MQTT Protocol For Face Recognition All-in-One Machine
V2.0.3

1、Documentation

1.1 Intended Readers

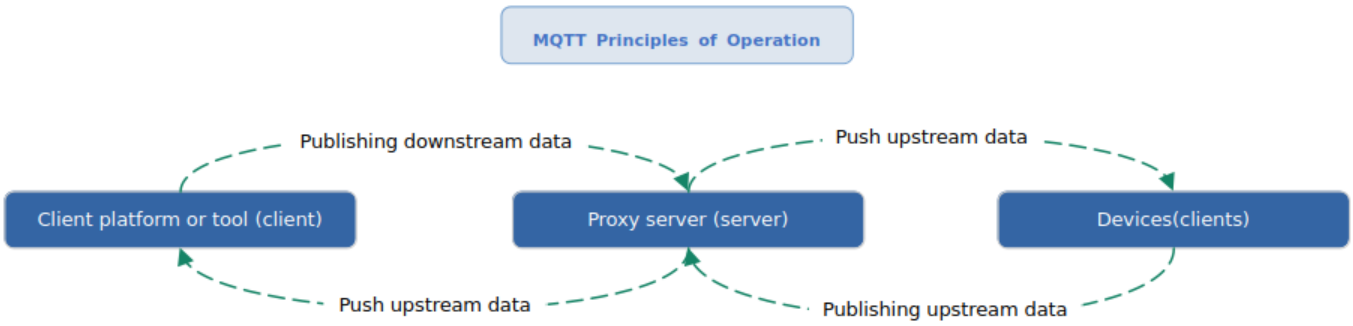
This development document is intended to be read by development, maintenance or management personnel who have certain development capabilities, understand PHP, JAVA, .NET and other development languages, and have a certain understanding of MQTT.

1.2 Introduction to MQTT

MQTT（Message Queuing Telemetry Transport） is an instant messaging protocol developed by IBM(International Business Machines Corporation).

It is a “lightweight” communications protocol based on a publish/subscribe model, built on the TCP/IP protocol and designed for constrained devices and low bandwidth, high latency or unreliable networks.

The way MQTT is implemented and how it works is shown below:



The MQTT protocol requires: clients and a server. There are three identities in the MQTT protocol: Publisher, Broker, Server and Subscriber. In this case, the message publisher and subscriber are clients, the message broker is a server, and the message publisher can be a subscriber at the same time.

MQTT transport messages are divided into: Topic and payload two parts Topic, can be understood as the type of message, subscribers subscribe , you will receive the topic of the message content payload, can be understood as the content of the message, refers to the subscriber to use the content of the specific.

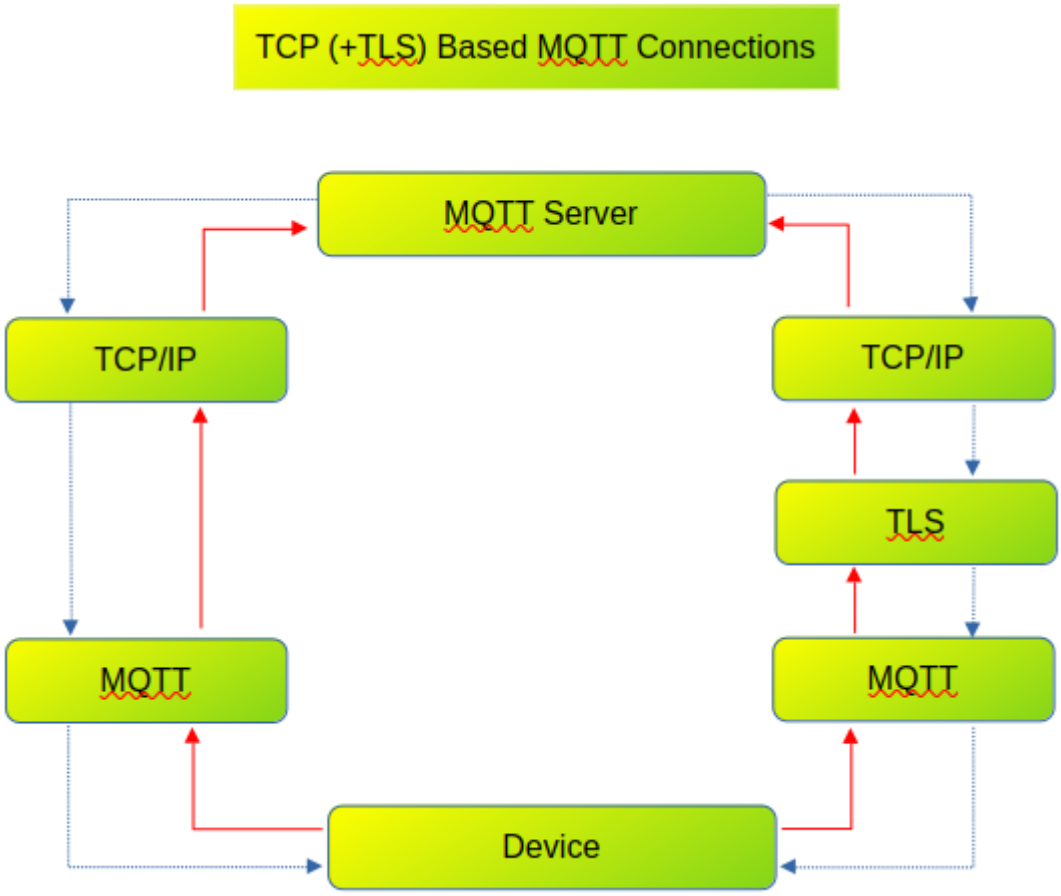
Downlink data includes (but is not limited to): data issued by the platform (data received by the equipment), personnel management, equipment management, etc...

Uplink data includes (but is not limited to)： data received by the platform (data reported by the device), record reporting, device status, return of downlink data execution results, etc.

1.3 MQTT-TCP Connection Communication

This article focuses on TCP-based MQTT connection, the connection method is MQTT client direct connection .

For secure communication, a TLS-encrypted connection can be used. When using a TLS connection, ssl:// is appended to the cloud address by default, as described in 3.2 Setting MQTT Parameters.



TLS (Transport Layer Security protocol) is used to provide confidentiality and data integrity between two applications. In short, TLS creates a channel on top of TCP that encrypts the plaintext of TCP transmissions, thus keeping the communication private.

1.4 MQTT Version Information

The MQTT version adopted by the device is the synchronization mode version of V3.1.1. QOS=1, and it adopts the testament function (used for offline notification), and the default bottom heartbeat interval is 30s, which can be set in the MQTT page of the device's web terminal.

