optional, string, sub status code, it is returned when adding failed:
    "badParameters"-incorrect parameter; "monitorNodeOverLimit"-no more device can
    be added; "noMemory"-insufficient device memory; "deviceExist"-the device
    already exist*/
    }
    },
    {...}
]
}
```

9.7 JSON_DevIndexList

DevIndexList message in JSON format

```
{
    "DevIndexList": [ "", "" ]
    /*required, device ID (uuid/uuid) array, string*/
}
```

9.8 JSON_DevList

DevList message in JSON format

```
{
    "DevList": [ "", ..., "" ]
    /*required, string type, list of device UUID*/
}
```

9.9 JSON_ErrorList

ErrorList message in JSON format

```
{
  "requestURL": "",
  /*optional, request URL*/
  "ErrorList":[{"
    "id": "",
    /*required, string type, error object ID*/
    "statusCode": ,
    /*status code, it will not be returned when succeeded or no exception*/
    "statusString": "",
    /*status description, it will not be returned when succeeded or no exception*/
    "subStatusCode": "",
    /*sub status code, it will not be returned when succeeded or no exception*/
    "errorCode": ,
    /*error code, it is optional when succeeded, and it corresponds to
subStatusCode*/
    "errorMsg": "ok"
  /*optional, error information, which corresponds to subStatusCode*/
  }]
}
```

9.10 JSON_EventNotificationAlert_AlarmEventInfo

EventNotificationAlert message with alarm/event information in JSON format

```
{
  "EventNotificationAlert": {
    "channelID": "",
    /*optional, dep, string, device channel No.*/
    "dateTime": "",
    /*required, alarm/event triggered or occurred time, it must contain time zone
    information, e.g., "2017-04-22T15:39:01+08:00"*/
    "activePostCount": ,
    /*required, integer, alarm/event frequency*/
    "eventType": "",
    /*required, string, alarm/event types, see details in the Event Types*/
    "eventState": "",
    /*required, string, durative alarm/event status: "active"-valid, "inactive"-
    invalid, e.g., when a moving target is detected, the alarm/event information
    will be uploaded continuously unit the status is set to "inactive"*/
    "eventDescription": "",
    /*required, string, description*/
    "devIndex": "",
    /*optional, string, device ID (uuid/guid)*/
    "...": "",
    /*for different alarm/event types, the nodes are different, see the message
    examples below for details*/
    "channelName": ""
  /*required, string, camera name*/
  }
}
```

See Also

Event Types

9.11 JSON_EventNotificationAlert_CidAlarmMsg

JSON message about CID (Contact ID) alarm details

```
{
  "EventNotificationAlert": {
    "channelID": "",
    "dateTime": "",
    "activePostCount": ,
    "eventType": "",
    /*required, string type, here it should be set to "CIDAlarm"*/
    "eventState": "",
    "eventDescription": "",
    "devIndex": "",
    "channelName": "",
    /*for the descriptions of above nodes, refer to
    JSON_EventNotificationAlert_AlarmEventInfo*/
    "deviceID": "",
    /*optional, string type, device ID*/
    "CIDAlarm": {
    /*optional, CID alarm information*/
      "subSys": ,
      /*optional, integer type, No. of partition that triggers alarm, which is
      between 0 and 32; 0-zone alarm*/
      "zoneNo": ,
      /*optional, integer type, zone No.*/
      "CIDCode": "",
      /*optional, string type, CID alarm code with 4-bit, the first three bits is the
      event/alarm code, and the last bit is the alarm status (1-triggered, 3-
      restored), e.g., "1103"-zone alarm triggered, "3103"-zone alarm restored, refer
      to CID Code for details*/
      "CIDType": ,
      /*optional, integer type, CID alarm type: 1 (zone alarm), 2 (video alarm), 3
      (virtual zone alarm), 4 (duress alarm), 5 (exception alarm), 6 (operation
      alarm)*/
      "CIDDescribe": "",
      /*optional, string type, description of CID code, the maximum description
      length is 127 bytes*/
      "timeZoneIdx": 0,
      /*optional, integer type, time zone index No.: 1 (GMT-12:00), 2 (GMT-11:00), 3
      (GMT-10:00), ..., 14 (GMT-01:00), 15 (GMT), 16 (GMT+01:00), ..., 34 (GMT-13:00), 35
      (GMT+14:00)*/
      "triggerTime": "",
      /*optional, string type, CID alarm time, e.g., 2009-02-24 16:59:00*/
      "uploadTime": "",
      /*optional, string type, CID alarm uploaded time, e.g., 2009-02-24 16:59:00*/
    }
  }
}
```

```
    "CIDParam": "",
    /*optional, string type, CID parameter, which contains 8 sub parameters, i.e.,
    userType, userNo, zoneNo, keyboardNo, videoChanNo, dskNo, moduleAddr, and
    userName; each sub parameter should be separated by comma*/
    "UUID": ""
    /*optional, string type, alarm ID*/
  }
}
```

Remarks

The descriptions of 8 sub parameters of CID parameter (**CIDParam**) are shown as below:

userType

User type, a 4-byte integer, values: 1-keyboard user, 2-network user, others-invalid

userNo

User No., a 4-byte integer, the value -1 is invalid.

zoneNo

Zone No., a 4-byte integer, the value -1 is invalid.

keyboardNo

Keyboard No., a 4-byte integer, the value -1 is invalid.

videoChanNo

Video channel No., a 4-byte integer, the value -1 is invalid.

dskNo

HDD No., a 4-byte integer, the value -1 is invalid.

moduleAddr

Module address, a 4-byte integer, the value -1 is invalid.

serName

User name, the maximum name length is 31 bytes, it can be set to "NONE".

See Also

JSON_EventNotificationAlert_AlarmEventInfo

Hikvision CID Code

Example

Sample Message of CID Alarm

