

External APIs of Uniview Access Control System V1.02 (Access Control + Visual Intercom)

Document Version: V1.02

Revision Record

Revision Date	Version	Involved Section	Description	Revised By	Remarks
2019-05-15	1.00	All chapters	Completed the draft.	Wang Xiaoyong (employee ID: 05574)	Matching firmware version: all versions later than PTS_Q2101-B0008
2019-08-05	1.01	All chapters		Wang Xiaoyong (employee ID: 05574), Zhang Haitao (employee ID: 03221), Wei Yongkang (employee ID: 03093), Wang Hui (employee ID: 00210)	
2019-10-17	1.02	All chapters		Wang Hui (employee ID: 00210)	<ol style="list-style-type: none"> 1. Changed the Num field in the data structure to the Total and Offset fields, and the PersonInfoList field to the PersonInfo field for the personnel information query API. 2. Changed the data format of the CurrentTime and TerminationTime fields to UTC seconds in packet response example of the subscription creation and refresh APIs. 3. Changed the value of the Content-Type field to text/plain in packet headers in record push requests. 4. Added the personnel library query API. 5. Added personnel information APIs and deleted the code examples and descriptions of some fields related to time template.
2019-10-29	1.02	All chapters		Wang Hui (employee ID: 00210)	<ol style="list-style-type: none"> 1. Modified the descriptions of the IPAddress and Port fields for subscription APIs and added the description of "subscriber". 2. Named the LibMatchInfoList structure CtrlLibMatInfo in push messages.
2019-12-24	1.02	All chapters		Wang Xiaoyong (employee ID: 05574)	<ol style="list-style-type: none"> 1. Changed the type of TimeTemplateNum to unsigned long in section 4.3 "Personnel Information Processing". 2. Modified section 3.2 "Basic Service Process in Internet Networking".
2020-02-18	1.02	All chapters		Wang Hui (employee ID: 00210)	<ol style="list-style-type: none"> 1. Added the body temperature and gauze mask fields to the FaceInfoList array and the face attribute abnormality status to the MatchStatus field in section 4.4.4 "Record Push".
2020-02-24	1.02	4.1.1 "Configuring a Keep-alive Connection"		Shi Jiahui (employee ID: 06085)	<ol style="list-style-type: none"> 1. Replaced original screenshots. 2. Modified some descriptions.

Thank you for purchasing our product. If you have any questions or need assistance, please contact the dealer.

Disclaimer

No part of this manual may be copied, reproduced, translated, or distributed in any form or by any means without prior written consent from Uniview.

The manual may be updated from time to time due to version upgrade or other reasons.




The manual is for reference only. All the statements, information, and suggestions contained herein do not constitute warranties of any kind, express or implied.

Conventions

- The figures, charts or photos in this manual are used for illustration only, which may differ from the actual product.
- Subject to uncertain factors such as the physical environment, actual values of some data may differ from the reference values described herein. In case of any doubt or dispute, the right of final interpretation resides with Uniview.
- Follow this manual when using the product. Professional guidance is recommended.
- Notational conventions used in this document are described as follows:

Format	Description
Boldface	Indicates buttons, menus, tabs, window names, dialog names, and parameter names. For example, click OK or select Device Management .
" "	Indicates messages. For example, "Hanging Up" is displayed on the interface.
>	Directs you to go to a multi-level menu. For example, go to Device Management > Add Device . In this example, Add Device is a submenu under Device Management .

- The symbols in the following table may be found in this manual. Carefully follow the instructions indicated by the symbols to avoid hazardous situations and use the product properly.

Symbol	Description
 WARNING!	Contains important safety instructions and indicates situations that could cause bodily injury.
 CAUTION!	Means reader be careful and improper operations may cause damage or malfunction to product.
 NOTE!	Means useful or supplemental information about the use of product.

Contents

1 Things to Know Before Development	1
1.1 Overview.....	1
1.2 API Use Methods and Rules.....	1
1.2.1 Calling Method.....	1
1.2.2 Example of a Request Packet.....	1
1.2.3 Example of a Response Packet.....	2
1.2.4 JSON.....	3
1.2.5 Parameter Requirements.....	3
1.3 Applicability.....	3
2 Application Scenarios	3
2.1 Terms.....	3
2.2 LAN Networking.....	4
2.3 Internet Networking.....	4
3 Service Process	5
3.1 Basic Service Process in LAN Networking.....	5
3.2 Basic Service Process in Internet Networking.....	6
3.3 Visual Intercom Service Process.....	8
3.3.1 An Indoor Monitor Delivers Binding Information to a Face Recognition Access Control Terminal.....	8
3.3.2 The Face Recognition Access Control Terminal Initiates a Call and the Indoor Monitor Answers the Call.....	9
3.3.3 The Indoor Monitor Opens the Door Remotely.....	9
3.3.4 The Face Recognition Access Control Terminal or Indoor Monitor Hangs Up.....	9
4 APIs	10
4.1 Keep-alive Connection.....	10
4.1.1 Configuring a Keep-alive Connection.....	10
4.1.2 Sending Heartbeat Data.....	11
4.2 Device Information.....	14
4.2.1 Querying the IP Address, Serial Number, and Version of a Device.....	14

4.2.2 Querying the Online Status of a Device	15
4.3 Personnel Information Processing.....	16
4.3.1 Querying the Personnel Library	16
4.3.2 Adding Persons.....	18
4.3.3 Modifying Person Information	23
4.3.4 Deleting Person Information.....	28
4.3.5 Querying Person Information.....	29
4.4 Access Control Record Push	33
4.4.1 Creating a Subscription	33
4.4.2 Refreshing a Subscription.....	36
4.4.3 Deleting a Subscription	37
4.4.4 Pushing Records.....	38
4.5 Visual Intercom.....	47
4.5.1 Querying the Current Location of a Face Recognition Access Control Terminal	47
4.5.2 Obtaining Binding Information About an Indoor Monitor	48
4.5.3 Setting Binding Information for an Indoor Monitor.....	50
4.5.4 Deleting Binding Information About an Indoor Monitor	51
4.5.5 Delivering the Call Status	53
4.5.6 Reporting the Call Status.....	54
4.6 Device Control	55
4.6.1 Opening the Door Remotely	55
5 Appendices	55
5.1 Error Code Description	55
5.2 Debugging Tool.....	57

1 Things to Know Before Development

1.1 Overview

This document is applicable when UNV face recognition terminals and face recognition access control terminals need to access various third-party platforms so that the third-party platforms can manage personnel information, push generated records, and perform other operations. It is also applicable when UNV face recognition access control terminals interconnect to third-party indoor monitors for visual intercom.

1.2 API Use Methods and Rules

1.2.1 Calling Method

Face recognition terminals and face recognition access control terminals interconnect to third-party platforms through standard HTTP APIs.

Face recognition access control terminals interconnect to third-party indoor monitors through standard HTTP APIs.

The following HTTP calling methods are supported:

- PUT: Updates a resource. A request message must contain some or all the members of the resource.
- POST: Creates a resource. A request message must contain one member of the resource.
- GET: Requests specified resource information.
- DELETE: Deletes one or all resource members that support the POST method.

1.2.2 Example of a Request Packet

```
POST /LAPI/V1.0/PACS/Controller/Event/Subscriptions HTTP/1.1
Content-Type: application/json
Content-Length: 190
Connection: close
```

```
{
  "AddressType": 0,
  "IPAddress": "192.174.12.95",
  "Port": 8080,
  "Duration": 600,
  "Type": 1,
  "LibIDNum": 1,
  "LibIDList": [
    {
      "LibID": 3
    }
  ]
}
```

```
    ]  
  }  
}
```

When the PUT or POST method is used, the body of an HTTP request usually contains JSON data. Requests must be sent to the HTTP/HTTPS port of a device.

1.2.3 Example of a Response Packet

```
HTTP/1.1 200 Ok  
Content-Length: 258  
Content-Type: text/plain  
Connection: close  
  
{  
  "Response": {  
    "ResponseURL": "/LAPI/V1.0/PeopleLibraries/BasicInfo",  
    "CreatedID": -1,  
    "ResponseCode": 0,  
    "ResponseString": "Succeed",  
    "StatusCode": 0,  
    "StatusString": "Succeed",  
    "Data": "null"  
  }  
}
```

- **ResponseURL:** Indicates the URL carried in an HTTP request message. When the GET, PUT, or DELETE request method is used, the URL is the same as that in the request. When the POST method is used, the value is the URL of the new object. Clients can use the URL to obtain the new object directly.
- **CreatedID:** Indicates that the resource supports the HTTP POST/DELETE method. The value is the ID of the new object, which is created by the server.
- **ResponseCode:** Indicates the system processing result. For details, see the description of error codes.
- **ResponseString:** Explains the system processing result.
- **StatusCode:** Indicates the service processing result. For details, see the description of error codes.
- **StatusString:** Explains the service processing result.
- **Data:** Indicates information or data of the requested or obtained resource. Data is in the JSON format in most cases. For the same URL, the request is the same as the data in get response. If the server fails to work, or the adopted HTTP method is PUT or POST, the value of **Data** is **null**.

1.2.4 JSON

For more information about JSON, visit www.json.org. **JSON string values can contain any Unicode characters except quotation marks ("), backslash (\), and control characters.**

1.2.5 Parameter Requirements

- **C:** Determines whether a byte exists based on the value in the returned result; not specific to certain products.
- **O:** Indicates that a field is optional. If the field does not exist, the client does not consider it as an error during parsing.
- **M:** Indicates that a node exists under any circumstance. If it does not exist, the client can process it as an error.

1.3 Applicability

APIs described in this document are applicable to UNV face recognition terminals and face recognition access control terminals.