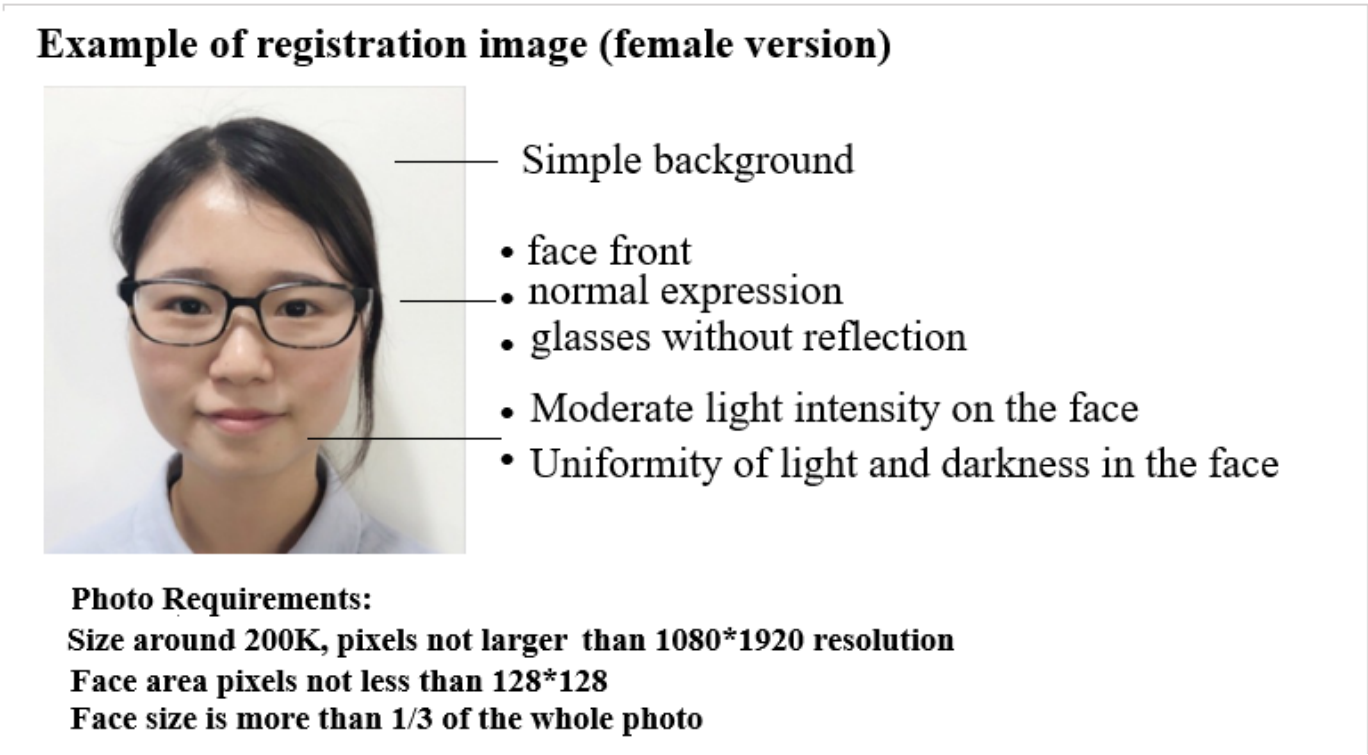
1.5 Limitations on Use

Generic interfaces are supported for different models of Face Machine, while special function interfaces are only supported for corresponding special models of device. You need to pay attention to the following usage restrictions during the process of interfacing with the Face Machine protocol.

1.5.1 Registered Image Requirements

Requirements to be followed for face registration images include (but are not limited to) :

1. Registered images are clear
2. Size in the range of 50K to 200K
3. The resolution is limited to 1080P (1080*1920).
4. Additional notes are available in the “Registered Photo Specifications” document, which can be provided by sales or technical support.



1.5.2 Data Length

1.MQTT Message Length
The maximum length of a single packet message **Downlink data** is limited to**1M**

1.5.3 MQTT Username and Password

1. MQTT Username
   MQTT Username(Cloud Username): Support UTF-8 encoded characters, length limit 64 bytes.
   The following special characters **cannot be supported** (but are not limited to) :
   / # & +

2. MQTT Password
   MQTT Password(Cloud Password):Support UTF-8 encoded characters, length limit 128 bytes.
   The following special characters **cannot be supported** (but are not limited to) :
   / # & +

1.5.4 Coding Requirements

All Chinese characters should be encoded in UTF-8.

2、Windows Debugging

In order to quickly get started and understand the working mechanism of MQTT, in this chapter explains how to quickly set up MQTT intermediary proxy server on the Windows side, as well as introducing MQTT client simulation tool mqtt.box or mqtt.fx (mqtt.box for large data sending and subscription will have lag, mqtt.fx is only suitable for subscription to receive the topic of the message, < font color="red">not suitable for simulation of sending data with Chinese characters) Formal online platform docking without building the middle proxy server and client tools and so on. not suitable for simulation of sending data with Chinese characters) .Formal online platform docking, no need to build the windows side of the intermediary proxy server and the client tools and so on. The following intermediate proxy server construction and MQTT client tools to use, the formal use of the platform needs to integrate the development or the use of third-party servers.

2.1 MQTT Server setup

MQTT server to build **non-mandatory steps**, just to achieve their own local debugging before building a temporary server, this document introduces the server **for testing purposes only, does not guarantee the stability and reliability of data transmission**.

1. According to their own computers and systems refer to the online steps to download and install jdk, as long as the JDK8 can be (the following is a test to install the jdk version)

```
C:\Users\Cff>java -version
java version "1.8.0_221"
Java(TM) SE Runtime Environment (build 1.8.0_221-b11)
Java HotSpot(TM) 64-Bit Server VM (build 25.221-b11, mixed mode)

C:\Users\Cff>
```

2. Download apache-apollo-1.7.1-windows version, here is a link to it.
   http://archive.apache.org/dist/activemq/activemq-apollo/1.7.1/
3. Open `cmd` as administrator and go to the `bin` directory in the unzipped file.
4. Execute the command: create a directory (the following path is an example)

```
apollo create myapollo C:\apache-apollo\broker
```