```
{
  "EventNotificationAlert": {
    "channelID": "1",
    "dateTime": "2018-01-21T12:50:39+08:00",
    "activePostCount": 1,
    "eventType": "videoReview",
    "eventState": "active",
```



```
"eventDescription": "video review",
"devIndex": "2cd6716d-767f-4756-ac55-50276a5e3b4a",
"channelName": "Ipdome",
"deviceID": "123456",
"CIDAlarm": {
  "subSys": 0,
  "zoneNo": 1,
  "CIDCode": "1103",
  "CIDType": 1,
  "CIDDescribe": "alarm",
  "timeZoneIdx": 0,
  "triggerTime": "2009-02-24 16:59:00",
  "uploadTime": "2009-02-24 16:59:00",
  "CIDParam": "1,2,1,1,1,1,0,admin",
  "UUID": "0"
}
}
```

9.12 JSON_EventNotificationAlert_DevStatusChangedAlarmMsg

JSON message about details of device status changed alarm

```
{
  "EventNotificationAlert": {
    "channelID": "",
    "dateTime": "",
    "activePostCount": ,
    "eventType": "",
    /*required, string, here it should be set to "devStatusChanged"*/
    "eventState": "",
    "eventDescription": "",
    "devIndex": "",
    "channelName": "",
    /*for the descriptions of above nodes, refer to
    JSON_EventNotificationAlert_AlarmEventInfo*/
    "status": ""
    /*required, string, device status: "online" and "offline"*/
  }
}
```

See Also

JSON_EventNotificationAlert_AlarmEventInfo

Example

Sample Code of Device Status Changed Alarm Details

```
{
  "EventNotificationAlert": {
    "channelID": "1",
```

```
"dateTime": "2018-01-21T12:50:39+08:00",
"activePostCount": 1,
"eventType": "devStatusChanged",
"eventState": "active",
"eventDescription": "device status about online or offline changed",
"devIndex": "2cd6716d-767f-4756-ac55-50276a5e3b4a",
"channelName": "Ipdome",
"status": "online"
}
}
```

9.13 JSON_EventNotificationAlert_HeartbeatInfo

EventNotificationAlert message with heartbeat information in JSON format

```
{
  "EventNotificationAlert": {
    "channelID": "",
    /*optional, dep, string, device channel No.*/
    "dateTime": "",
    /*required, heartbeat uploaded time, e.g., "2017-04-22T15:39:01+08:00"*/
    "activePostCount": ,
    /*required, integer, heartbeat frequency*/
    "eventType": "",
    /*required, string, for heartbeat, it is "heartBeat"*/
    "eventState": "",
    /*required, string, heartbeat triggering status: "active"-triggered, "inactive"-
    not triggered*/
    "eventDescription": ""
    /*required, string, description*/
  }
}
```

Remarks

You can calculate the arming duration according to the value of node **activePostCount** and the heartbeat interval.

Example

Message Example of Uploading Heartbeat Information

```
{
  "EventNotificationAlert": {
    "channelID": "1",
    "dateTime": "2017-04-22T15:39:01+08:00",
    "activePostCount": 2,
    "eventType": "heartBeat ",
    "eventState": "active",
    "eventDescription": "Heartbeat"
  }
}
```

```
}  
}
```

9.14 JSON_EventNotificationAlert_VideoVerificationAlarmMsg

JSON message about video verification alarm details

```
{  
  "EventNotificationAlert": {  
    "VideoReview": {  
/*required, video verification alarm information*/  
      "HikCIDParams": {  
        "CIDCode": "",  
/*optional, string, CID alarm code with 4-bit, the first three bits is the  
event/alarm code, and the last bit is the alarm status (1-triggered, 3-  
restored), e.g., "1103"-zone alarm triggered, "3103"-zone alarm restored, refer  
to "Hikvision CID Code" for details*/  
        "CIDDescribe": "",  
/*optional, string, description of CID code, the maximum description length is  
127 bytes*/  
        "CIDParam": "",  
/*optional, string, CID parameter, which contains 8 sub parameters, i.e.,  
userType, userNo, zoneNo, keyboardNo, videoChanNo, dskNo, moduleAddr, and  
userName; each sub parameter should be separated by comma*/  
        "CIDType": ,  
/*optional, int, CID alarm type: 1 (zone alarm), 2 (video alarm), 3 (virtual  
zone alarm), 4 (duress alarm), 5 (exception alarm), 6 (operation alarm)*/  
        "UUID": "",  
/*optional, string, alarm ID*/  
        "ZoneNo": ,  
/*optional, int, zone No.*/  
        "subSys": ,  
/*optional, int, No. of partition that triggers alarm, which is between 0 and  
32; 0-zone alarm*/  
        "timeZoneIdx": ,  
/*optional, int, time zone index No.: 1 (GMT-12:00), 2 (GMT-11:00), 3  
(GMT-10:00), ..., 14 (GMT-01:00), 15 (GMT), 16 (GMT+01:00), ..., 34 (GMT-13:00), 35  
(GMT+14:00)*/  
        "triggerTime": "",  
/*optional, string, CID alarm time, e.g., 2009-10-09 16:59:00*/  
        "uploadTime": "",  
/*optional, string, CID alarm uploaded time, e.g., 2019-10-09 17:55:46*/  
        "videoURL": ""  
/*optional, string, video file URL, you can download video file via the URL by  
the request URL: http://<ipAddress>[:port][videoURI], e.g., "/HikGatewayStorage/  
pic?F14EEF8D4F9A900DF3048E0F920BB7E4Q00162958_CH02_20191009175543.mp4"*/  
      },  
      "protocolType": ""  
/*required, string type, protocol type: "HikCIDProtocol"*/  
    },  
  },  
}
```