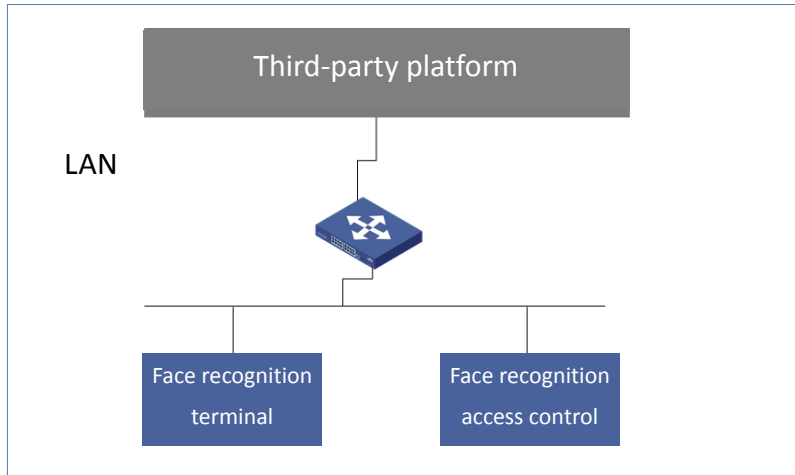
2 Application Scenarios

2.1 Terms

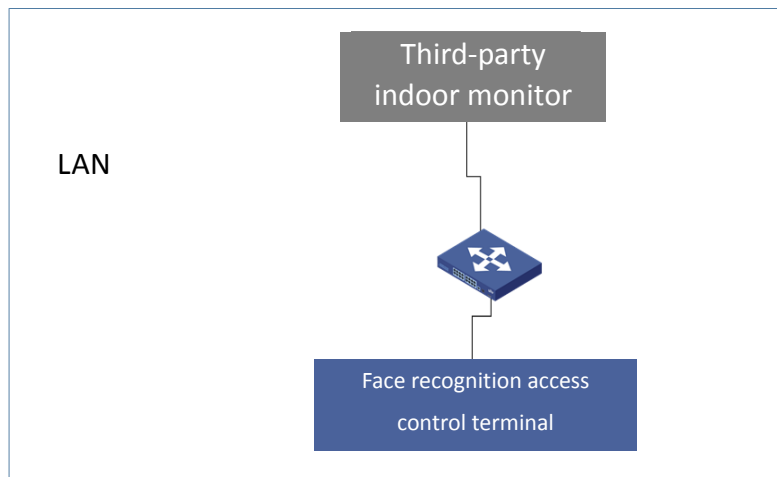
- **Face recognition terminal:** installed on a PTS gate machine. This access control device supports face recognition and controls the opening/closing of the gate machine.
- **Face recognition access control terminal:** mainly installed on a wall. This access control device supports face recognition and visual intercom, and controls the electric lock to open/close the door.
- **Outdoor monitor:** The face recognition access control terminal also serves as an outdoor monitor when providing the visual intercom service.
- **Third-party indoor monitor:** developed by a third party and mainly installed on the indoor wall. This access control device is used in combination with a face recognition access control terminal to implement the visual intercom function.
- **Third-party platform:** developed by a third party and deployed on a LAN or Internet. This software/hardware platform connects to access control devices via a network and manages the access control devices and personnel, and receives records.

2.2 LAN Networking

In the LAN networking scenario, all devices are installed on a LAN and can communicate with each other directly. The figure below shows the network topology of the access control service on a LAN.



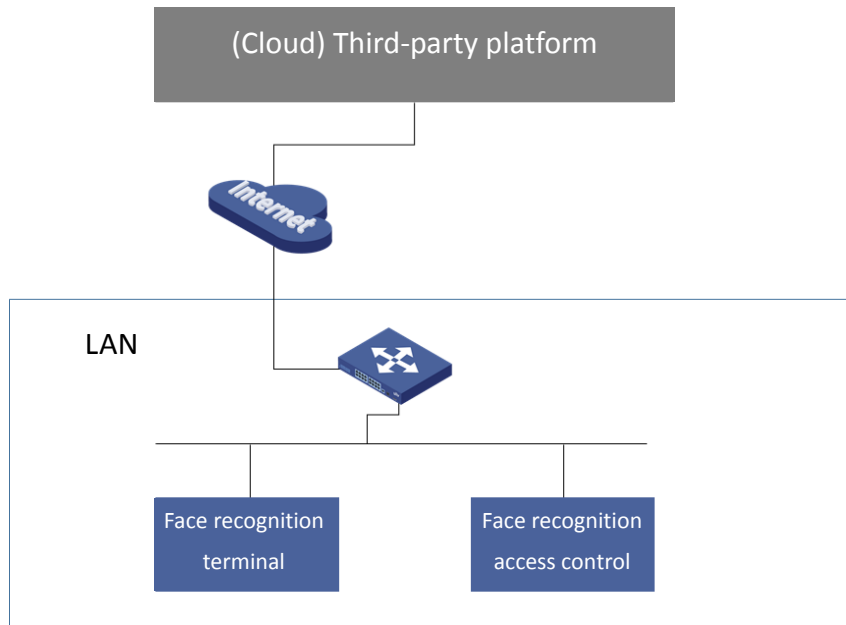
The figure below shows the network topology of the visual intercom service on a LAN.



2.3 Internet Networking

In the Internet networking scenario, terminals are deployed on a LAN, third-party platforms are deployed on the Internet, and the terminals need to communicate with the platforms via Internet.

The figure below shows the network topology of the access control service on the Internet.

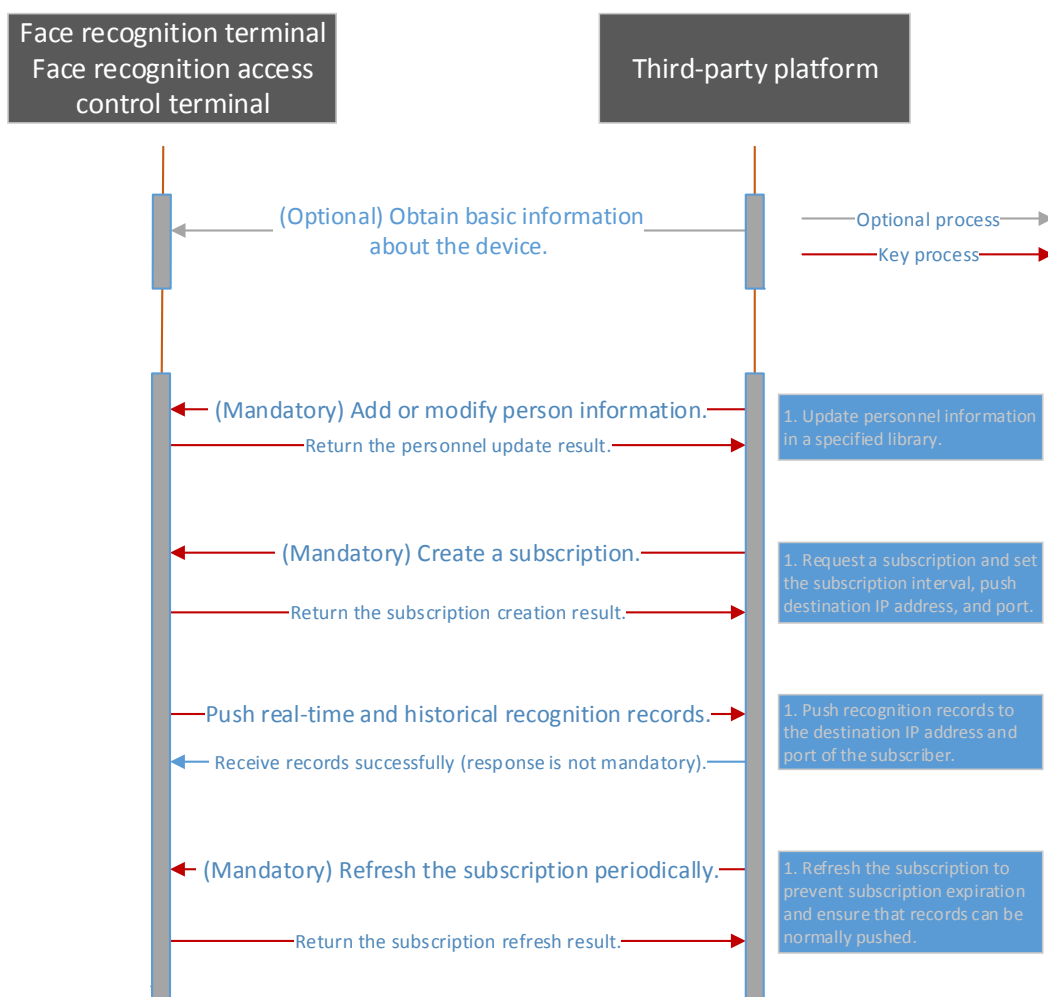


3 Service Process

3.1 Basic Service Process in LAN Networking

In LAN networking, a third-party platform can directly communicate with terminals through L APIs to implement configuration delivery, personnel information delivery, record subscription, and data push.

The figure below shows the basic service process in LAN networking.



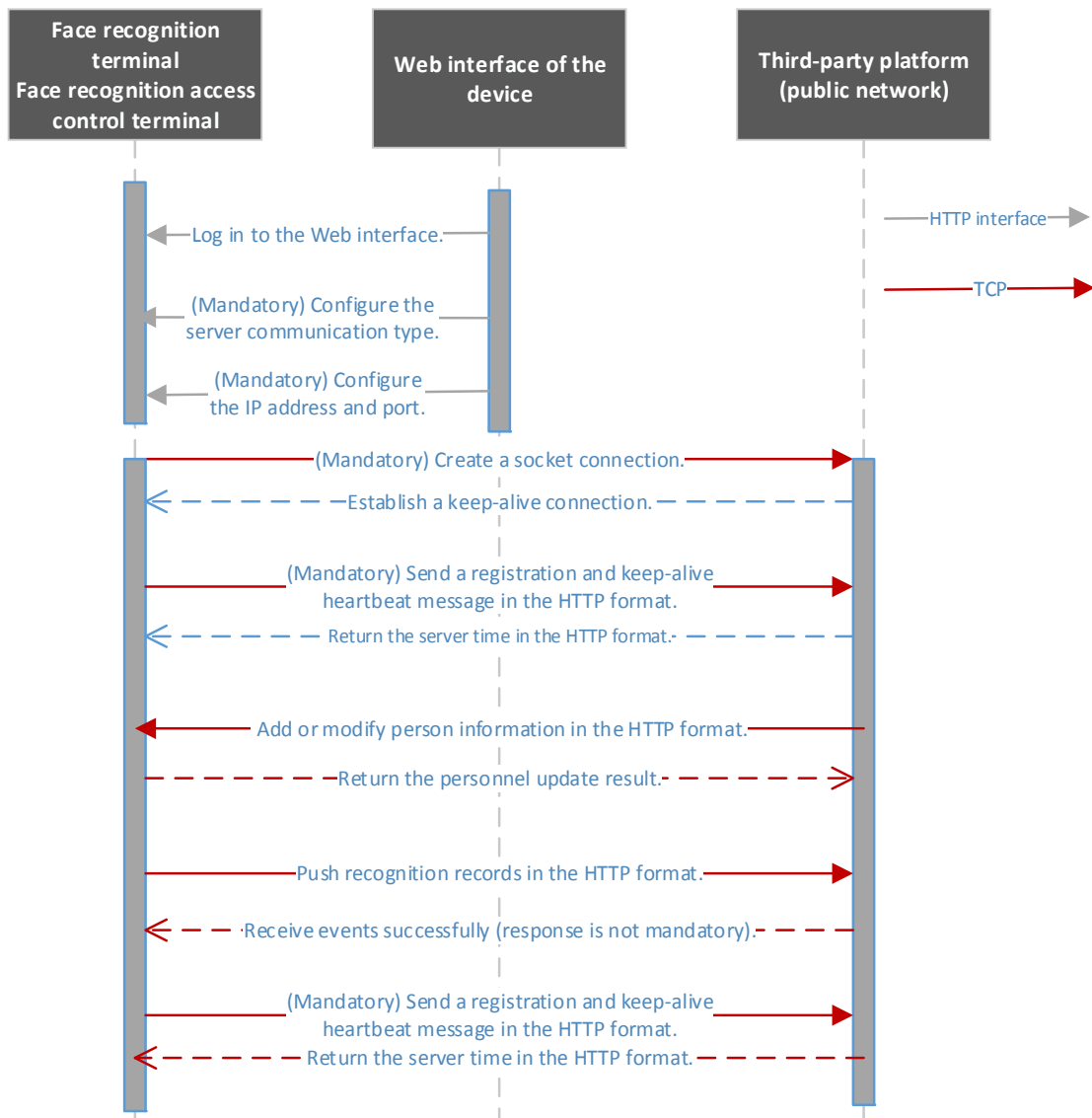
Relevant APIs are described as follows:

- For the API for obtaining basic information about a device, see [4.2.1 "Querying the IP Address, Serial Number, and Version of a Device"](#).
- For the API for adding personnel information, see [4.3.2 "Adding Persons"](#).
- For the API for modifying personnel information, see [4.3.3 "Modifying Person Information"](#).
- For the API for creating a subscription, see [4.4.1 "Creating a Subscription"](#).
- For the API for pushing real-time and historical recognition records, see [4.4.4 "Pushing Records"](#).
- For the API for refreshing a subscription periodically, see [4.4.2 "Refreshing a Subscription"](#).