1. After successful creation, go to the `broker\bin` directory and execute the `apollo-broker.cmd run` command to start the service.

```
D:\apache-apollo-1.7.1\bin\mybroker\bin>
D:\apache-apollo-1.7.1\bin\mybroker\bin>
D:\apache-apollo-1.7.1\bin\mybroker\bin>apollo-broker.cmd run
[0;40;37m
    [1m_____      .__  .__
   /  [32m_  \ [37m____   ____ |  [32m| [37m|  [32m|   [37m____
  /  [32m/[37m_\  [32m\[37m\[32m____ \ [37m/  [32m_ \[37m|  [32m| [37m|  [32m| [37m/  [32m_ \
 [37m/    |    [32m\  |[37m_> [32m>  <[37m_> [32m)  |[37m_|  [32m| [37m_(  [32m<[37m_> [32m)
 \____[37m|[32m__  /   __/ \____/[37m|[32m____/____/[37m\[32m____/
[9C\/[37m|[32m__|   [37mApache Apollo[2m (1.7.1)
[0m

Loading configuration file 'D:\apache-apollo-1.7.1\bin\mybroker\etc\apollo.xml'.
INFO | OS     : Windows 10 10.0
INFO | JVM    : Java HotSpot(TM) 64-Bit Server VM 1.8.0_221 (Oracle Corporation)
INFO | Apollo : 1.7.1 (at: D:\apache-apollo-1.7.1)
INFO | Starting store: leveldb store at D:\apache-apollo-1.7.1\bin\mybroker\data
INFO | virtual host startup is waiting on store startup
INFO | virtual host startup is no longer waiting.  It waited a total of 1 seconds.
INFO | Accepting connections at: tcp://0.0.0.0:61613
INFO | Accepting connections at: tls://0.0.0.0:61614
INFO | Accepting connections at: ws://0.0.0.0:61623/
INFO | Accepting connections at: wss://0.0.0.0:61624/
INFO | Administration interface available at: https://127.0.0.1:61681/
INFO | Administration interface available at: http://127.0.0.1:61680/
```

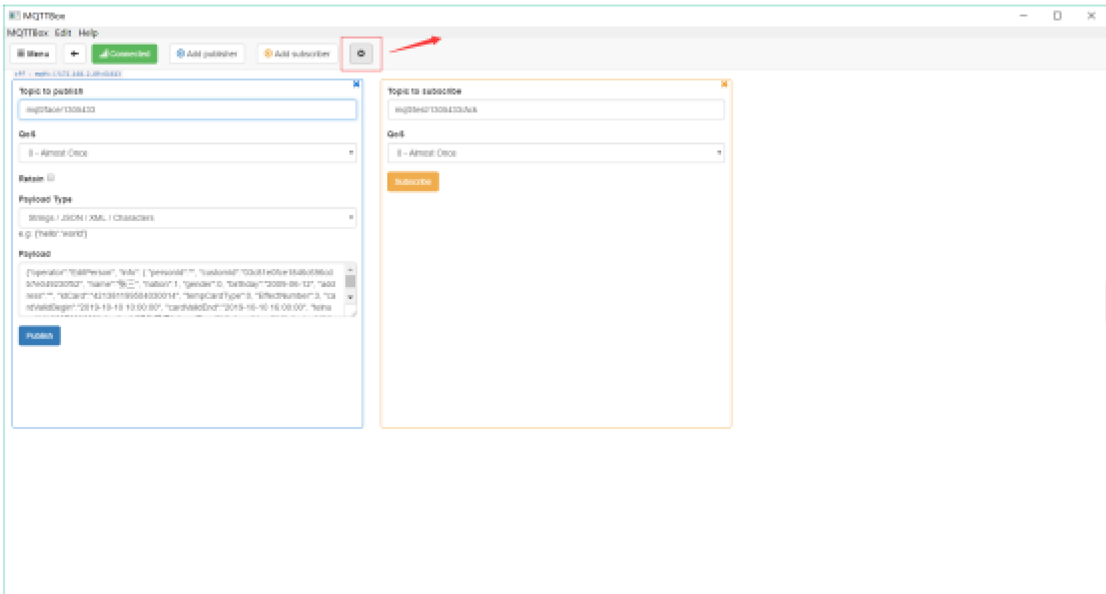
2. Log in to http://127.0.0.1:61680, the default account is `admin`,the password is `password`, note that the port of the web page is 61680, but the port of the MQTT service is 61613.

2.2 Introduction to MQTT Debugging Tools

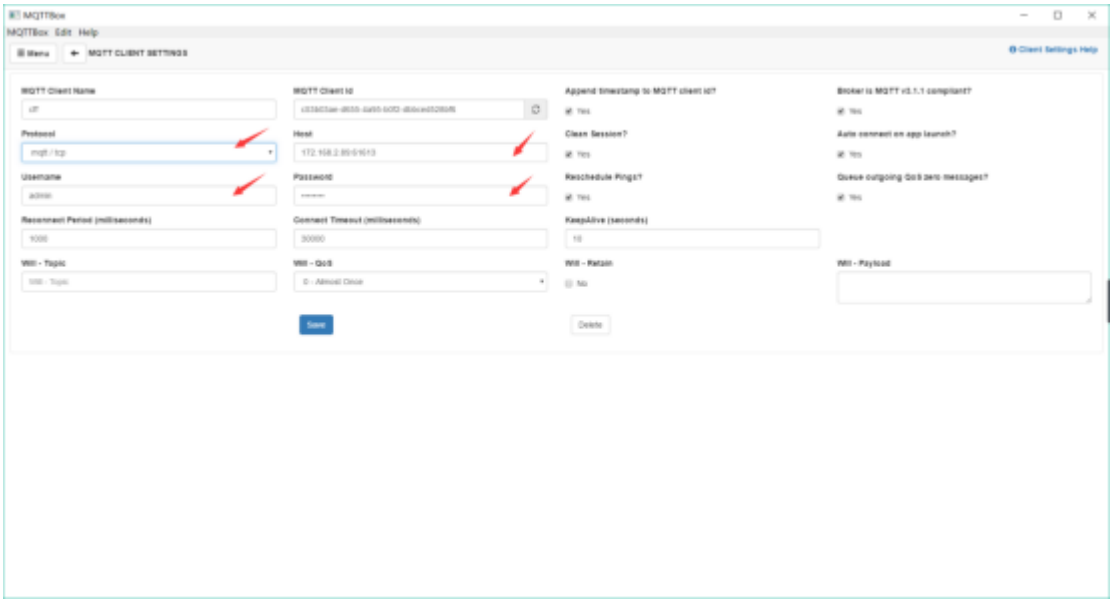
Recommended to use mqttbox MQTTBox , you can also use mqttfx (This tool under the Chinese will be messy, does not support Chinese) , Note: This tool is mainly for debugging purposes.

2.2.1 MQTT Connection Steps

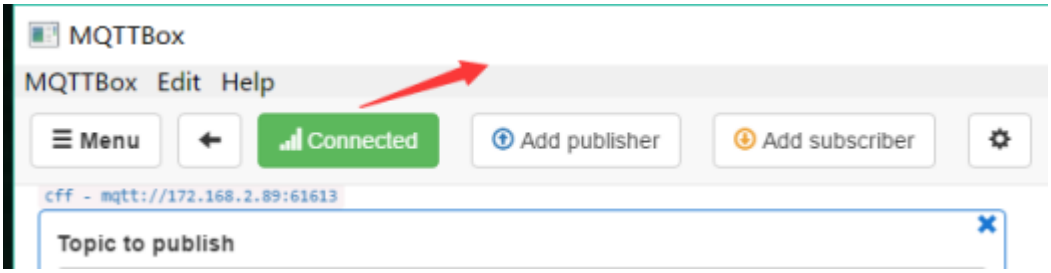
Confirm that the system firewall has been turned off, first add a new connection and open the connection settings.