```
"activePostCount": ,
/*required, int, alarm/event frequency*/
"channelID": "",
/*optional, string, device channel ID*/
"channelName": "",
/*required, string, camera name*/
"dateTime": "",
/*required, alarm/event triggered or occurred time, it must contain time zone
information, e.g., "2019-10-10T10:55:43+08:00"*/
"devIndex": "",
/*optional, string, device No.*/
"deviceId": "",
/*optional, string, device ID*/
"eventDescription": "",
/*required, string, event description*/
"eventState": "",
/*required, string, alarm/event status: "active"-valid, "inactive"-invalid,
e.g., when a moving target is detected, the alarm/event information will be
uploaded continuously until the status is set to "inactive"*/
"eventType": ""
/*required, string, alarm/event types, here it should be set to "videoReview"*/
}
}
```

Remarks

The descriptions of 8 sub parameters of CID parameter (**CIDParam**) are shown as below:

userType

User type, a 4-byte integer, values: 1-keyboard user, 2-network user, others-invalid

userNo

User No., a 4-byte integer, the value -1 is invalid.

zoneNo

Zone No., a 4-byte integer, the value -1 is invalid.

keyboardNo

Keyboard No., a 4-byte integer, the value -1 is invalid.

videoChanNo

Video channel No., a 4-byte integer, the value -1 is invalid.

dskNo

HDD No., a 4-byte integer, the value -1 is invalid.

moduleAddr

Module address, a 4-byte integer, the value -1 is invalid.

serName

User name, the maximum name length is 31 bytes, it can be set to "NONE".

See Also

Hikvision CID Code

9.15 JSON_ExDevStatus

ExDevStatus message in JSON format

```
{
  "ExDevStatus":{
    "OutputModList":[{
/*optional, wireless output module list/
      "OutputMod":{
        "id": ,
/*required, integer type, wireless output module No.*/
        "seq": "",
/*required, string type, serial No. of wireless output module*/
        "status": "",
/*optional, string type, wireless output module status: "notRelated"-unlinked,
"online", "offline"*/
        "tamperEvident": ,
/*optional, boolean type, whether to enable zone tampering: true=yes, false=no*/
        "charge": ""
/*optional, string type, power status: "normal", "lowPower"*/
      }
    ]},
    "OutputList":[{
/*optional, relay list*/
      "Output":{
        "id": ,
/*required, integer type, relay No.*/
        "name": "",
/*optional, string type, relay name*/
        "status": "",
/*optional, string type, relay status: "notRelated"-unlinked, "on"-open, "off"-
closed, "offline"*/
        "tamperEvident": ,
/*optional, boolean type, whether to enable zone tampering: true=yes, false=no*/
        "charge": "",
/*optional, string type, power status: "normal", "lowPower"*/
        "linkage": ""
/*optional, string type, event linkage type of relay: "alarm", "arming",
"disarming", "manualCtrl"-manual control*/
      }
    ]},
    "SirenList":[{
/*optional, siren list*/
      "Siren":{
        "id": ,
/*required, integer type, siren No.*/
```

```
        "seq": "",
/*required, string type, siren serial No.*/
        "name": "",
/*optional, string type, siren name*/
        "status": "",
/*optional, string type, siren status: "notRelated"-unlinked, "on"-enabled,
"off"-disabled, "offline"*/
        "tamperEvident": ,
/*optional, boolean type, whether to enable zone tampering: true=yes, false=no*/
        "charge": ""
/*optional, string type, power status: "normal", "lowPower"*/
    }
}],
    "RepeaterList": [{
/*optional, repeater list*/
        "Repeater": {
            "id": ,
/*required, integer type, repeater No.*/
            "seq": "",
/*required, string type, repeater serial No.*/
            "name": "",
/*optional, string type, repeater name*/
            "status": "",
/*optional, string type, repeater status: "notRelated"-unlinked, "online",
"offline"*/
            "tamperEvident": ,
/*optional, boolean type, whether to enable zone tampering: true=yes, false=no*/
            "charge": ""
/*optional, string type, power status: "normal", "lowPower"*/
        }
    ]
}
}
```

9.16 JSON_IOPortData

IOPortData message in JSON format

```
{
    "IOPortData": {
        "outputState": ""
/*required, string, output status, opt="high,low" */
    }
}
```

9.17 JSON_IOPortList

JSON message about alarm input and output list