3.2 Basic Service Process in Internet Networking

In Internet networking, Network Address Translation (NAT) between the public and private networks is involved. Keep-alive connection is required for network connection establishment, and then configuration delivery, personnel information delivery, record subscription, and data push.

The figure below shows the basic service process in Internet networking.



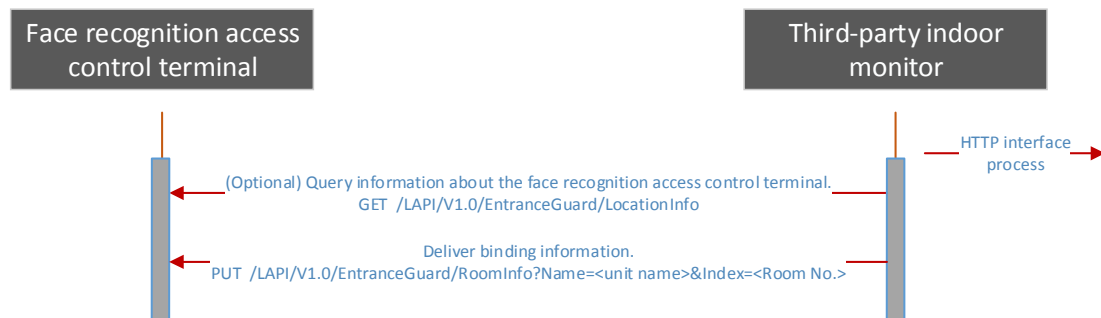
Relevant APIs are described as follows:

- For the API for configuring the server communication type, IP address, and port, see [4.1.1 "Configuring a Keep-alive Connection"](#).
- For the API for sending the registration and keep-alive heartbeat messages, see [4.1.2 "Sending Heartbeat Data"](#).
- For the API for obtaining basic information about a device, see [4.2.1 "Querying the IP Address, Serial Number, and Version of a Device"](#).
- For the API for adding personnel information, see [4.3.2 "Adding Persons"](#).
- For the API for modifying personnel information, see [4.3.3 "Modifying Person Information"](#).
- For the API for pushing real-time and historical recognition records, see [4.4.4 "Pushing Records"](#).

3.3 Visual Intercom Service Process

3.3.1 An Indoor Monitor Delivers Binding Information to a Face Recognition Access Control Terminal

Before the visual intercom function is used, binding relationships, that is, associated indoor monitors, need to be delivered to a face recognition access control terminal. The binding relationship can be delivered by associated indoor monitors or the platform.



Relevant APIs are described as follows:

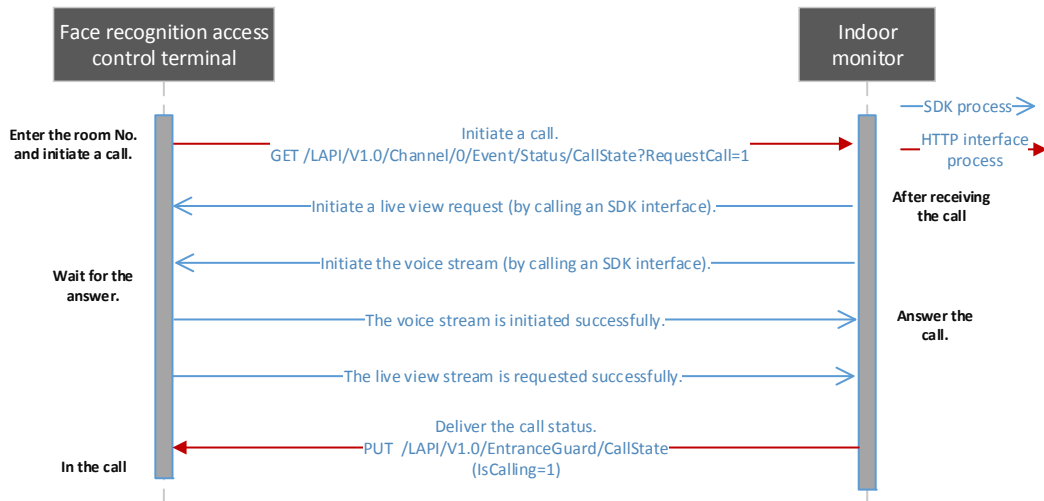
- For the API for querying information about a face recognition access control terminal, see [4.5.1 "Querying the Current Location of a Face Recognition Access Control Terminal"](#).
- For the API for delivering binding information, see [4.5.3 "Setting Binding Information for an Indoor Monitor"](#).



NOTE!

The building and unit in the binding information delivered by an indoor monitor must be consistent with those in the face recognition access control terminal. You can obtain buildings and units in the face recognition access control terminal for verification by using the API described in [4.5.1 "Querying the Current Location of a Face Recognition Access Control Terminal"](#).

3.3.2 The Face Recognition Access Control Terminal Initiates a Call and the Indoor Monitor Answers the Call

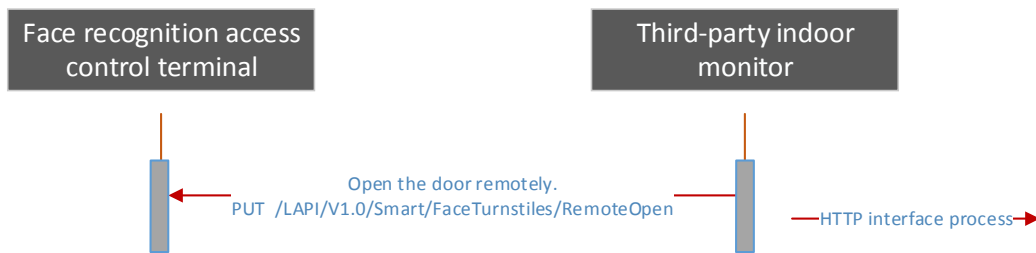


Relevant APIs are described as follows:

- For the API for initiating a call, see [4.5.6 "Reporting the Call Status"](#).
- For the API for delivering the call status, see [4.5.5 "Delivering the Call Status"](#).

3.3.3 The Indoor Monitor Opens the Door Remotely

During a call or in non-call state, the indoor monitor can send a remote door opening message to control the bound outdoor monitor to open the door.

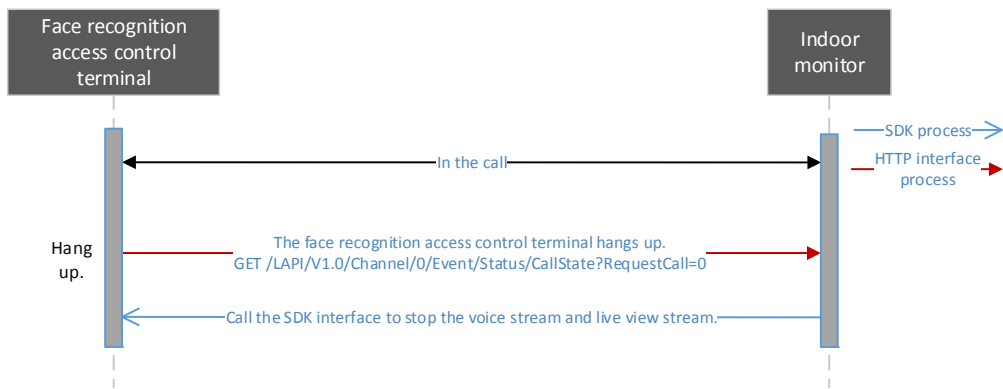


Relevant APIs are described as follows:

- For the API for remotely opening the door, see [4.6.1 "Opening the Door Remotely"](#).

3.3.4 The Face Recognition Access Control Terminal or Indoor Monitor Hangs Up

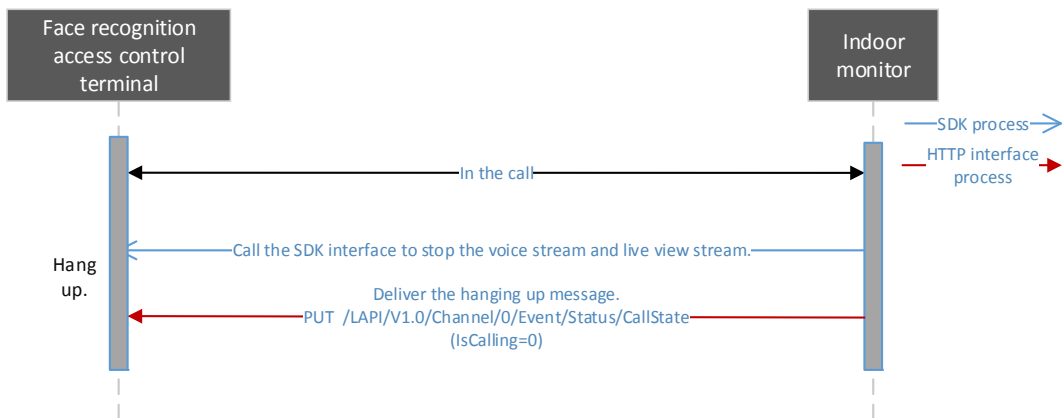
The figure below shows the process in which the face recognition access control terminal hangs up.



Relevant APIs are described as follows:

- For the API for enabling the face recognition access control terminal to hang up, see [4.5.6 "Reporting the Call Status"](#).

The figure below shows the process in which the indoor monitor hangs up.



Relevant APIs are described as follows:

- For the API for delivering the hanging up information, see [4.5.5 "Delivering the Call Status"](#).