Mainly fill in the mqtt server host address and port (port default 61613) account and password as shown below:



A green color means the connection is successful, as shown below:



3、Message Communication Topic

3.1 What is Topic

Topic is a UTF-8 string that acts as a transport intermediary for publish /subscribe messages. You can publish or subscribe messages to a Topic.

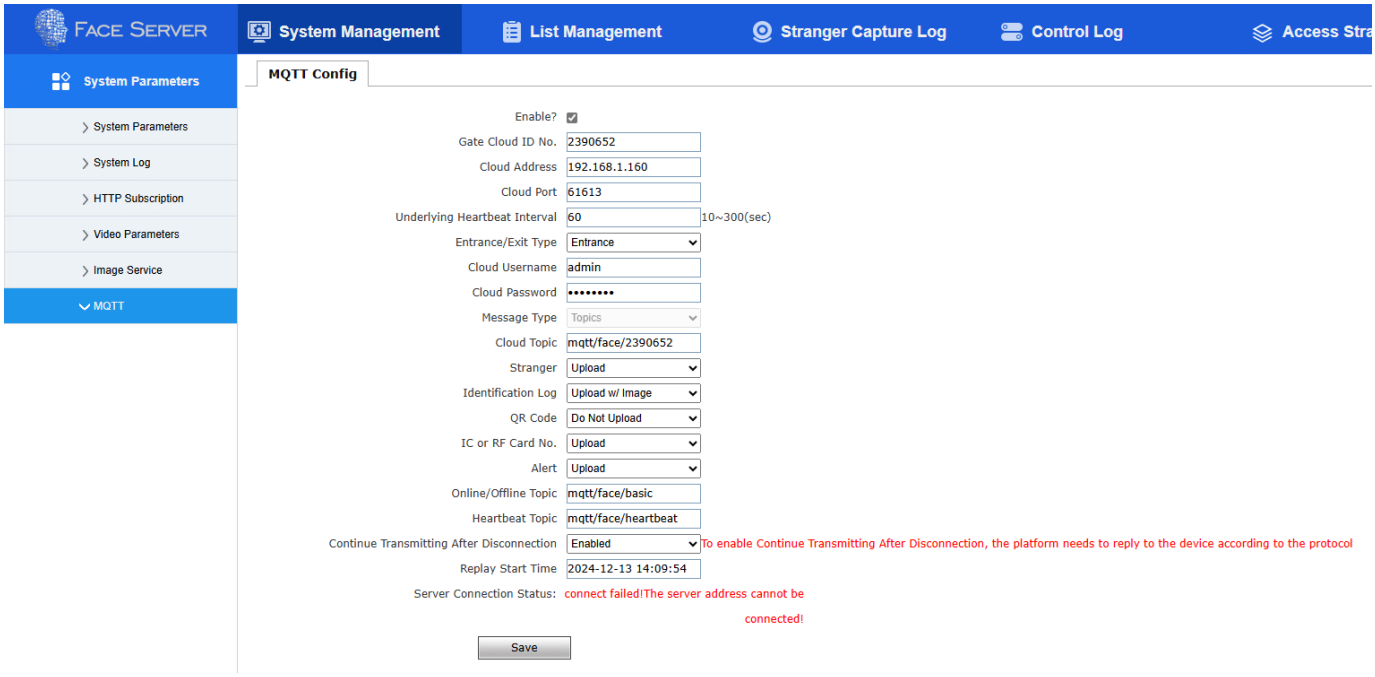
Device-related Topics can be divided into two categories, device-subscribed Topics (corresponding to topics where the platform or tool side publishes downlink data to the device), and device-published Topics (corresponding to topics where the platform or tool side receives device uplink data Topics).

Downlink data includes (but is not limited to): data issued by the platform (data received by the equipment), personnel management, equipment management, etc...

Uplink data includes (but is not limited to)： data received by the platform (data reported by the device), record reporting, device status, return of downlink data execution results, etc.

3.2 Setting MQTT Parameters

The MQTT parameter settings for the device are shown below and include the following settings (but are not limited to):



MQTT Setting parameters description

Items	Description	Examples/Selections
Gate Cloud ID	MQTT clientid (device ID/SN is used by default to ensure uniqueness)	1306612
Cloud address	The address of the MQTT server (domain name address is supported, the DNS of the device should be set correctly, if TLS encryption is used then fill in ssl://172.168.2.90)	ssl://172.168.2.90
Cloud port	MQTT server port (tcp default：61613 ; tls default ：61614)	61613
Type of entrance/exit	MQTT identifies the direction of entry and exit of the device using	Entrance/exit/no direction
Cloud Username	Connecting to the MQTT server username	admin
Cloud Password	Connecting to the MQTT server password	****
Cloud Topic	Message topics subscribed to on the device side (topics for which the platform or tool sends downstream data)	mqtt/face/ID(where ID refers to the device ID/SN，e.g,:1306612)
Strangers	Stranger capture logs upload settings for devices	Upload/No Upload
Control logs	Control logs upload settings for devices	No upload/upload with image/upload without image
QR code	Device transmitting QR code data (devices with QR code function) settings	Upload/No Upload
identity cards	ID card data subscription settings (ID card capable devices)	Upload/No Upload
IC or RF card number	IC or RF card number subscription setting (Device with card swipe function)	Upload/No Upload
Alarm	Device Related Alarms Subscription	Upload/No Upload
Up and down Topic	Device up and down the line will post topics	mqtt/face/basic
Heartbeat Topic	Application layer heartbeat	mqtt/face/heartbeat

Items	Description	Examples/Selections
Continue Transmitting After Disconnection	Whether or not the uploading of the device's stranger capture logs and control logs is enabled for `Continue Transmitting After Disconnection` mode (**A reply is required to enable the** `Continue Transmitting After Disconnection` **mode, see below**)	Disable/Enable
Replay Start Time	Any data after the point in time after the `Continue Transmitting After Disconnection` enable is retransmitted.	2023-01-01 00:00:00

3.3 The Device Subscribes to the Message of the Topic

Since devices connecting to the server need a unique clientid to distinguish between different devices, device ID/SN is used by default to ensure uniqueness. Topic of the device subscription message: mqtt/face/ID (where ID refers to the device ID/SN, e.g., 1306612), this topic is the topic on which the device receives all downstream data sent down by the client platform or tool.