```
{
  "IOPortList":{
    "IOInputPortList":[{
      "IOInputPort":{
        "id": ,
/*required, int, alarm input No.*/
        "enable": ,
/*required, boolean, whether to enable alarm input*/
        "triggering": "",
/*optional, string, triggering level: "high", "low"*/
        "name": ""
/*optional, string, alarm input name*/
      }
    }],
    "IOOutputPortList":[{
      "IOOutputPort":{
        "id": ,
/*required, int, alarm output No.*/
        "enable": ,
/*required, boolean, whether to enable alarm output*/
        "name": ""
/*optional, string, alarm output name*/
      }
    }]
  }
}
```

9.18 JSON_IPAddress

IPAddress message in JSON format

```
{
  "IPAddress":{
    "ipVersion": "",
/*required, IP address version, string, "v4, v6, dual"*/
    "addressingType": "",
/*required, string, IP address type, "static, dynamic, apipa"*/
    "ipAddress": "",
/*required, string, IPv4 address, e.g., 255.255.255.0*/
    "subnetMask": "",
/*required, subnet mask of IPv4, e.g., 255.255.255.0*/
    "ipv6Address": "",
/*required, IPv6 address, e.g., 2001:DB8:2de::1*/
    "bitMask": ,
/*required, string, prefix size of IPv6 subnet mask, e.g., 48*/
    "DefaultGateway":{
/*default gateway*/
      "ipAddress": "",
      "ipv6Address": ""
    },
  },
}
```

```
"PrimaryDNS":{
/*major DNS, which depends on IP address type*/
  "ipAddress":"","
  "ipv6Address":""
},
"SecondaryDNS":{
/*minor DNS, which depends on IP address type*/
  "ipAddress":"","
  "ipv6Address":""
},
"Ipv6Mode":{
/*optional, IPv6 mode*/
  "ipV6AddressingType": "",
/*optional, IPv6 address type, which depends on ipVersion, "ra, manual, dhcp",
string, ra-advertised by router, manual: configured manually, dhcp: allocated
by server*/
  "ipv6AddressList":[
/*optional, IPv6 address configuration, multiple addresses are available*/
    {
      "v6Address":{
        "id": "",
/*optional, ID of IPv6 address, string*/
        "type": "",
/*optional, IPv6 address type, "ra, manual, dhcp"*/
        "address": "",
/*optional, IPv6 address, string*/
        "bitMask": 60
/*optional, number of IPv6 mask bit, integer*/
      }
    }
  ],
},
}
```

Note

- If **addressingType**="dynamic", the nodes below this node is not required, and the DHCP service should be enabled; if **addressingType**="static", the IP address should be configured manually, and the Device Gateway and DNS are optional; if **addressingType**="APIPA", the IP address will be automatically configured, and the Device Gateway and DNS are optional, but the DHCP service is not required.
 - The validity of nodes **ipAddress** and **ipv6Address** are determined by **ipVersion**. If **ipVersion**="v4", only the node **ipAddress** is valid; if **ipVersion**="v6", only the node **ipv6Address** is valid; if **ipVersion**="dual", both the nodes **ipAddress** and **ipv6Address** are valid.
 - For Device Gateway, the node **ipv6Mode** is not available.
-

9.19 JSON_List

List message in JSON format

```
{
  "List": [{
    "id":
/*integer type, zone No., it is required when control multiple zones*/
  }]
}
```

9.20 JSON_NetworkInterface

NetworkInterface message in JSON format

```
{
  "NetworkInterface": {
    "id": "",
/*required, NIC ID*/
    "IPAddress": {...}
/*required, IP address, see details in
                                JSON_IPAddress
                                message*/
  }
}
```



Note

For NIC, only **id** and **IPAddress** field are required.

9.21 JSON_NetworkInterfaceList

NetworkInterfaceList message in JSON format

```
{
  "NetworkInterfaceList": [{
    "NetworkInterface": {...}
/*see details in
                                JSON_NetworkInterface
                                message*/
  }]
}
```

9.22 JSON_OutputsCtrl

OutputsCtrl message in JSON format

```
{
  "OutputsCtrl":{
    "switch":""
    /*required, string type, whether to open relay: "open"-yes, "close"-no*/
    "List":[{
      /*this node is required when control multiple relays*/
      "id":
      /*required, integer type, relay No., which starts from 1*/
    }]
  }
}
```

9.23 JSON_ResponseStatus

ResponseStatus message in JSON format

```
{
  "requestURL": "",
  /*optional, request URL,string*/
  "statusCode": ,
  /*required, status code, int*/
  "statusString": "",
  /*required, status description, string*/
  "subStatusCode": "",
  /*required, sub status code, string*/
  "errorCode": ,
  /*optional, this field is required when statusCode is not 1. Error code, which
  correspond to subStatusCode,int*/
  "errorMsg": ""
  /*optional, his field is required when statusCode is not 1. Error details,
  string*/
  "rebootRequired":
  /*optional, whether the reboot is required: 1=yes, reboot to take effect; this
  node is not returned or other values-reboot is not required*/
}
```



Note

See **Error Codes** for details about the status codes, sub status codes, error codes, and error descriptions.

9.24 JSON_ResponseStatus_AuthenticationFailed

ResponseStatus message in JSON format for failed authentication

```
{
  "requestURL": "",
  /*optional,URL,string, e.g., "/ISAPI/Streaming/channels/1"*/
  "statusCode": ,
  /*required, status code, int*/
  "statusString": "",
  /*required, status description,string*/
  "subStatusCode": "",
  /*required, sub status code, string*/
  "errorCode": ,
  /*error code, when statusCode is not 1, it is required, and which corresponds
  to subStatusCode*/
  "errorMsg": "",
  /*error description, when statusCode is not 1, it is required*/
  "lockStatus": "",
  /*locking status: "unlock", "lock", string, optional*/
  "retryTimes": ,
  /*remaining authentication attempts, integer, optional*/
  "resLockTime":
  /*remaining locking time, unit: second, integer, optional*/
}
```

Example

Message Example Returned When Authentication Failed