4 APIs

4.1 Keep-alive Connection

4.1.1 Configuring a Keep-alive Connection

■ Function description

The server does not need to initiate a subscription. A terminal initiates a keep-alive connection for NAT according to the configuration in **LAPI V2**.

■ Web configuration guide

- Log in to the Web configuration page.

- Choose **Setup > Common > Server** and click the **Intelligent Server** tab.
- Select **LAPI V2** from the **Platform Communication Type** drop-down list. (If this option does not exist, contact engineers in the local office to upgrade the firmware version.)
- Configure the server address, port, and heartbeat keep-alive interval.

4.1.2 Sending Heartbeat Data

- **Function description**
This API is used by terminals to actively send heartbeat packets to a cloud third-party platform.

**NOTE!**

1. The keep-alive interval can be configured on the Web interface as required. The recommended value is **600** seconds.
2. If the interval returned from the cloud differs from that on a terminal, the terminal changes the local interval.

■ Calling direction

A face recognition terminal or face recognition access control terminal calls the API to a third-party platform.

■ Request description

- Request method: POST
- Request URL: /LAPI/V1.0/PACS/Controller/HeartReportInfo
- Content-Type: application/json

■ Request parameters

Example:

POST /LAPI/V1.0/PACS/Controller/HeartReportInfo HTTP/1.1

Content-Type: application/json

Content-Length: 178

```
{
  "RefId": "2936f461-6e79-465c-996d-f7ddb9660346",
  "Time": "1900-01-00 00:00:00",
  "NextTime": "1970-01-01 08:00:01",
  "DeviceCode": "210235C31L3186000023",
  "DeviceType": 1
}
```

Param	Requirement	Type	Description	Example
RefId	M	string	Request ID (UUID) Length range: [0, 36]	"30f83ce3-bba2-48b5-bc75-8f0b724c5f00"
Time	M	string	Current heartbeat report time Length range: [0, 20]	"2019-04-16 20:13:45"
NextTime	M	string	Next heartbeat report time Length range: [0, 20]	"2019-04-16 8:18:45 PM"

DeviceCode	M	string	Device code (serial number) Length range: [0, 24]	"210235C31L3186000023"
DeviceType	O	Unsigned long	Device type 1: common access control device 2: visual intercom access control device 3: attendance machine 4: input device 5: indoor monitor	1

■ Returned results

Example:

HTTP/1.1 200 Ok

Content-Length: 139

Content-Type: text/plain

Connection: close

X-Frame-Options: SAMEORIGIN

```
{
  "ResponseURL": "/LAPI/V1.0/PACS/Controller/HeartReportInfo",
  "Code": 0,
  "Data": {
    "Time": "2019-04-22 16:17:45"
  }
}
```

■ Returned parameters

Param	Requirement	Type	Description	Example
Response URL	M	string	Response URL, that is, the URL for which the request is responded. Length range: [0, 128]	"/LAPI/V1.0/PACS/Controller/HeartReportInfo"
Code	M	Unsigned long	Response code (to be added) 0: succeeded 1: failed	0
Data	M	Json	Response data	/
Time	M	string	Current time at the cloud Length range: [0, 20]	"2019-04-16 20:13:45"

4.2 Device Information

4.2.1 Querying the IP Address, Serial Number, and Version of a Device

■ Function description

This API is used to obtain basic information about a device.

■ Calling direction

A third-party platform calls the API to a face recognition terminal or face recognition access control terminal.

■ Request description

- Request method: GET
- Request URL: /LAPI/V1.0/System/DeviceBasicInfo
- Content-Type: application/json

■ Returned results

Example:

```
{
  "Response": {
    "ResponseURL": "/LAPI/V1.0/System/DeviceBasicInfo",
    "CreatedID": -1,
    "ResponseCode": 0,
    "ResponseString": "Succeed",
    "StatusCode": 0,
    "StatusString": "Succeed",
    "Data": {
      "Manufacturer": "UNIVIEW",
      "DeviceModel": "EG131-HF",
      "DeviceConfig": "",
      "SerialNumber": "210235C31L3186000023",
      "MAC": "48ea63885d47",
      "FirmwareVersion": "PTS_Q2101-B0008P01L29.190218",
      "HardwareID": "A",
      "PCBVersion": "A",
      "UbootVersion": "V1.2",
      "CameraVersion": "",
      "Address": "192.174.12.10",
      "Netmask": "255.255.255.0",
      "Gateway": "192.174.12.1"
    }
  }
}
```

■ Returned parameters

Param	Requirement	Type	Description	Example
Manufacturer	M	String	Manufacturer	UNIVIEW
DeviceModel	M	String	Device model	EG131-HF
DeviceConfig	M	String	Device configuration	/
SerialNuber	M	String	Device serial number	210235C31L3186000023
MAC	M	String	Device MAC address	48ea63885d47
FirmwareVersion	M	String	Firmware version	PTS_Q2101-B0008P01L29.190218
HardwareID	M	String	Hardware ID	A
PCBVersion	M	String	PCB version	A
UbootVersion	M	String	U-Boot version	V1.2
CameraVersion	M	String	Camera module version	/
Address	M	String	IP address	192.174.12.10
Netmask	M	String	Subnet mask	255.255.255.0
Gateway	M	String	Gateway address	192.174.12.1

4.2.2 Querying the Online Status of a Device

■ Function description

This API is used to obtain the online status of a device at an interval of 3s.

■ Calling direction

A third-party platform calls the API to a face recognition terminal or face recognition access control terminal.

■ Request description

- Request method: GET
- Request URL: /LAPI/V1.0/System/KeepAlive
- Content-Type: application/json

■ Returned results

Example:

```
{
  "Response": {
    "ResponseURL": "/LAPI/V1.0/System/KeepAlive",
```

```
    "CreatedID": -1,  
    "ResponseCode": 0,  
    "ResponseString": "Succeed",  
    "StatusCode": 0,  
    "StatusString": "Succeed",  
    "Data": "null"  
  }  
}
```

 **CAUTION!**

If a terminal is offline, the platform receives no response and considers the terminal abnormal.

4.3 Personnel Information Processing

4.3.1 Querying the Personnel Library

■ Function description

This API is called to query the personnel library.

■ Calling direction

A third-party platform calls the API to a face recognition terminal or face recognition access control terminal.

■ Request description

- Request method: GET
- Request URL: /LAPI/V1.0/PeopleLibraries/BasicInfo
- Content-Type: application/json

■ Returned results

Example:

```
{  
  "Response": {  
    "ResponseURL": "/LAPI/V1.0/PeopleLibraries/BasicInfo",  
    "CreatedID": -1,  
    "ResponseCode": 0,  
    "ResponseString": "Succeed",  
    "StatusCode": 0,  
    "StatusString": "Succeed",  
    "Data": {  
      "Num": 2,  
    }  
  }  
}
```

```

"LibList": [
  {
    "ID": 3,
    "Type": 3,
    "PersonNum": 3,
    "MemberNum": 3,
    "FaceNum": 3,
    "LastChange": 1510811178,
    "Name": "Default employee library",
    "BelongIndex": ""
  },
  {
    "ID": 4,
    "Type": 4,
    "PersonNum": 2,
    "MemberNum": 3,
    "FaceNum": 2,
    "LastChange": 1510811178,
    "Name": "Default visitor library",
    "BelongIndex": ""
  }
]
}
}
}

```

■ Returned parameters

Param	Requirement	Type	Description	Example
ID	M	unsigned long	Library ID Only library IDs 3 and 4 are supported currently.	3
Type	M	unsigned long	Type of the personnel library 0: invalid value by default 1: blacklist 2: graylist/stranger 3: employee 4: visitor	3
PersonNum	M	unsigned long	Total number of persons in the library	1
MemberNum	M	unsigned long	Total number of members in the library This is a compatible field and can be	10

			ignored.	
FaceNum	M	unsigned long	Total number of face photos in the library	10
LastChange	M	unsigned long	Last modification time of library information (Unix timestamp)	1510811178
Name	M	string	Library name Length range: [1, 63]	"Default employee library"
BelongIndex	M	string	Unique index that identifies the device to which the library belongs	

4.3.2 Adding Persons

■ Function description

This API is used to add persons to the personnel library.



NOTE!

- If a person does not exist, the person information is added.
If a person already exists, the original person information is deleted and then the new information is added.
- If some face photos are not synchronized successfully, the entire personnel information fails to be added.
- The HTTP packet cannot exceed 6MB (Content-Length: 6291456). **No more than six persons can be added in batches each time (one person can have only one photo).**

■ Calling direction

A third-party platform calls the API to a face recognition terminal or face recognition access control terminal.

■ Request description

- Request method: POST
- Request URL: /LAPI/V1.0/PeopleLibraries/<ID>/People
- Content-Type: application/json

Param	Requirement	Type	Description	Example
<ID>	M	unsigned long	ID of the personnel library	3

Request parameters

Example:

```
{
  "Num": 1,
  "PersonInfoList": [
    {
      "PersonID": 1,
      "LastChange": 1564022548,
      "PersonCode": "1001",
      "PersonName": "uniview",
      "Remarks": "Uniview",
      "TimeTemplateNum": 0,
      "IdentificationNum": 2,
      "IdentificationList": [
        {
          "Type": 1,
          "Number": "12345678"
        },
        {
          "Type": 99,
          "Number": "3214124"
        }
      ],
      "ImageNum": 1,
      "ImageList": [
        {
          "FaceID": 1,
          "Name": "1_1.jpg",
          "Size": 166736,
          "Data": "..."
        }
      ]
    }
  ]
}
```

Param	Requirement	Type	Description	Example
Num	M	unsigned long	Number of persons in the personnel library A maximum of six persons can be added in batches each time.	1
PersonInfoList	M	Array	Personnel information list	/

PersonInfo	C	Json Block	Person information. If Num is 0 , this field is optional.	/
PersonID	M	unsigned long	Person ID, which must be unique	1
LastChange	M	unsigned long	Last modification time of person information (Unix timestamp)	1510925018
PersonCode	M	String	Person code, such as a student ID or an employee ID. Length range: [1, 15]	"1001"
PersonName	M	String	Person name Length range: [1, 63]	Zhang San
Remarks	O	string	Remarks Length range: [1, 63]	"Pan Security Product Development Department I"
TimeTemplateNum	M	unsigned long	Number of time templates This field is mandatory for the compatibility between the new and old APIs.	/
IdentificationNum	M	unsigned long	Number of certificates Range: [0, 2] Note: The PTS supports only one certificate currently.	/
IdentificationList	M	Array	Certificate information	/
IdentificationInfo	C	Json Block	If IdentificationNum is 0 , this field is optional.	/
Type	M	unsigned long	Certificate type 0: ID card; 1: IC card 99: others	0
Number	M	String	Certificate No. Length range: [1, 20]	"33022319890520222X"
ImageNum	M	unsigned long	Number of face photos Range: [0, 6]	/
ImageList	M	Array	Face photo information list	/
FaceID	C	unsigned long	Face photo ID, which must be unique If ImageNum is 0 , this field is optional.	23
FileInfo	C	Json Block	Photo information. For details, refer to FileInfo Json block. When ImageNum is 0 , this field is optional. It is in the same level as FaceID, and	/

			no brackets are required, that is: {"FaceID", "Name"; "Size"; "Data":} Note: File size range: [0, 768KB]	
Name	M	String	File name. Length range: [1, 16]. The delivered photo name is converted into the format of PersonID_FaceID.jpg.	/
Size	M	unsigned long	Data size, with the unit of byte Range: [0, 1048576 (1MB)]	21783
Data	M	String	Photo data in the Base64 encoding format. The data header does not need to begin with base64 but begins with /9j/.	/