3.4 The Device Publishes the Message of the Topic

The Topics for which the device publishes messages refers to the Topics for which the device publishes **uplink data** messages.This section of topics is divided into fixed and unfixed Topics.

3.4.1 Fixed Topic

The following two topics in the software old version (before 20210803) are fixed on the device side and cannot be set, while the new software version can be set and modified.

1. Application-layer heartbeat topic：
   mqtt/face/heartbeat
2. Topic for up and down notifications：
   mqtt/face/basic

3.4.2 Unfixed Topic

This part mainly focuses on the set cloud topic: mqtt/face/ID (where ID refers to the device ID/SN, e.g., 1306612), and according to different data types, appending suffixes to the cloud topic Topic constitutes the Topic for the device to publish different messages, including the following (but not limited to):

1. Stranger Capture Logs Topic：
   mqtt/face/ID/Snap (where ID refers to the device ID/SN, e.g., 1306612)
2. Authentication (Control) Logs Topic：
   mqtt/face/ID/Rec (where ID refers to the device ID/SN, e.g., 1306612)
3. QRCode Information Topic：
   mqtt/face/ID/QRCode (where ID refers to the device ID/SN, e.g., 1306612,The device needs to have the corresponding function)

4. ID Information Topic：
   mqtt/face/ID/IDCard（where ID refers to the device ID/SN, e.g., 1306612,The device needs to have the corresponding function）
5. IC or RF Card Number Information Topic：
   mqtt/face/ID/Card（where ID refers to the device ID/SN, e.g., 1306612,The device needs to have the corresponding function）
6. Door Magnetic or Alarm Message Topic：
   mqtt/face/ID/Alarm（where ID refers to the device ID/SN, e.g., 1306612,The device needs to have the corresponding function）
7. Downstream data execution results Topic：
   mqtt/face/ID/Ack（where ID refers to the device ID/SN, e.g., 1306612）

4、Device Status

4.1 Application Layer Heartbeat

1.Description

Application layer heartbeats are distinguished from the underlying heartbeats that connect the device to the server, and are a special message type added by the device to assist in determining the device's online status; the device's application layer heartbeats do not need to be replied to. The interval heartbeat time for the device is roughly 60 seconds, and this value can be set on the web-side page.

2.API Description

Items	**Description**
Interface name	Application layer heartbeat
Data flow (active command side)	Device->Platform
Uplink data topics	mqtt/face/heartbeat
Downlink data topic	NULL

3.Description of reported fields
Parameter information(Note:`Optional`is optional):

Key	**Type**	**Valuse**	**Description**
operator		HeartBeat	Device heartbeat
info	Object		Concrete content
facesluiceId	String		The clientID for connecting to the server,device ID/SN is used by default to ensure uniqueness
time	String	2023-01-01 00:00:00	Heartbeat time YYYY-MM-DD hh:mm:ss

4.Reporting Example

```
{
    "operator": "HeartBeat",
    "info": {
        "facesluiceId": "1306612",
        "time": "2023-01-01 00:00:00"
    }
}
```

4.2 Device On-Line and Off-Line Notifications

Devices going online and offline will be notified to the corresponding customer platforms via this topic. Offline notifications include offline caused by canceling MQTT services or underlying IO anomalies, and so on.

4.2.1 On-Line Notification

1.Description

Device on-line notification，on-line notification needs to be replied to by the platform, reply error or do not reply to the on-line notification, will be sent again in the interval on-line notification.

2.API Description

Items	Description
Interface Name	On-Line notification
Data flow (active command side)	Device->Platform
Uplink data topic	mqtt/face/basic
Downlink data topic	mqtt/face/ID（Where ID refers to the device ID/SN，e.g,:1306612）

3.Description of reported fields

Parameter information(Note:`Optional`is optional):

Key	Type	Values	Description
operator		Online	On-Line notification
info	Object		Concrete content
facesluiceId	String		The clientID for connecting to the server,device ID/SN is used by default to ensure uniqueness
usename	String		Cloud Username
time	String	2023-01-01 00:00:00	Heartbeat time，YYYY-MM-DD hh:mm:ss
ip	String		Device ip address
facesname	String		Device name

4.Reporting Example

```
{
    "operator": "Online",
    "info": {
        "facesluiceId": "1306612",
        "username": "admin",
        "time": "2020-05-12 15:11:10",
        "ip": "172.168.2.202",
        "facesname": "Enter"
    }
}
```

5.Explanation of the Paragraph of the Reply Message
Parameter information(Note:`Optional`is optional):

key	Type	Values	Description
operator		Online-Ack	Platform replies to receive on-line notifications.
info	Object		Concrete content
facesluiceId	String		The clientID for connecting to the server,device ID/SN is used by default to ensure uniqueness.
result	String	"ok"	Received on-line notification status successfully.

6.Example of Reply Message

```
{
    "operator": "Online-Ack",
    "info": {
        "facesluiceId": "1305433",
        "result": "ok"
    }
}
```

4.2.2 Off-Line Notification

1.Description
Canceling the device connection to the MQTT service or an error in the underlying IO, etc., the device generates offline notification.
2.API Description

Items	Description
Interface Name	Off-Line notification
Data flow (active command side)	Device->Platform
Uplink data topic	mqtt/face/basic
Downlink data topic	NULL

3.Description of reported fields
Parameter information(Note: `Optional` is optional):

Key	Type	Values	Description
operator		Offline	Off-Line notification
info	Object		Concrete content
facesluiceId	String		The clientID for connecting to the server,device ID/SN is used by default to ensure uniqueness
time	String	2023-01-01 00:00:00	Offline notification time，YYYY-MM-DD hh:mm:ss，the will message does not have time field.

4.Reporting Example

```
{
    "operator": "Offline",
    "info": {
        "facesluiceId": "1305433",
        "time": "2023-01-01 00:00:00"
    }
}
```

4.3 Access Control Alarm Messages

1.Description

This interface is only suitable for special models (door access control series) to use, the gate head series of models do not apply to this interface, dismantling alarm and tamper-proof alarm at the same time to support the model for the 4.3-inch, 5-inch and 8-inch access control series, other access control models only support the fire alarm, all the access control series of products to support the door magnetism function, please refer to the actual model! Please refer to the actual model.