```
{
  "requestURL": "/ISAPI/Streaming/channels/1",
  "statusCode": 4,
  "statusString": "Invalid Operation",
  "subStatusCode": "badAuthorization",
  "errorCode": 0x40000003,
  "errorMsg": "authentication failed",
  "lockStatus": "unlock",
  "retryTimes": 5,
  "resLockTime": 30
}
```

9.25 JSON_SearchDescription

JSON Message about Search Conditions

```
{
  "SearchDescription": {
    "position": ,
    /*required, int, No., which starts from 0*/
  }
}
```

```
"maxResult": ,
/*required, int, the maximum number of searched results*/
"Filter":{
  "key": "",
/*optional, search by keyword, for example: device ID, device name (e.g.,
pointOne), or model (e.g., DS-2CD4065F)*/
  "devType": "",
/*optional, device type, string type, "encodingDev"-encoding device,
"SecurityCP"-alarm device (this value is not available for Hik Video Device
Gateway), "AccessControl"-access control device (this value is not available
for Hik Video Device Gateway)*/
  "protocolType": "",
/*optional, protocol type, string type, "ehome"-ISUP (Intelligent Security
Uplink Protocol), "ehomeV5"-ISUP 5.0*/
  "devStatus": "",
/*optional, device status, "online, offline", string*/
  "visualVerification":
/*optional, boolean, visual verification status: true (enabled), false
(disabled); this node is not available for Hik Video Device Gateway; if
enabled, the uploaded alarm video file will be verified by Device Gateway when
alarm is triggered; this node is only available for the alarm device that
supports ISUP 5.0*/
  "ISAPIUserBound":
/*optional, boolean, whether it is bound with the user who has the ISAPI
integration permission: true, false*/
}
}
```

9.26 JSON_SearchResult

JSON Message about Search Results

```
{
  "SearchResult": {
    "numOfMatches": ,
/*optional, int, number of returned records*/
    "totalMatches": ,
/*optional, int, total number of matched records*/
    "MatchList": [{
/*MatchList is returned when totalMatches is larger than 0*/
      "Device":{
        "devIndex": "",
/*required, device ID (uuid/guid), string*/
        "devName": "",
/*required, device name, string, up to 32 characters are allowed*/
        "devMode": "",
/*optional, device model, string*/
        "devType": "",
/*required, device type, string, "encodingDev"-encoding device, "SecurityCP"-
```

```
alarm device (this value is not available for Hik Video Device Gateway),
"AccessControl"-access control device (this value is not available for Hik
Video Device Gateway)*/
  "protocolType": "",
  /*required, protocol type, string, "ehome"-ISUP (Intelligent Security Uplink
  Protocol), "ehomeV5"-ISUP 5.0*/
  "EhomeParams":{
    /*EhomeParams is returned when protocolType is "ehome"*/
    "EhomeID": ""
    /*required, ISUP ID, string, up to 31 characters are allowed*/
    "heartInterval": ,
    /*optional, int, heartbeat interval, unit: second, the default value is 10;
    this node is not available for Hik Video Device Gateway*/
    "heartExceededCount":
    /*optional, int, timeout attempts of heartbeat, the default value is 3; this
    node is not available for Hik Video Device Gateway*/
    },
    "devStatus": "",
    /*required, device status, "online, offline", string*/
    "activeStatus": ,
    /*boolean, activation status: false (inactivated), true (activated), it is
    returned when devStatus is set to "online"*/
    "videoChannelNum": ,
    /*optional, int, number of video channels*/
    "visualVerification":
    /*optional, boolean, visual verification status: true (enabled), false
    (disabled); this node is not available for Hik Video Device Gateway; if
    enabled, the uploaded alarm video file will be verified by Device Gateway when
    alarm is triggered; this node is only available for the alarm device that
    supports ISUP 5.0*/
    "ISAPIUserBound":
    /*optional, boolean, whether it is bound with the user who has the ISAPI
    integration permission: true, false*/
    }
    },
    {...}
  ]
}
}
```

9.27 JSON_SubscribeDevEvent

SubscribeDevEvent message in JSON format

```
{
  "SubscribeDevEvent":{
    "devIndex": "",
    /*required, string, device ID*/
    "uploadMode": ""
    /*required, string, subscribe events of a specific device in arming mode, e.g.,
```

```
"all"-subscribe all events of device, "part"-subscribe some events of device*/
    "eventList":["",...]
/*this node is valid only when uploadMode is set to "part", supported event
type: "videoReview"-video verification alarm, "CIDAlarm"-CID (Contact ID)
alarm*/
    }
}
```

9.28 JSON_SubscribeDeviceMgmt

SubscribeDeviceMgmt message in JSON format