■ Returned results

Example:

```
{
  "Response": {
    "ResponseURL": "/LAPI/V1.0/PeopleLibraries/3/People",
    "CreatedID": -1,
    "ResponseCode": 0,
    "ResponseString": "Succeed",
    "StatusCode": 0,
    "StatusString": "Succeed",
    "Data": {
      "Num": 1,
      "PersonList": [
        {
          "PersonID": 1,
          "FaceNum": 1,
          "FaceList": [
            {
              "FaceID": 1,
              "ResultCode": 0
            }
          ]
        }
      ]
    }
  }
}
```

■ Returned parameters

Param	Requirement	Type	Description	Example
Num	M	unsigned long	Number of persons	3
Person ResultList	M	Array	Personnel information result list	/
PersonID	M	unsigned long	Person ID	2
FaceNum	M	unsigned long	Number of face photos	2
FaceList	M	Array	Face photo information result list	/
FaceID	C	unsigned long	Face photo ID If FaceNum is 0, this field is optional.	22
ResultCode	C	unsigned long	Processing result status code 1000: Algorithm initialization fails. 1001: Face detection fails. 1002: No face is found in the picture. 1003: The JPEG picture fails to be decoded. 1004: The face picture does not meet quality requirements. 1005: The picture fails to be zoomed. 1006: The intelligence function is disabled. 1007: The imported picture is undersized. 1008: The imported picture is oversized. 1009: The resolution of the imported picture exceeds 1920x1080. 1010: The imported picture does not exist. 1011: The number of face elements reaches the upper limit. 1012: The intelligent algorithm model does not match. 1013: The certificate ID of the member whose face photo is to be imported into the library is invalid. 1014: The picture format of the member whose face photo is to be imported into the library is incorrect. 1015: The channel arming has reached the upper limit of the device. 1016: Another client is operating the face library. 1017: The face library file is being updated. 1018: JSON deserialization fails. 1019: Base64 decoding fails. 1020: The size of the encoded face photo is different from the photo size in	1001

			the request. If FaceNum is 0 , this field is optional.	
--	--	--	---	--

4.3.3 Modifying Person Information

■ Function description

This API is used to modify information about a person.



NOTE!

Only information about one person can be modified each time. When **Num** is **1**, it indicates a single person.

■ Calling direction

A third-party platform calls the API to a face recognition terminal or face recognition access control terminal.

■ Request description

- Request method: PUT
- Request URL: /LAPI/V1.0/PeopleLibraries/<ID>/People
- Content-Type: application/json

■ Request parameters

Example:

```
{
  "Num": 1,
  "PersonInfoList": [
    {
      "PersonID": 1,
      "LastChange": 1564022548,
      "PersonCode": "1001",
      "PersonName": "uniview",
      "Remarks": "Uniview",
      "TimeTemplateNum": 0,
      "TimeTemplateList": [],
      "IdentificationNum": 2,
      "IdentificationList": [
        {
          "Type": 1,
          "Number": "12345678"
        }
      ]
    }
  ]
}
```

```

    {
      "Type": 99,
      "Number": "3214124"
    }
  ],
  "ImageNum": 1,
  "ImageList": [
    {
      "FaceID": 1,
      "Name": "1_1.jpg",
      "Size": 166736,
      "Data": ""
    }
  ]
}
]
}

```

Param	Requirement	Type	Description	Example
Num	M	unsigned long	Number of persons in the personnel library A maximum of six persons can be added in batches each time.	1
PersonInfoList	M	Array	Personnel information list	/
PersonInfo	C	Json Block	Person information. If Num is 0 , this field is optional.	/
PersonID	M	unsigned long	Person ID, which must be unique	1
LastChange	M	unsigned long	Last modification time of person information (Unix timestamp)	1510925018
PersonCode	M	String	Person code, such as a student ID or an employee ID. Length range: [1, 15]	"1001"
PersonName	M	String	Person name Length range: [1, 63]	Zhang San
Remarks	O	string	Remarks Length range: [1, 63]	"Pan Security Product Development Department I"
TimeTemplateNum	M	unsigned long	Number of time templates	/
TimeTemplateList	O	array	Time template information list	/
PersonTimeTempl	M	Json	Time template information about	/

atInfo		Block	the person	
BeginTime	M	unsigned long	Start time of the validity period of the time template (Unix timestamp) If it is not configured, set the value to 0 .	1510925018
EndTime	M	unsigned long	End time of the validity period of the time template (Unix timestamp). If it is not configured, set the value to 4294967295 (0xFFFFFFFF).	1510925018
Index	M	unsigned long	Index of the time template If it is not configured, set the value to 0 .	3
IdentificationNum	M	unsigned long	Number of certificates Range: [0, 2] Note: The PTS supports only one certificate currently.	/
IdentificationList	M	Array	Certificate information	/
IdentificationInfo	C	Json Block	If IdentificationNum is 0 , this field is optional.	/
Type	M	unsigned long	Certificate type 0: ID card; 1: IC card 99: others	0
Number	M	String	Certificate No. Length range: [1, 20]	"33022319890520222X"
ImageNum	M	unsigned long	Number of face photos Range: [0, 6]	/
ImageList	M	Array	Face photo information list	/
FaceID	C	unsigned long	Face photo ID, which must be unique If ImageNum is 0 , this field is optional.	23
FileInfo	C	Json Block	Photo information. For details, refer to FileInfo Json block. When ImageNum is 0 , this field is optional. It is in the same level as FaceID, and no brackets are required, that is: <pre>{"FaceID", "Name":, "Size":, "Data":}</pre> Note: File size range: [0, 768KB]	/

Name	M	String	File name. Length range: [1, 16]. The delivered photo name is converted into the format of PersonID_FaceID.jpg.	/
Size	M	unsigned long	Data size, with the unit of byte Range: [0, 1048576 (1MB)]	21783
Data	M	String	Photo data in the Base64 encoding format. The data header does not need to begin with base64 but begins with /9j/.	/

■ Returned results

Example:

```
{
  "Response": {
    "ResponseURL": "/LAPI/V1.0/PeopleLibraries/3/People",
    "CreatedID": -1,
    "ResponseCode": 0,
    "ResponseString": "Succeed",
    "StatusCode": 0,
    "StatusString": "Succeed",
    "Data": {
      "Num": 1,
      "PersonList": [
        {
          "PersonID": 1,
          "FaceNum": 1,
          "FaceList": [
            {
              "FaceID": 1,
              "ResultCode": 0
            }
          ]
        }
      ]
    }
  }
}
```

■ Returned parameters

Param	Requirement	Type	Description	Example
Num	M	unsigned long	Number of persons	3
Person ResultList	M	Array	Personnel information result list	/

PersonID	M	unsigned long	Person ID	2
FaceNum	M	unsigned long	Number of face photos	2
FaceList	M	Array	Face photo information result list	/
FaceID	C	unsigned long	Face photo ID If FaceNum is 0 , this field is optional.	22
ResultCode	C	unsigned long	Processing result status code 1000: Algorithm initialization fails. 1001: Face detection fails. 1002: No face is found in the picture. 1003: The JPEG picture fails to be decoded. 1004: The face picture does not meet quality requirements. 1005: The picture fails to be zoomed. 1006: The intelligence function is disabled. 1007: The imported picture is undersized. 1008: The imported picture is oversized. 1009: The resolution of the imported picture exceeds 1920x1080. 1010: The imported picture does not exist. 1011: The number of face elements reaches the upper limit. 1012: The intelligent algorithm model does not match. 1013: The certificate ID of the member whose face photo is to be imported into the library is invalid. 1014: The picture format of the member whose face photo is to be imported into the library is incorrect. 1015: The channel arming has reached the upper limit of the device. 1016: Another client is operating the face library. 1017: The face library file is being updated. 1018: JSON deserialization fails. 1019: Base64 decoding fails. 1020: The size of the encoded face photo is different from the photo size in the request. If FaceNum is 0 , this field is optional.	1001

4.3.4 Deleting Person Information

■ Function description

This API is used to delete information about a specified person.

■ Calling direction

A third-party platform calls the API to a face recognition terminal or face recognition access control terminal.

■ Request description

- Request method: DELETE
- Request URL:
/LAPI/V1.0/PeopleLibraries/<ID>/People/<ID>?LastChange=<LastChange>
- Content-Type: application/json

■ Request parameters

Param	Requirement	Type	Description	Example
<ID>	M	unsigned long	ID of the personnel library Note: PeopleLibraries/<ID>	3
<ID>	M	unsigned long	Person ID Note: People/<ID>	3
LastChange	M	unsigned long	Last modification time with a timestamp It is used to synchronize with the last modification time of the library.	1510925018

■ Returned results

Example:

```
{
  "Response": {
    "ResponseURL": "/LAPI/V1.0/PeopleLibraries/3/People/1020",
    "CreatedID": -1,
    "ResponseCode": 0,
    "ResponseString": "Succeed",
    "StatusCode": 0,
    "StatusString": "Succeed",
    "Data": "null"
  }
}
```

4.3.5 Querying Person Information

■ Function description

This API is used to query person information by different criteria.

■ Calling direction

A third-party platform calls the API to a face recognition terminal or face recognition access control terminal.