2.API Description

Items	Description
Interface Name	Access control alarm message information reporting
Uplink data topic	mqtt/face/ID/Alarm（Where ID refers to the device ID/SN，e.g,:1306612）
Downlink data topic	mqtt/face/ID（Where ID refers to the device ID/SN，e.g,:1306612）

3.Description of reported fields

Key	Type	Values	Description
operator	String	AlarmInfoPush	Access control alarm message information reporting
info	String		Concrete content
facesluiceId	String		The clientID for connecting to the server,device ID/SN is used by default to ensure uniqueness
code	int		Alarm event code 1: Door magnetic door opening timeout alarm 2: Forced door opening alarm 3: Takedown alarm 4: Live attack alarm 5: Fire alarm 6: Door magnetic status
description	String		Alarm even sescription
event_time	String		Alarm event timestamp
status	Int		Alarm reporting event 1. Alarm occurs 2. Alarm is canceled When code is 6: 1. Door magnet is in open (disconnected) state 2. Door magnet is in closed (suction) state
reserve	String		Reserve

4.Reporting Example

4.1 Door Magnetic Status Reporting Example

```
{
    "operator": "AlarmInfoPush",
    "info": {
        "code": 6,
        "facesluiceId": "1478428",
        "description": "Magnetic door state",
        "event_time": "1621244121606",
        "status": 1,
        "reserve": "reserve"
    }
}
```

4.2 Tamper Alarm Reporting Example

```
{
    "operator": "AlarmInfoPush",
    "info": {
        "code": 3,
        "facesluiceId": "1478428",
        "description": "Tamper alarm"",
        "event_time": "1621250724208",
        "status": 1,
        "reserve": "reserve"
    }
}
```

5、System Interface

5.1 Personnel Management

Personnel management mainly involves the management of personnel interactions between the platform and the Face Recognition All-in-One Machine.

5.1.1 Single Personnel List Operation

5.1.1.1 Single Personnel List Addition or Modification

1.Description

This interface distinguishes whether this person is added or modified by determining the customId of the person's information; if the corresponding value of the customId is not found in the device, it is determined to be added; if the corresponding value of the customId exists, it is determined to be modified. Due to the queuing mechanism of MQTT and the consumption speed of the device, it is necessary to wait for the last call of this interface to return an Ack before the next call can be executed.

2.API Description

Items	Description
Interface Name	Single personnel list addition or modification
Data flow (active command side)	Platform->Device
Downlink data topic	mqtt/face/ID（Where ID refers to the device ID/SN，e.g,:1306612）
Uplink data topic	mqtt/face/ID/Ack（Where ID refers to the device ID/SN，e.g,:1306612）

3.Single Personnel List Field Description
Parameter information(Note:`Optional`is optional):

Key	Type	Values	Description
operator	String	EditPerson	Single personnel list addition or modification
messageId	String		Message id, used by the platform to distinguish each message.
info	Object		Concrete content
customId	String	Length 48 characters (including terminator)	Platform generated id, uniquely identifies different personnel, it is recommended to use the identity card number, the incoming customId device already exists as a modified personnel, otherwise the increase in personnel
name	String (optional)	Length 32 characters (including terminator)	Name
personType	int	0~1	0: white list 1: black list
tempCardType	int	0~3	List type 0: Permanent list 1: Temporary list 1 (starting and ending time periods) 2: Temporary list 2 (daily time slots) 3: Temporary list 3 (number of times valid) 4: Temporary List 4 (Valid for the same time period every day) Note: Default invalidated lists will be automatically deleted by the device after the next recognition or after 24 hours.
cardValidBegin	String (optional)	2020-04-18 00:00:00	Start time of the temporary list , YYYY-MM-DD hh:mm:ss Required if list type is temporary 1, 2, 4
cardValidEnd	String (optional)	2020-12-20 23:59:59	End time of the temporary list Required if list type is temporary 1, 2, 4

Key	Type	Values	Description
EffectNumber	int (optional)		Effective number of passes through Temporary List 3 or Temporary List 4
nation	int (optional)Reserve	Reserve	Reserve
gender	int (optional)	0~1	Genders 0: male 1: female
idCard	String (optional)		ID number,maximum length is 32 characters (including terminator)
birthday	String (optional)	1992-06-15	Birthday，YYYY-MM-DD
telnum1	String (optional)		Telephone number ,maximum length is 32 characters (including terminator)
native	String (optional)		Native，maximum length is 32 characters (including terminator)
address	String (optional)		Address，maximum length is 72 characters (including terminator)
notes	String (optional)		Notes，maximum length is 64 characters (including terminator)
cardType	int (optional)	0	ID Type 0:ID
cardType2	int (optional)	0~3	Wiegand Card Number Generation Method 0: public number 1: automatic generation 2: manual input 3: do not use access card numbers
CardMode	unsigned int (optional)	0~1	Component Access Card Number Model 0: Decimal Composition Card Number 1: Hexadecimal Composition Card Number **Default 0: Decimal Composition Card Number**

Key	Type	Values	Description
WiegandType	int	0~1 or 6~7	The Wiegand protocols that form the basis for the Wiegand card numbers **Must be filled in when cardType2=2** ; 0: 26 bits 1: 34 bits 6:26 bits(8+16 facility code+userid (fill in separately); 7:34 bits(8+24 facility code+userid(fill in separately)) **Default:1(34 bits)**
cardNum2	String (optional)		Wiegand Access Card Number(userid) Must be filled in when cardType2=2
WGFacilityCode	int (optional)		Facility code Must be filled in when WiegandType= 6 or 7 use with cardNum2. Not required WiegandType=0 or 1
RFCardMode	unsigned int (optional)	0~1	Component RF (ID) Card Number Model，For built-in card machine type 0:Decimal Composition Card Number 1:Hexadecimal Composition Card Number Default 1: Hexadecimal Composition Card Number
RFIDCard	String (optional)	If RFCardMode=0,fill in the decimal string (“1369406761”) ; If RFCardMode=1,fill in the decimal string (“519F7D29”)	ID card number, the maximum length of the decimal length of 10 characters, for the built-in card machine models (including terminator), if you just re-edit the list, do not change the card number, you do not need to pass the RFIDCard keyword

Key	Type	Values	Description
isCheckSimilarity	int (optional)	0-1	Whether or not to perform image verification (if the similarity of the added image is greater than the black/white list threshold, it indicates that the list already exists) 0:not calibrated 1:calibrated
pic	String	Use either with picURI Not required when changing the list without replacing the image.	Personnel images (base64 encoded, no more than 1M)
picURI	String	Use either with pic Not required when changing the list without replacing the image.	Personnel images (URI) **Note: When a domain address is used, the device's DNS needs to be set correctly**
PersonalPassword	char (optional)		Personal password(six-digit access password)
strategyInfo	String (optional)		List Associated Access Strategy Information Keywords
strategyData	String (optional)		List Associated Access Policy Information JSON Array Keywords **Required if strategyInfo is not empty**
strategyNum	int (optional)		Access strategy number **Required if strategyData is not empty**
strategyID	String (optional)		Access strategy ID **Required if strategyData is not empty**
strategyName	String (optional)		Access strategy name (64 bytes with terminator) , Reserve