```
{
  "SubscribeDeviceMgmt":{
    "eventMode": "",
/*required, string, subscription mode: "device"-subscribe by device, "all"-
subscribe all*/
    "DevEventList":[{
/*this node is valid when eventMode is "device"*/
      "Dev":{
        "devIndex": "",
/*required, string, device ID*/
        "uploadMode": ""
/*required, string, event arming mode of device: "all"-subscribe all events of
device, "part"-subscribe some events of device*/
        "eventList":["",...]
/*this node is valid only when uploadMode is set to "part", supported event
type: "videoReview"-video verification alarm, "CIDAlarm"-CID (Contact ID)
alarm*/
      }
    ]},
    "defenceMode":"",
/*required, string type, event arming mode: "all"-subscribe all events of
device, "part"-subscribe some events of device; this node is valid only when
eventMode is set to "all"*/
    "eventList":["",...]
/*this node is valid only when defenceMode is "part", supported event type:
"videoReview"-video verification alarm, "CIDAlarm"-CID (Contact ID) alarm*/
  }
}
```

9.29 JSON_SubscribeDeviceMgmtRsp

SubscribeDeviceMgmtRsp message in JSON format

```
{
  "SubscribeDeviceMgmtRsp":{
    "id": "",
/*optional, integer, subscription ID*/
```

```
"FailList":[{
/*this node is returned when subscribing failed*/
  "Dev":{
    "devIndex": "",
/*required, string, device ID*/
    "subStatusCode": ""
/*required, error code for subscription failed, see details in
                                     Error Codes
                                     */
  }
}]
}
```

9.30 JSON_SubscribeQueryStatusList

SubscribeQueryStatusList message in JSON format

```
{
  "SubscribeQueryStatusList":[{
    "Dev":{
      "devIndex": "",
/*required, string, device ID (uuid or guid)*/
      "status": ""
/*required, arming status: "continue"-connected, "abnormalLink"-connection
exception, "fail"-subscribing failed*/
    }
  ]
}
```

9.31 JSON_SubSysList

SubSysList message in JSON format

```
{
  "SubSysList":[{
    "SubSys":{
      "id":,
/*required, integer type, partition No.*/
      "arming":"","
/*optional, string type, partition arming status: "stay"-stay armed, "away"-
away armed, "disarm"-disarmed*/
      "alarm":""
/*optional, boolean type, whether to enable partition alarm: true=yes, false-
no*/
    }
  ]
}
```

9.32 JSON_Time

JSON Message about Time Synchronization Parameters

```
{
  "Time":{
    "timeMode":"","
/*required, string, time synchronization mode: "manual"*/
    "localTime":""
/*depend, string, local time, e.g., 2019-02-28T10:50:44+08:00, this node is
valid when the timeMode is set to "manual"*/
  }
}
```

9.33 JSON_UserCheck

UserCheck message in JSON format

```
{
  "userCheck":{
    "statusValue": ,
/*required, "200,401", integer*/
    "statusString": "",
/*optional, "OK, Unauthorized", string, "OK"-authenticated, "Unauthorized"-
authentication failed*/
    "isDefaultPassword": ,
/*optional, whether it is default password, boolean, "true"-yes, "false"-no*/
    "isRiskPassword": ,
/*optional, whether it is risky password, boolean, "true"-yes, "false"-no*/
    "isActivated": ,
/*optional, whether it is activated, boolean, "true"-yes, "false"-no*/
    "lockStatus":"","
/*optional, string, lock status, "unlocked, locked"*/
    "unlockTime": ,
/*optional, integer, remaining locking time, unit: second*/
    "retryLoginTime":
/*optional, integer,remaining login attempts*/
  }
}
```

Remarks

- The user name and password are verified according to the value of **statusValue** node, if the value is 200, it indicates authenticated, otherwise, authenticating failed.
- If the value of node **retryLoginTime** is 0, the user name will be locked.

9.34 JSON_ZoneList

ZoneList message in JSON format