■ Request description

- Request method: POST
- Request URL: /LAPI/V1.0/PeopleLibraries/<ID>/People/Info
- Content-Type: application/json

Param	Requirement	Type	Description	Example
<ID>	M	unsigned long	ID of the personnel library	3

■ Request parameters

Example:

```
{
  "Num": 0,
  "QueryInfos": [
    {
      "QryType": 27,
      "QryCondition": 0,
      "QryData": "1001"
    },
    {
      "QryType": 55,
      "QryCondition": 0,
      "QryData": "Uniview"
    }
  ],
  "Limit": 10,
  "Offset": 0
}
```

Param	Requirement	Type	Description	Example
Num	M	unsigned long	Number of search criteria	3
QueryInfos	C	Array	Search criteria list. When Num is 0 , this field is not carried.	/
QueryInfo	M	Json Block	Search criteria details. For details, see QueryInfo json block.	/
Limit	M	unsigned long	Number of entries that can be queried each time. The maximum value is 20.	12
Offset	M	unsigned long	Search begin number, starting from 0	0

QueryInfo:

Param	Requirement	Type	Description	Example
QryType	M	unsigned long	Search criteria 27: employee ID 55: person name 58: certificate number Information about all persons is obtained if no search criteria is specified.	4
QryCondition	M	unsigned long	Logic type of the search criteria 0: equal	0
QryData	C	string	Rvalue of the search criteria	"1476028800"

- Returned results

Example:

```
{
  "Response": {
    "ResponseURL": "/LAPI/V1.0/PeopleLibraries/3/People/Info",
    "CreatedID": -1,
    "ResponseCode": 0,
    "SubResponseCode": 0,
    "ResponseString": "Succeed",
    "StatusCode": 0,
    "StatusString": "Succeed",
    "Data": {
      "Total": 1,
      "Offset": 0,
      "PersonList": [
```

```

{
  "PersonID": 1,
  "LastChange": 1564022548,
  "PersonCode": "1001",
  "PersonName": "uniview",
  "Remarks": "Uniview",
  "TimeTemplateNum": 0,
  "TimeTemplateList": [],
  "IdentificationNum": 2,
  "IdentificationList": [
    {
      "Type": 1,
      "Number": "12345678"
    },
    {
      "Type": 99,
      "Number": "3214124"
    }
  ],
  "ImageNum": 1,
  "ImageList": [
    {
      "FaceID": 1,
      "Name": "1_1.jpg",
      "Size": 166736,
      "Data": ""
    }
  ]
}

```

■ Returned parameters

Param	Requirement	Type	Description	Example
Total	M	unsigned long	Total number of entries that meet the search criteria	500
Offset	M	unsigned long	Current number, starting from 0.	0
PersonInfoList	M	Array	Personnel information list	/
PersonInfo	C	Json Block	Person information. If Num is 0, this field is optional.	/

PersonID	M	unsigned long	Person ID, which must be unique	1
LastChange	M	unsigned long	Last modification time of person information (Unix timestamp)	1510925018
PersonCode	M	String	Person code, such as a student ID or an employee ID. Length range: [1, 15]	"1001"
PersonName	M	String	Person name Length range: [1, 63]	Zhang San
Remarks	O	string	Remarks Length range: [1, 63]	"Pan Security Product Development Department I"
TimeTemplateNum	M	unsigned long	Number of time templates	/
TimeTemplateList	O	array	Time template information list	/
PersonTimeTemplateInfo	M	Json Block	Time template information about the person	/
BeginTime	M	unsigned long	Start time of the validity period of the time template (Unix timestamp) If it is not configured, set the value to 0.	1510925018
EndTime	M	unsigned long	End time of the validity period of the time template (Unix timestamp). If it is not configured, set the value to 4294967295 (0xFFFFFFFF).	1510925018
Index	M	unsigned long	Index of the time template If it is not configured, set the value to 0.	3
IdentificationNum	M	unsigned long	Number of certificates Range: [0, 2] Note: The PTS supports only one certificate currently.	/
IdentificationList	M	Array	Certificate information	/
IdentificationInfo	C	Json Block	If IdentificationNum is 0, this field is optional.	/
Type	M	unsigned long	Certificate type 0: ID card; 1: IC card 99: others	0
Number	M	String	Certificate No. Length range: [1, 20]	"33022319890520222X"
ImageNum	M	unsigned long	Number of face photos Range: [0, 6]	/

ImageList	M	Array	Face photo information list	/
FaceID	C	unsigned long	Face photo ID, which must be unique If ImageNum is 0 , this field is optional.	23
FileInfo	C	Json Block	Photo information. For details, refer to FileInfo Json block. When ImageNum is 0 , this field is optional. It is in the same level as FaceID, and no brackets are required, that is: <pre> {"FaceID", "Name"; "Size"; "Data":;} </pre> Note: File size range: [0, 768KB]	/
Name	M	String	File name. Length range: [1, 16]. Format: PersonID_FaceID.jpg.	/
Size	M	unsigned long	Data size , with the unit of byte Range: [0, 1048576 (1MB)]	21783
Data	M	String	Photo data in the Base64 encoding format. The data header does not need to begin with base64 but begins with /9j/.	/

4.4 Access Control Record Push

4.4.1 Creating a Subscription

■ Function description

This API is used to create a record push subscription.



NOTE!

1. **Only two subscribers** are supported currently. The subscription ID starts from 0.
2. If the third-party platform encounters a power failure and the terminal subscription already exists, re-subscription will fail. In this case, refresh the subscription on the third-party platform. If the subscription fails, create a subscription again.
3. **This API is not required for keep-alive connection. It is applicable only to short connection in LAN networking.**

■ Calling direction

A third-party platform calls the API to a face recognition terminal or face recognition access control terminal.

Request description

- Request method: POST
- Request URL: /LAPI/V1.0/System/Event/Subscription
- Content-Type: application/json

Request parameters

Example:

```
{
  "AddressType": 0,
  "IPAddress": "204.2.1.20",
  "Port": 5118,
  "Duration": 600,
  "Type": 1024,
  "SubscribePersonCondition": {
    "LibIDNum": 1,
    "LibIDList": [
      {
        "LibID": 3
      }
    ]
  }
}
```

Param	Requirement	Type	Description	Example
AddressType	M	unsigned long	IP address type 0: IPv4 address 1: IPv6 address Currently, only IPv4 addresses are supported.	1
IPAddress	M	string	Device IP address of the subscriber Length range: [0, 64]	206.5.99.17
Port	M	unsigned long	Device port of the subscriber Range: [1, 65535]	80
Duration	M	unsigned long	Subscription time, in seconds Range: [30, 3600]	60
Type	C	unsigned long	Subscription type 1024: person verification	Type
SubscribePersonCondition	C	Json Block	Subscription content. See SubscribePersonConditionJson Block.	SubscribePersonCondition

LibIDNum	M	unsigned long	Number of subscribed library IDs When the subscription type is 0, the LibIDNum and LibIDList fields can be omitted. When the subscription type is 1, the number of subscribed library IDs and the list of the IDs need to be specified. When LibIDNum is 0xFFFF , all libraries are subscribed.	0
LibIDList	M	Array	List of subscribed library IDs	/
LibID	C	unsigned long	If LibIDNum is 0 , this field is optional.	3

■ Returned results

Example:

```
{
  "Response": {
    "ResponseURL": "/LAPI/V1.0/System/Event/Subscription",
    "CreatedID": 0,
    "ResponseCode": 0,
    "ResponseString": "Succeed",
    "StatusCode": 0,
    "StatusString": "Succeed",
    "Data": {
      "ID": "0",
      "Reference": "204.2.1.127:5118/Subscription/Subscribers/0",
      "CurrrentTime": 1477104900,
      "TerminationTime": 1477104900
    }
  }
}
```

■ Returned parameters

Param	Requirement	Type	Description	Example
ID	M	unsigned long	Subscription ID	1
Reference	M	string	Subscriber description in the URL format	"192.168.0.13:80/Subscription/Subscribers/0"
CurrentTime	M	unsigned long	Current time in the UTC format, in seconds	/
TerminationTime	M	unsigned long	End time in the UTC format, in seconds	/

4.4.2 Refreshing a Subscription

■ Function description

This API is used to refresh the access control subscription. Periodical refresh can ensure that the server and terminal are always connected.



NOTE!

1. This API is not required for keep-alive connection. It is applicable only to short connection in LAN networking.

■ Calling direction

A third-party platform calls the API to a face recognition terminal or face recognition access control terminal.

■ Request description

- Request method: PUT
- Request URL: /LAPI/V1.0/System/Event/Subscription/<ID>
- Content-Type: application/json

■ Request parameters

Example:

```
{
  "Duration": 3600
}
```

Param	Requirement	Type	Description	Example
<ID>	M	unsigned long	Subscription ID It is the subscription ID returned by a device when a subscriber subscribes to access control notifications.	0
Duration	M	unsigned long	Subscription time, in seconds Range: [30, 3600]	3600

■ Returned results

Example:

```
{
  "Response": {
    "ResponseURL": "/LAPI/V1.0/System/Event/Subscription/0",
    "CreatedID": 0,
    "ResponseCode": 0,
  }
}
```

```

"ResponseString": "Succeed",
"StatusCode": 0,
"StatusString": "Succeed",
"Data": {
  "Reference": "204.2.1.127:5118/Subscription/Subscribers/0",
  "CurrntTime": 1477104900,
  "TerminationTime": 1477104900
}
}
}

```

■ Returned parameters

Param	Requirement	Type	Description	Example
Reference	M	string	Subscriber description in the URL format	"192.168.0.13:80/Subscription/Subscribers/0"
CurrentTime	M	unsigned long	Current time in the UTC format, in seconds	/
TerminationTime	M	unsigned long	End time in the UTC format, in seconds	/

4.4.3 Deleting a Subscription

■ Function description

This API is used to delete an access control subscription.



NOTE!

1. This API is not required for keep-alive connection. It is applicable only to short connection in LAN networking.

■ Calling direction

A third-party platform calls the API to a face recognition terminal or face recognition access control terminal.

■ Request description

- Request method: DELETE
- Request URL: /LAPI/V1.0/System/Event/Subscription/<ID>
- Content-Type: application/json

■ Request parameters

Param	Requirement	Type	Description	Example
-------	-------------	------	-------------	---------

<ID>	M	unsigned long	Subscription ID It is the subscription ID returned by a device when a subscriber subscribes to access control notifications.	0
------	---	---------------	---	---

■ Returned results

Example:

```

{
  "Response": {
    "ResponseURL": "/LAPI/V1.0/PACS/Controller/Event/Subscriptions/0",
    "CreatedID": -1,
    "ResponseCode": 0,
    "ResponseString": "Succeed",
    "StatusCode": 0,
    "StatusString": "Succeed",
    "Data": "null"
  }
}

```

4.4.4 Pushing Records

■ Function description

This API is used by an access control device to report personnel pass-through records.



NOTE!

1. When pushing a notification, a terminal establishes a socket connection with the subscriber by using the IP address and port, and then sends records. After sending, **the terminal closes the connection if short connection is used, and keeps keep-alive connection.**
2. If a terminal connects to and sends records to a third-party platform successfully, it does not care about the parsing result of the platform but directly deletes the recognition records.
3. Terminals do not care about the response of the server but pay attention to whether TCP data is sent successfully. If the transmission fails, it retransmits the data till the data is transmitted successfully.
4. **This API is used to report records, irrespective of keep-alive connection or short connection.**

■ Calling direction

A face recognition terminal or face recognition access control terminal calls the API to a third-party platform.