4.Example of Single Personnel List Addition or Modification

4.1 Base64 image data

```
{
    "messageId": "ID:localhost-637050272518414388:79346:87:5",
    "operator": "EditPerson",
    "info": {
        "customId": "063c81e0fce184c696cdb7e049230f5e",
        "name": "AAA",
        "nation": 1,
        "gender": 0,
        "birthday": "1995-06-12",
        "address": "",
        "idCard": "421381199504030001",
        "tempCardType": 0,
        "EffectNumber": 3,
        "cardValidBegin": "2019-10-10 10:00:00",
        "cardValidEnd": "2019-10-10 16:00:00",
        "telnum1": "18888888888",
        "native": "Shenzhen",
        "cardType2": 0,
        "cardNum2": "",
        "notes": "",
        "personType": 0,
        "cardType": 0,
        "strategyInfo":
        {
            "strategyNum":1,
            "strategyData":[
                {"strategyID":"1","strategyName":"test1"}
            ]
        },
        "pic": "......../*Fill in the image base64 encoded data, not more than 1M*/"
    }
}
```

4.2 Get the image data via URI.

```
{
    "messageId": "ID:localhost-637050272518414388:79346:87:5",
    "operator": "EditPerson",
    "info": {
        "customId": "063c81e0fce184c696cdb7e049230f5e",
        "name": "AAA",
        "nation": 1,
        "gender": 0,
        "birthday": "1995-06-12",
        "address": "",
        "idCard": "421381199504030001",
        "tempCardType": 0,
        "EffectNumber": 3,
```

```
        "cardValidBegin": "2019-10-10 10:00:00",
        "cardValidEnd": "2020-10-10 16:00:00",
        "telnum1": "18888888888",
        "native": "Guangdong Shenzhen",
        "cardType2": 0,
        "cardNum2": "",
        "notes": "",
        "personType": 0,
        "cardType": 0,
        "strategyInfo"://Associated Access Strategy Information
        {
            "strategyNum":1,
            "strategyData":[
                {"strategyID":"1","strategyName":"test1"}
            ]
        },
        "picURI": "https://btgoss.oss-cn-beijing.aliyuncs.com/image/xxx.jpg"
    }
}
```

5.Device Response Field Description

Parameter information(Note: `Optional` is optional):

Key	Type	Values	Description
operator		EditPerson	Answer to single personnel list addition or modification
messageId	String		Message id, used by the platform to distinguish each message.
code	String		Command execution error code 200-successes,see also 10.1 Single Personnel List Addition or Modification Error Codes
info	Object		Concrete content
facesluiceId	string		The clientID for connecting to the server,device ID/SN is used by default to ensure uniqueness
personId	String(option)	1-N	Face Machine Database Self-Added ID
customId	String	Length 48 characters (including terminator)	Platform generated id, uniquely identifies different personnel
result	String	"ok"	Operating result
detail	String (option)		Error message when Result is "Fail"

6.Device Response Example

```
{
    "messageId": "ID:localhost-637050272518414388:79346:87:5",
    "operator": "EditPerson-Ack",
    "code": "200",
    "info": {
        "facesluiceId": "1306612",
        "personId": "3",
        "customId": "063c81e0fce184c696cdb7e049230f5e",
        "result": "ok"
    }
}
```

5.1.2 Batch Personnel Operation

The batch personnel operation interface is designed to reduce the slow consumption speed on the device side caused by a single personnel operation, which may lead to equipment and server reconnection problems. The batch personnel operation interfaces are mutually exclusive, and the QueryProgress interface is polled to query the completion status of the last batch personnel operation interface within a certain time interval. Batch personnel operations are suitable for full (large) updating of personnel.

5.1.2.1 Batch List Additions or Modifications

1.Description

The maximum number of lists is limited to 1,000, and the platform needs to wait for the success of the last add or modify list before sending out the lists again. Only support URI way to get images. The device processes the list of personnel based on the value of the incoming personnel's unique identifier customId. If the value of customId doesn't exist, then new personnel are added, and if the value of customId exists, then the corresponding personnel information is modified.After this interface is completed, it will return the number of failures and successes as well as information about the failure or success of a specific person at one time. The list of failed additions or modifications will be accompanied by an error code, please refer to the Error Cross-Reference Table for the error code.You can't call other commands related to adding/modifying/deleting lists during the process of adding or modifying people in bulk. After issuing batch add/modify interface, you can call QueryProgress at intervals to judge the completion of batch add/modify personnel interface.

In batch issuing personnel, it is recommended to issue personnel without binding access strategy , which is slow in efficiency, and it is recommended to separate personnel issuing and personnel binding access strategy . The steps are as follows:

2.API Description

Items	Description
Interface Name	Batch list additions or modifications
Data flow (active command side)	Platform->Device
Downlink data topic	mqtt/face/ID（Where ID refers to the device ID/SN，e.g,:1306612）
Uplink data topic	mqtt/face/ID/Ack（Where ID refers to the device ID/SN，e.g,:1306612）

3.Batch List Additions or Modifications Field Descriptions
Parameter information(Note:`Optional`is optional):

Key	Type	Values	Description
operator		EditPersonsNew	Batch list additions or modifications
messageId	String		Message id, used by the platform to distinguish each message.
DataBegin	String	Settled:BeginFlag	Packet start marker, detects packet integrity
PersonNum	int	1~1000	The number of people, must be the same as the number of json corresponding to the person information.
info	Object		Concrete content
[ ]			List array
customId	String		Platform generated id, uniquely identifies different personnel. It is recommended to use the identity card number, the incoming customId device already exists as a modified personnel, otherwise the increase in personnel
name	String (optional)		Name
personType	int	0~1	0: white list 1: black list
tempCardType	int	0~3	List type 0: Permanent list 1: Temporary list 1 (starting and ending time periods) 2: Temporary list 2 (daily time slots) 3: Temporary list 3 (number of times valid) 4: Temporary List 4 (Valid for the same time period every day) Note: Default invalidated lists will be automatically deleted by the device after the next recognition or after 24 hours.
cardValidBegin	String (optional)	2020-04-18 00:00:00	Start time of the temporary list，YYYY-MM-DD hh:mm:ss Required if list type is temporary 1, 2, 4
cardValidEnd	String (optional)	2020-12-20 23:59:59	End time of the temporary list Required if list type is temporary 1, 2, 4