```
{
  "ZoneList": [{
    "Zone": {
      "id": ,
      /*required, integer type, zone No.*/
      "name": "",
      /*optional, string type, zone name*/
      "status": "",
      /*optional, string type, zone status: "notRelated"-unlinked, "online",
      "offline", "trigger"-alarm triggered, "breakDown"-fault*/
      "tamperEvident": ,
      /*optional, boolean type, whether to enable zone tampering: true=yes, false=no*/
      "shielded": ,
      /*optional, boolean type, whether to enable zone: true=yes, false=no*/
      "bypassed": "",
      /*optional, boolean type, whether to bypass zone: true=yes, false=no*/
      "armed": "",
      /*optional, boolean type, whether to arm zone: true=yes, false=no*/
      "alarm": "",
      /*optional, boolean type, whether to trigger zone alarm: true=yes, false=no*/
      "charge": "",
      /*optional, string type, power status: "normal", "lowPower"*/
    }
  }]
}
```

Appendix A. Appendixes

A.1 Event Types

General Event

eventType	Event Type Name and Details
devStatusChanged	Device Status Changed Alarm, see details in <i>JSON_EventNotificationAlert_DevStatusChangedAlarmMsg</i>

Security Control Event

eventType	Event Type Name and Details
videoReview	Video Verification Alarm, see details in <i>JSON_EventNotificationAlert_VideoVerificationAlarmMsg</i> .
CIDAlarm	CID (Contact ID) Alarm, see details in <i>JSON_EventNotificationAlert_CidAlarmMsg</i>

A.2 Error Codes

The error classification of Device Gateway SDK is based on the status codes of HTTP. 7 kinds of status codes are predefined, including 1 (OK), 2 (Device Busy), 3 (Device Error), 4 (Invalid Operation), 5 (Invalid Message Format), 6 (Invalid Message Content), and 7 (Reboot Required). Each kind of status code contains multiple sub status codes, and the error codes are in a one-to-one correspondence with the sub status codes.

StatusCode=1

SubStatusCode	Error Code	Description
ok	0x1	Operation completed.
riskPassword	0x10000002	Risky password.

StatusCode=2

SubStatusCode	Error Code	Description
noMemory	0x20000001	Insufficient memory.
upgrading	0x20000003	Upgrading.
networkError	0x20000009	Network error.

StatusCode=3

SubStatusCode	Error Code	Description
deviceError	0x30000001	Device hardware error.
createSocketError	0x30000004	Creating socket failed.
sendRequestError	0x30000006	Sending request failed.
passwordDecodeError	0x30000008	Decrypting password failed.
passwordEncryptError	0x30000009	Encrypting password failed.
pictureUploadFailed	0x3000000B	Uploading picture failed.
connecteDatabaseError	0x3000000E	Connecting to database failed.
internalError	0x30000014	Internal error.
uninitialized	0x3000000C	Uninitialized.

StatusCode=4

SubStatusCode	Error Code	Description
notSupport	0x40000001	Not supported.
lowPrivilege	0x40000002	No permission.
badAuthorization	0x40000003	Authentication failed.
methodNotAllowed	0x40000004	Invalid HTTP method.
notActivated	0x40000007	Inactivated.
hasActivated	0x40000008	Activated.
invalidContent	0x4000000A	Invalid message content.
maxSessionUserLink	0x4000000B	No more user can log in.
loginPasswordError	0x4000000C	Incorrect password.
interfaceOperationError	0x40001002	Operation failed.
openFileError	0x40001014	Opening file failed.
taskPacking	0x40001034	The resource is already occupied.
taskNoRecFile	0x40001039	The video file does not exist.
updateLangUnmatched	0x40001042	Upgrade packet language mismatches.

SubStatusCode	Error Code	Description
userMaxNum	0x40001047	No more user can be added.
monitorNodeOverLimit	0x4000104D	No more camera can be added.
deviceExist	0x40001054	The device is already added.
pwdErrorLoginFailed	0x40001055	Login failed. Check the user name and password.
setArmingError	0x40001083	Setting arming information failed.
taskModifyFailed	0x400010B1	Editing task failed.
getDeviceInfoFailed	0x400010BC	Getting device information failed.
noDiskSpace	0x400010E6	Insufficient HDD space.
cannotSameAsOldPassword	0x400010E8	The new password and old password must be different.
originalPassError	0x400010E9	Incorrect old password.
writeFileError	0x400010EA	Writing file failed.
accessFileDirectoryFailed	0x40001104	Accessing file path failed.
unknownErrorCode	0x4000111D	Unknown error code.
deviceVersionNotMatch	0x40001128	Device version mismatches.
theSessionIdDoesNotExist	0x40001135	The session ID does not exist.
theCameraIdDoesNotExist	0x40001137	The camera ID does not exist.
theDeviceIdDoesNotExist	0x4000113B	The device ID does not exist.
gettingResourceNodeInformationFailed	0x40001176	Getting resource node information failed.
noMoreTasksCanBeAdded	0x4000118A	No more task can be added.

Status Code=5

SubStatusCode	Error Code	Description
badJsonFormat	0x50000002	Invalid JSON format.
badURLFormat	0x50000003	Invalid URL format.

StatusCode=6

SubStatusCode	Error Code	Description
badParameters	0x60000001	Incorrect parameter.
badXmlContent	0x60000003	Incorrect XML message content.
badPort	0x6000000B	Port number conflicted.
portError	0x6000000C	Invalid port number.
badVersion	0x6000000F	Version mismatches.
requestMemoryNULL	0x6000003F	No memory is requested.
tokenTimeout	0x60000040	The token timed out.
passwordLenNoMoreThan16	0x6000005F	Up to 16 characters are allowed in the password.
diskError	0x60001009	HDD error.

StatusCode=7