■ Request description

- Request method: POST
- Request URL: /LAPI/V1.0/System/Event/Notification/PersonVerification
- Content-Type: text/plain

■ Request parameters

Example:

```
{
  "Reference": "204.2.1.20:5118/LAPI/V1.0/System/Event/Subscription/0",
  "Seq": 5,
  "Timestamp": 1564735558,
  "NotificationType": 1,
  "FacelInfoNum": 1,
  "FacelInfoList": [
    {
      "ID": 5,
      "Timestamp": 1564707615,
      "CapSrc": 1,
      "FeatureNum": 0,
      "FeatureList": [
        {
          "FeatureVersion": "",
          "Feature": ""
        },
        {
          "FeatureVersion": "",
          "Feature": ""
        }
      ]
    },
    {
      "Temperature": 36.5,
      "MaskFlag": 1,
      "PanoImage": {
        "Name": "1564707615_1_86.jpg",
        "Size": 101780,
        "Data": "..."
      },
      "FaceImage": {
        "Name": "1564707615_2_86.jpg",
        "Size": 35528,
        "Data": "..."
      },
      "FaceArea": {
        "LeftTopX": 4981,
        "LeftTopY": 3744,
        "RightBottomX": 8250,
        "RightBottomY": 5583
      }
    }
  ]
}
```

```

    }
  ],
  "CardInfoNum": 0,
  "CardInfoList": [],
  "GateInfoNum": 0,
  "GateInfoList": [],
  "LibMatInfoNum": 1,
  "LibMatInfoList": [
    {
      "ID": 5,
      "LibID": 3,
      "LibType": 4,
      "MatchStatus": 2,
      "MatchPersonID": 0,
      "MatchFaceID": 0,
      "MatchPersonInfo": {
        "PersonName": "",
        "Gender": 0,
        "CardID": "",
        "IdentityNo": ""
      }
    }
  ]
}

```

Param	Requirement	Type	Description	Example
Reference	M	string	Subscriber description in the URL format	"192.168.0.13:80/Subscription/Subscribers/0"
Seq	M	unsigned long	Notification record number	1
Timestamp	M	unsigned long	Notification report time in the UTC format, in seconds	1510925018
NotificationType	M	unsigned long	Notification type 0: real-time notification 1: historical notification	1
FaceInfoNum	O	unsigned long	Number of face photo information entries. Range: [0, 1] When collected information does not contain face photos, FaceInfo-relevant fields can be omitted.	/
FaceInfoList	O	Array	Face photo information list. See <FaceInfoList> . When collected information does not contain face photos,	/

			FaceInfo-relevant fields can be omitted.	
CardInfoNum	O	unsigned long	Number of card information entries Range: [0, 1] When collected information does not contain cards or certificates, CardInfo-relevant fields can be omitted.	/
CardInfoList	O	Array	Card information list. See< CardInfoList >. When collected information does not contain cards or certificates, CardInfo-relevant fields can be omitted.	/
GateInfoNum	O	unsigned long	Number of gate machine information entries Range: [0, 1] When collected information does not contain gate machine information, GateInfo-relevant fields can be omitted.	/
GateInfoList	O	Array	Gate machine information list. See< GateInfoList >. When collected information does not contain gate machine information, GateInfo-relevant fields can be omitted.	/
LibMatInfoNum	O	unsigned long	Number of library match information entries Range: [0, 16] When the collection type is face collection, LibMatInfo-relevant fields can be omitted.	/
LibMatInfoList	O	Array	Library match information list. See< CtrlLibMatInfo >. When the collection type is face collection, LibMatInfo-relevant fields can be omitted.	/

(1) FaceInfoList:

Param	Requirement	Type	Description	Example
-------	-------------	------	-------------	---------

ID	M	unsigned long	Record ID	1
Timestamp	O	unsigned long	Collection time in the UTC format, in seconds If there is no collection time, this field can be omitted.	1510925018
CapSrc	M	unsigned long	Collection source 1: face information collected by a face recognition terminal 2: access control card information collected by a card reader 3: ID card information collected by a card reader 4: gate machine information collected by a gate machine For FaceInfo, set the value to 1 .	1
FeatureNum	O	unsigned long	Number of semi-structured features If there is no semi-structured feature, related fields can be omitted.	/
FeatureList	O	Array	Semi-structured feature list If there is no semi-structured feature, related fields can be omitted.	/
FeatureVersion	O	String	Version of the face semi-structured feature extraction algorithm, for example ISFRFR259.2.0 Length range: [0, 20]	/
Feature	O	String	Base64 encoding method The feature information extracted from faces is used to assist backend servers in comparing faces. Currently, the leftmost 512 bytes are encrypted.	/
Temperature	O	float	Body temperature If the body temperature is unknown or detection is not enabled, set the value to 0 .	
MaskFlag	O	unsigned long	Whether a person wears a gauze mask 0: unknown or detection disabled	

			1: no mask 2: with a mask	
PanoramaImage	O	Json Block	Face panorama. The field can be reported as required. For details, see FileInfo. Note: File size range on the PTS: [0, 1M]	/
FaceImage	O	Json Block	Face thumbnail. This field can be reported as required. For details, see FileInfo. Note: File size range on the PTS: [0 256K]	/
FaceArea	M	Json Block	Coordinates of the face area in a face panorama It is the face position in the face panorama. Coordinate normalization: 0-10000 Upper left and lower right points in the rectangle: "138,315,282,684"	/
LeftTopX	M	unsigned long	X coordinate of the upper left point	/
LeftTopY	M	unsigned long	Y coordinate of the upper left point	/
RightBottomX	M	unsigned long	X coordinate of the lower right point.	/
RightBottomY	M	unsigned long	Y coordinate of the lower right point.	/
FileInfo	C	Json Block	/	/
Name	C	String	File name Length range: [1, 16]	/
Size	C	unsigned long	Data size, with the unit of byte Range: [0, 1048576 (1MB)]	21783
Data	C	String	File data using the Base64 encoding format Base64 does not need to be added to the data header.	/

(2) CardInfoList

Param	Requirement	Type	Description	Example
ID	M	unsigned long	Record ID	1

Timestamp	O	unsigned long	Collection time in the UTC format, in seconds If there is no collection time, this field can be omitted.	1510925018
CapSrc	M	unsigned long	Collection source 1: face information collected by a face recognition terminal 2: access control card information collected by a card reader 3: ID card information collected by a card reader 4: gate machine information collected by a gate machine For CardInfo, set the value to 2 or 3 .	/
CardType	M	unsigned long	0: ID card; 1: access control card	0
CardID	C	String	Physical access control card ID. The maximum length is 18 characters.	123456789
CardStatus	C	unsigned long	Status of the access control card: 1: valid; 0: invalid	1
Name	C	String	Name on the ID card. Length range: [1, 63]	/
Gender	C	unsigned long	Gender on the ID card. 0: unknown; 1: male; 2: female; 9: unspecified	1
Ethnicity	C	unsigned long	Nationality on the ID card. See the Romanization and corresponding code of GB/T 3304 <i>Names of Nationalities of China in Romanization with Codes</i> . 01: Han nationality	01
Birthday	C	String	Birthday on the ID card. Format: YYYYMMDD	20111230
ResidentialAddress	C	String	Residential address on the ID card	/
IdentityNo	C	String	ID card number. The maximum length is 18 characters.	/
IssuingAuthority	C	String	Issuing department on the ID card. Format: Public Security Bureau of XX District (County),	/


			XX City, XX Province	
IssuingDate	C	String	Date of issuance on the ID card. Format: YYYYMMDD	/
ValidDateStart	C	String	Start date of the validity period of the ID card. Format: YYYYMMDD	/
ValidDateEnd	C	String	End date of the validity period of the ID card. Format: YYYYMMDD	/
IDImage	C	FileInfo	Photo on the ID card Note: File size range on the PTS: [0, 32K]	/

(3) GateInfoList

Param	Requirement	Type	Description	Example
ID	M	unsigned long	Record ID	1
Timestamp	M	unsigned long	Collection time in the UTC format, in seconds If there is no collection time, this field can be omitted.	1510925018
CapSrc	M	unsigned long	Collection source 1: face information collected by a face recognition terminal 2: access control card information collected by a card reader 3: ID card information collected by a card reader 4: gate machine information collected by a gate machine For GateInfo, set the value to 4.	/
InPersonCnt	M	unsigned long	Number of people going inside	/
OutPersonCnt	M	unsigned long	Number of people going outside	/

(4) CtrlLibMatInfoList

Param	Requirement	Type	Description	Example
ID	M	unsigned long	Record ID	1
LibID	M	unsigned long	Library ID	3

LibType	O	unsigned long	Library type. It is optional. 0: invalid value by default 1: blacklist 2: graylist/stranger 3: employee 4: visitor	3
MatchStatus	M	unsigned long	Match status 1: verification succeeded 2: verification failed (comparison failed) 3: verification failed (comparison succeeded, but not within the arming time) 10: verification failed (comparison succeeded, but face attribute abnormal) 41: registered picture collection succeeded 42: registered picture collection failed  NOTE! When MatchStatus is 10 , the Temperature and MaskFlag fields in FaceInfoList need to be obtained to get detailed abnormality information.	1
MatchPersonID	M	unsigned long	Matched person ID	1020
MatchFaceID	M	unsigned long	Matched face photo ID	10201
MatchPersonInfo	O	Json Block	Matched person information. It is optional.	/
PersonName	O	String	Member name. Length range: [1, 63]	/
Gender	O	unsigned long	Member gender 0: unknown; 1: male 2: female; 9: unspecified	1
CardID	O	String	Access control card No. It is left blank if there is no access control card.	123456789
IdentityNo	O	String	ID card No. It is left blank if there is no ID card.	/

4.5 Visual Intercom

4.5.1 Querying the Current Location of a Face Recognition Access Control Terminal

■ Function description

This API is used to query the current location of a face recognition access control terminal.

■ Calling direction

A third-party platform or indoor monitor calls the API to a face recognition access control terminal.

■ Request description

- Request method: GET
- Request URL: /LAPI/V1.0/EntranceGuard/LocationInfo
- Content-Type: application/json

■ Returned results

Example:

```
{
  "Response": {
    "ResponseURL": "/LAPI/V1.0/EntranceGuard/LocationInfo",
    "CreatedID": -1,
    "ResponseCode": 0,
    "ResponseString": "Succeed",
    "StatusCode": 0,
    "StatusString": "Succeed",
    "Data": {
      "PropertyIPAdd": "0.0.0.0",
      "Community": "",
      "Building": "1",
      "UnitToNum": 1,
      "UnitInfo": [
        {
          "Unit": 1
        }
      ]
    }
  }
}
```

■ Returned parameters

Param	Requirement	Type	Description	Example
PropertyIPAdd	M	String	Management center	0.0.0.0

			IP address Length range: [0, 16]	
Community	M	String	Community information Length range: [0, 64]	Jinse Qiantang
Building	M	String	Building information Length range: [0, 64]	1
UnitToINum	M	unsigned long	Number of units Range: [0, 8]	1
UnitInfo	M		Unit information	
Unit	M	unsigned long	Unit No.	1

4.5.2 Obtaining Binding Information About an Indoor Monitor

■ Function description

This API is used to obtain/set the unit room binding information and delete binding information about the entire unit or a room in the unit.



NOTE!

1. The **Name** parameter specifies the unit room for which binding information needs to be obtained/set.
2. The **Index** parameter is used only in the GET operation to specify the start room No.

■ Calling direction

A third-party platform or indoor monitor calls the API to a face recognition access control terminal.

■ Request description

- Request method: GET
- Request URL: /LAPI/V1.0/EntranceGuard/RoomInfo?Name=<Unit>&Index=<Room No.>
- Content-Type: application/json

■ Request example

- Request URL: /LAPI/V1.0/EntranceGuard/RoomInfo?Name=1_2&Index=1

■ Request parameters

Param	Requirement	Type	Description	Example
<Unit>	M	String	Unit name in the format of building_unit	1_2

<Room No.>	M	unsigned long	Start room No.	1
------------	---	---------------	----------------	---

■ Returned results

```

{
  "Response": {
    "ResponseURL": "/LAPI/V1.0/EntranceGuard/RoomInfo",
    "CreatedID": -1,
    "ResponseCode": 0,
    "ResponseString": "Succeed",
    "StatusCode": 0,
    "StatusString": "Succeed",
    "Data": {
      "TolNum": 1,
      "CurrentNum": 1,
      "PathInfoList": [
        {
          "Room": "3",
          "IpAdd": "192.174.12.130",
          "Code": ""
        }
      ]
    }
  }
}

```

■ Returned parameters

Param	Requirement	Type	Description	Example
TolNum	M	unsigned long	Total quantity (assigned only in the GET operation)	1
CurrentNum	M	unsigned long	Number of rooms of this operation (32 at most each time) In the GET operation, the Index value is determined based on the last Index value plus the last CurrentNum value.	1
PathInfoLis	M		Path or file information	
Room	M	String	Room No. Length range: [0, 32]	1201
IpAdd	M	String	IP address. Length range: [0, 15]	0.0.0.0

Code	M	String	Password. Length range: [6]	123456
------	---	--------	-----------------------------	--------

4.5.3 Setting Binding Information for an Indoor Monitor

■ Function description

This API is used to set binding information for a unit room.



NOTE!

The **Name** parameter specifies the unit room for which binding information needs to be set.

■ Calling direction

A third-party platform or indoor monitor calls the API to a face recognition access control terminal.

■ Request description

- Request method: PUT
- Request URL: /LAPI/V1.0/EntranceGuard/RoomInfo?Name=<Unit>
- Content-Type: application/json

■ Request example

- Request URL: /LAPI/V1.0/EntranceGuard/RoomInfo?Name=1_2

Example:

```
{
  "TotNum": 1,
  "CurrentNum": 1,
  "PathInfoList": [
    {
      "Room": "3",
      "IpAdd": "192.174.12.130",
      "Code": "123458"
    }
  ]
}
```

■ Request parameters

Param	Requirement	Type	Description	Example
<Unit>	M	String	Unit name in the format of building_unit	1_2

Param	Requirement	Type	Description	Example
TolNum	M	unsigned long	Total quantity (assigned only in the GET operation)	1
CurrentNum	M	unsigned long	Number of rooms of this operation (32 at most each time) In the GET operation, the Index value is determined based on the last Index value plus the last CurrentNum value.	1
PathInfoList	M		Path or file information	
Room	M	String	Room No. Length range: [0, 32]	1201
IpAdd	M	String	IP address. Length range: [0, 15]	0.0.0.0
Code	M	String	Password. Length range: [6]	123456

■ Returned results

```
{
  "Response": {
    "ResponseURL": "/LAPI/V1.0/EntranceGuard/RoomInfo",
    "CreatedID": -1,
    "ResponseCode": 0,
    "ResponseString": "Succeed",
    "StatusCode": 0,
    "StatusString": "Succeed",
    "Data": "null"
  }
}
```

4.5.4 Deleting Binding Information About an Indoor Monitor

■ Function description

This interface is used to delete binding information about a unit room.



NOTE!

If the value of the **Name** parameter is only a unit name, binding information about the entire unit will

be deleted. If the value ends with "/room No.", the binding information about the specified unit room will be deleted. (The length range of a unit name is 0–64, while that of a room No. is 0–32.)

■ Calling direction

A third-party platform or indoor monitor calls the API to a face recognition access control terminal.

■ Request description

- Request method: DELETE
- Request URL: /LAPI/V1.0/EntranceGuard/RoomInfo?Name=<Unit>
- Content-Type: application/json

■ Request example

- Request URL: /LAPI/V1.0/EntranceGuard/RoomInfo?Name=1_2

■ Request parameters

Param	Requirement	Type	Description	Example
<Unit>	M	String	Unit name in the format of building_unit Note: To delete the binding information about a unit room, set the value in the format of building_unit/room No.	1_2/101

■ Returned results

```
{
  "Response": {
    "ResponseURL": "/LAPI/V1.0/EntranceGuard/RoomInfo",
    "CreatedID": -1,
    "ResponseCode": 0,
    "ResponseString": "Succeed",
    "StatusCode": 0,
    "StatusString": "Succeed",
    "Data": "null"
  }
}
```

4.5.5 Delivering the Call Status

■ Function description

This API is used to deliver the call status.

■ Calling direction

A third-party platform or indoor monitor calls the API to a face recognition access control terminal.

■ Request description

- Request method: PUT
- Request URL: /LAPI/V1.0/EntranceGuard/CallState
- Content-Type: application/json

■ Request parameters

Example:

```
{
  "IsCalling": 0
}
```

Param	Requirement	Type	Description	Example
IsCalling	M	unsigned long	<p>Call status</p> <p>0: The indoor monitor hangs up. (When a face recognition access control terminal makes a call request to an indoor monitor, and the indoor monitor hangs up, the indoor monitor delivers this state to the face recognition access control terminal.)</p> <p>1: The indoor monitor delivers a call response. (When a face recognition access control terminal makes a call request to an indoor monitor, and the indoor monitor answers the call, the indoor monitor delivers this state to the face recognition access control terminal.)</p> <p>2: The indoor monitor is busy. (When a face recognition access control terminal makes a call request to an indoor monitor, and the indoor monitor is communicating with another outdoor monitor, the indoor monitor delivers this state to the face recognition access control terminal.)</p> <p>3: The indoor monitor rejects the call. (When</p>	0

			a face recognition access control terminal makes a call request to an indoor monitor, and automatic answer is configured on the indoor monitor, the indoor monitor delivers this state to the face recognition access control terminal.)	
--	--	--	--	--

■ Returned results

Example:

```
{
  "Response": {
    "ResponseURL": "/LAPI/V1.0/EntranceGuard/CallState",
    "CreatedID": -1,
    "ResponseCode": 0,
    "ResponseString": "Succeed",
    "StatusCode": 0,
    "StatusString": "Succeed",
    "Data": "null"
  }
}
```

4.5.6 Reporting the Call Status

■ Function description

This API is used to push the call status.

■ Calling direction

A face recognition access control terminal calls the API to a third-party platform or indoor monitor.

■ Request description

- Request method: GET
- Request URL: /LAPI/V1.0/EntranceGuard/CallState?RequestCall=1
- Content-Type: application/json

■ Request parameters

Example: RequestCall=1: A call is reported.

RequestCall=0: Hanging up is reported.

4.6 Device Control

4.6.1 Opening the Door Remotely

■ Function description

This API is used by a face recognition terminal or face recognition access control terminal to control the door opening.

■ Calling direction

A third-party platform/third-party indoor monitor calls the API to a face recognition terminal or face recognition access control terminal.

■ Request description

- Request method: PUT
- Request URL: /LAPI/V1.0/PACS/Controller/RemoteOpened
- Content-Type: application/json

■ Request example

- Request URL: /LAPI/V1.0/PACS/Controller/RemoteOpened

■ Returned results

```
{
  "Response": {
    "ResponseURL": "/LAPI/V1.0/PACS/Controller/RemoteOpened",
    "CreatedID": -1,
    "ResponseCode": 0,
    "ResponseString": "Succeed",
    "StatusCode": 0,
    "StatusString": "Succeed",
    "Data": "null"
  }
}
```

5 Appendices

5.1 Error Code Description

(1) System processing error codes --- "ResponseCode"

Error Code	Description
------------	-------------

0: Succeed	Succeeded
1: Common Error	Common error
2: Invalid Arguments	Invalid parameter
3: Not Authorized	User not authorized
4: Not Supported	Not supported by the device
5: User status exception	Abnormal user status

(2) Service processing error codes --- "StatusCode"

Error Code	Description
0: Succeed	Execution succeeded
1: Common Fail	Execution failed
2: Invalid Param	Invalid input parameter
3: No Memory	Insufficient system memory
8: UnSupport	Function unsupported
10: SRLZ Fail	Serialization failed
11: Unauthorized	Basic authentication failed
12: Unauthorized	Digest authentication failed
13: NO Space	Subscription full
14: Already Exit	Repeated subscription

(3) HTTP communication protocol error codes

Error Code	Description
• 2xx success	
200	Normal; the request is completed.
201	The resource is correctly created.
202	Normal; the message is processed but the processing is not completed.
203	Normal; some information — only a part of information is returned.
204	Normal; no response — the request is received but no message needs to be returned.
• 3xx redirection	
301	Moved — the requested data has a new location and the change is permanent.
302	Found — the requested data has a different URL temporarily.
303	See others — the response to the request can be found in another URL and the GET method shall be used to search for the response.
304	Unmodified — the document is not modified as expected.

305	Use proxy — the proxy provided in the location field must be used to access the requested resource.
306	Unused — the code is no longer used and reserved for future use.
• 4xx client error	
400	Error request — the request has syntax errors or the request cannot be met.
401	Unauthorized — an unauthorized client accesses data.
402	Payment required — the billing system is available.
403	Prohibited — access is denied even if authorization is obtained.
404	Not found — the server does not find the specified resource; the document does not exist.
405	Method Not Allow — the requested resource does not support the request method.
406	The client browser does not accept the requested page of the MIME type.
407	Proxy authentication requests — the client must first authenticate itself using the proxy.
410	The requested Web page does not exist (permanently).
415	Media type not supported — the server rejects the service request because it does not support the format of the request entity.
• 5xx server errors	
500	Internal error — the server cannot complete the request due to an unexpected situation.
501	Not executed — the server cannot identify or does not support the request method.
502	Wrong gateway — the server receives an invalid response from the upstream server.
503	Service unavailable — the server is unable to process the request due to temporary overload or maintenance.

5.2 Debugging Tool

See the *Debugging Tool Operation Guide V1.01*.



Debugging Tool
Operation Guide