SubStatusCode	Error Code	Description
rebootRequired	0x70000001	Reboot device to take effect.

A.3 Hikvision CID Code

CID Code	Trigger (1)/Restore (3)	Description
100	1100	Medical Alarm
	3100	Medical Alarm Restored
103	1103	Instant Alarm
	3103	Instant Alarm Restored
110	1110	Fire Alarm
	3110	Fire Alarm Restored
121	1121	Duress
122	1122	24H Silent Alarm
	3122	24H Silent Alarm Restored
123	1123	24H Audible Alarm

CID Code	Trigger (1)/Restore (3)	Description
	3123	24H Audible Alarm Restored
124	1124	24H Aux Alarm
	3124	24H Aux Alarm Restored
125	1125	24H Shock Alarm
	3125	24H Shock Alarm Restored
126	1126	Overtime Alarm
	3126	Overtime Alarm Restored
129	1129	Panic Alarm
	3129	Panic Alarm Restored
130	1130	Burglary Alarm
	3130	Burglary Alarm Restored
131	1131	Perimeter Alarm
	3131	Perimeter Alarm Restored
132	1132	Interior Alarm
	3132	Interior Alarm Restored
134	1134	Delayed Alarm
	3134	Delayed Alarm Restored
137	1137	Device Tampered
	3137	Device Tamper Restored
141	1141	BUS Open-circuit Alarm
	3141	BUS Open-circuit Restored
142	1142	BUS Short-circuit Alarm
	3142	BUS Short-circuit Restored
148	1148	Device Motion Alarm
	3148	Device Motion Alarm Restored
151	1151	Gas Leakage Alarm
	3151	Gas Leakage Alarm Restored
207	1207	Zone Pre-Alarm
	3207	Zone Pre-Alarm Restored

CID Code	Trigger (1)/Restore (3)	Description
301	1301	AC Power Down
	3301	AC Power On
302	1302	Low Battery
	3302	Normal Battery
305	1305	Control Panel Reset
333	1333	Expander Exception
	3333	Expander Restored
336	1336	Printer Disconnected
	3336	Printer Connected
338	1338	Expander Low Voltage
	3338	Normal Expander Voltage
341	1341	Expander Tampered
	3341	Expander Restored
342	1342	Expander AC Power Down
	3342	Expander AC Power On
343	1343	Wireless Repeater Tampered
	3343	Wireless Repeater Restored
344	1344	Wireless Siren Tampered
	3344	Wireless Siren Restored
345	1345	Wireless Siren Disconnected
	3345	Wireless Siren Connected
354	1354	Telephone Line Disconnected
	3354	Telephone Line Connected
382	1382	BUS Supervision Fault
	3382	BUS Supervision Restored
383	1383	Detector Tampered
	3383	Detector Tamper Restored
386	1386	Zone Open-circuit Alarm
387	1387	Zone Short-circuit Alarm

CID Code	Trigger (1)/Restore (3)	Description
401	1401	Disarming
	3401	Arming
403	1403	Auto Disarming
	3403	Auto Arming
406	1406	Alarm Clearing
	3408	Instant Arming
409	1409	Key Zone Disarming
	3409	Key Zone Arming
441	3441	Stay Arming
442	3442	Forced Arming
443	1443	Turn On Output by Schedule
	3443	Turn Off Output by Schedule
452	1452	Late Disarming
455	1455	Auto Arming Failed
460	1460	Turning On Output Failed
461	1461	Turning Off Output Failed
462	1462	Auto Disarming Failed
507	1570	Bypass
	3570	Bypass Restored
574	1574	Group Bypass
	3574	Group Bypass Restored
601	1601	Manual Report Test
602	1602	Periodic Report Test
	3602	Device Heartbeat Event
617	1617	Telephone Connection Test
627	1627	Enter Programming
628	1628	Exit Programming
810	1810	Virtual Zone Panic Alarm
811	1811	Virtual Zone Fire Alarm

CID Code	Trigger (1)/Restore (3)	Description
812	1812	Virtual Zone Burglary Alarm
847	1847	Virtual Zone Medical Alarm
862	1862	Keypad Locked
	3862	Keypad Unlocked
863	1863	Absence Alarm
910	1910	Keypad Disconnected
	3910	Keypad Connected
911	1911	KBUS Relay Disconnected
	3911	KBUS Relay Connected
912	1912	KBUS GP/K Disconnected
	3912	KBUS GP/K Connected
913	1913	KBUS MN/K Disconnected
	3913	KBUS MN/K Connected
914	1914	Wireless Detector Disconnected
	3914	Wireless Detector Connected
915	1915	Wireless Detector Low Battery
	3915	Normal Wireless Detector Battery
916	1916	Expander Disconnected
	3916	Expander Connected
917	1917	Wireless Repeater Disconnected
	3917	Wireless Repeater Connected
918	1918	Radar Transmitter Fault
	3918	Radar Transmitter Restored
919	1919	Wireless Siren Low Battery
	3919	Normal Wireless Siren Battery
920	1920	Cellular Data Network Disconnected
	3920	Cellular Data Network Connected
921	1921	SIM Card Exception
	3921	SIM Card Restored

CID Code	Trigger (1)/Restore (3)	Description
922	1922	Wi-Fi Communication Fault
	3922	Wi-Fi Connected
923	1923	RF Signal Exception
	3923	Normal RF Signal
924	1924	Network Flow Exceeded
930	1930	IP Address Conflicted
	3930	Normal IP address
931	1931	Wired Network Exception
	3931	Normal Wired Network
940	1940	Motion Detection Alarm Started
	3940	Motion Detection Alarm Stopped
941	1941	Tampering Alarm Started
	3941	Tampering Alarm Stopped
942	1942	Video Signal Loss
	3942	Video Signal Restored
943	1943	Input/Output Format Unmatched
	3943	Input/Output Format Restored
944	1944	Video Input Exception
	3944	Video Input Restored
945	1945	Full HDD
	3945	Free HDD
946	1946	HDD Exception
	3946	HDD Restored
947	1947	Upload Picture Failed
948	1948	Sending Email Failed.
949	1949	Network Camera Disconnected.
	3949	Network Camera Connected.
960	1960	Duty Checking
961	1961	Post Response

CID Code	Trigger (1)/Restore (3)	Description
970	1970	BUS Query
971	1971	BUS Registration
973	1973	Single-Zone Disarming
	3973	Single-Zone Arming
974	1974	Single-Zone Alarm Cleared
975	1975	Delete Detector
	3975	Add Detector
976	1976	Business Consulting
	3976	Business Consulting Over
977	1977	Delete Expander
	3977	Add Expander
978	1978	Delete Wireless Repeater
	3978	Add Wireless Repeater
979	1979	Delete Wireless Siren
	3979	Add Wireless Siren



See Far, Go Further