Key	Type	Values	Description
EffectNumber	int (optional)		Effective number of passes through Temporary List 3 or Temporary List 4
nation	int (optional)	Reserve	Reserve
gender	int (optional)	0~1	Genders 0: male 1: female
idCard	String (optional)		ID number,maximum length is 32 characters (including terminator)
telnum1	String (optional)		Telephone number ,maximum length is 32 characters (including terminator)
native	String (optional)		Native，maximum length is 32 characters (including terminator)
address	String (optional)		Address，maximum length is 72 characters (including terminator)
birthday	String (optional)	1992-06-15	Birthday，YYYY-MM-DD
notes	String (optional)		Notes，maximum length is 64 characters (including terminator)
isCheckSimilarity	int (optional)	0~1	Whether or not to perform image verification (if the similarity of the added image is greater than the black/white list threshold, it indicates that the list already exists) 0:not calibrated 1:calibrated
cardType2	int (optional)	0~3	Wiegand Card Number Generation Method 0: public number 1: automatic generation 2: manual input 3: do not use access card numbers

Key	Type	Values	Description
WiegandType	int	0~1 or 6~7	Wiegand protocols that form the basis for the Wiegand card numbers **Must be filled in when cardType2=2** ; 0: 26 bits 1: 34 bits 6:26 bits(8+16 facility code+userid (fill in separately); 7:34 bits(8+24 facility code+userid(fill in separately)) **Default:1(34 bits)**
WGFacilityCode	int (optional)		Facility code Must be filled in when WiegandType= 6 or 7 use with cardNum2. Not required WiegandType=0 or 1
cardNum2	int (optional)		Wiegand Access Card Number(userid) Must be filled in when cardType2=2
CardMode	unsigned int (optional)	0~1	Component Access Card Number Model 0: Decimal Composition Card Number 1: Hexadecimal Composition Card Number **Default 0: Decimal Composition Card Number**
RFCardMode	unsigned int (optional)	0~1	Component RF (ID) Card Number Model , For built-in card machine type 0:Decimal Composition Card Number 1:Hexadecimal Composition Card Number Default 1: Hexadecimal Composition Card Number
RFIDCard	int (optional)	If RFCardMode=0,fill in the decimal string (“1369406761”) ; If RFCardMode=1,fill in the decimal string (“519F7D29”)	ID card number, the maximum length of the decimal length of 10 characters, for the built-in card machine models (including terminator), if you just re-edit the list, do not change the card number, you do not need to pass the RFIDCard keyword
cardType	int (optional)		ID Type 0:ID

Key	Type	Values	Description
picURI	String		Use either with pic Not required when changing the list without replacing the image.
PersonalPassword	char (optional)		Personal password(six-digit access password)
strategyInfo	String (optional)		List Associated Access Strategy Information Keywords
strategyDate	String (optional)		List Associated Access Policy Information JSON Array Keywords **Required if strategyInfo is not empty**
strategyNum	int (optional)		Access strategy number , Reserve
strategyID	String (optional)		Access strategy ID
strategyName	String (optional)		Access strategy name (64 bytes with terminator) , Reserve
DataEnd	String	Settled:EndFlag	Packet end marker, detects packet integrity

4.Example of Batch List Additions or Modifications

```
{
    "messageId": "EditPersonsNewlist2020-07-24T19:07:00_00002",
    "DataBegin": "BeginFlag",
    "operator": "EditPersonsNew",
    "PersonNum": 1000,
    "info": [{
            "customId": "063c81e0fce184c696cdb7e049230f5e23dfqwxc230000",
            "name": "modify000",
            "telnum1": "13700880000"
        },
        /*List of 998 persons omitted here*/
        {

            "customId": "063c81e0fce184c696cdb7e049230f5e23dfqwxc230999",
            "name": "modify999",
            "picURI": "https://btgongpluss.oss-cn-
beijing.aliyuncs.com/bigheadphoto/xxx111.jpg"
        }
    ],
    "DataEnd": "EndFlag"
}
```

5.Device Response Field Descriptions
Parameter information(Note: `Optional` is optional):

Key	Type	Values	Description
operator		EditPersonsNew-Ack	Answer to batch list additions or modifications
messageId	String		Message id, used by the platform to distinguish each message.
code	String		Command execution error code 200-successes,see also 10.2 Batch List Additions or Modifications Error Codes
info	Object		Concrete content
facesluiceId	String		The clientID for connecting to the server,device ID/SN is used by default to ensure uniqueness
AddErrNum	String	0~1000	Number of failed batch additions or modifications, return parameter
AddErrInfo	Object		Batch add or modify personnel failure information(CustomizeID/PersonUUID + errcode),return parameter，errcode see appendice
AddSucNum	String	0~1000	Number of successful batch additions or modifications,return parameter
AddSucInfo	Object		Batch add or modify personnel success information(CustomizeID/PersonUUID) ,return parameter(customId)
customId	String	Length 48 characters (including terminator)	Platform generated id, uniquely identifies different personnel.
result	String	"ok"/"fail"	Operating result
detail	String (optional)		Error message when Result is "Fail"

6.Device Response Example

```
{
    "messageId": "EditPersonsNewlist2020-07-24T19:07:00_00002",
    "operator": "EditPersonsNew-Ack",
```

```
    "code": "200",
    "info": {
        "facesluiceId": "1306612",
        "AddErrNum": "1",
        "AddSucNum": "999",
        "AddErrInfo": [{
            "customId": "063c81e0fce184c696cdb7e049230f5e23dfqwxc230898",
            "errcode": "461"
        }],
        "AddSucInfo": [{
                "customId": "063c81e0fce184c696cdb7e049230f5e23dfqwxc230000"
            },
            {
                "customId": "063c81e0fce184c696cdb7e049230f5e23dfqwxc230000"
            },
            /*List of 996 persons omitted here*/
            {
                "customId": "063c81e0fce184c696cdb7e049230f5e23dfqwxc230999"
            }
        ],
        "result": "ok"
    }
}
```

5.1.2.2 Batch Add or Modify Progress Query

1.Description

This interface allows you to query the progress of batch interface execution, whether the device is currently processing AddPersons (legacy interfaces), EditPersons (legacy interfaces), EditPersonsNew batch interface tasks or is in the state of None not processing batch tasks, the platform can obtain the status of the device through this interface, to coordinate the processing of tasks issued by the list.

2.API Description

Items	Description
Interface Name	Batch add or modify progress query
Data flow (active command side)	Platform->Device
Downlink data topic	mqtt/face/ID（Where ID refers to the device ID/SN，e.g,:1306612）
Uplink data topic	mqtt/face/ID/Ack（Where ID refers to the device ID/SN，e.g,:1306612）

3.Batch Add or Modify Progress Query Field Descriptions
Parameter information(Note:`Optional`is optional):

Key	Type	Values	Description
operator	String	QueryProgress	Batch add or modify progress query
messageId	String		Message id, used by the platform to distinguish each message.
info	Object		Concrete content

4.Example of Batch Add or Modify Progress Query

```
{
    "messageId": "ID:localhost-637050272518414388:79346:87:1",
    "operator": "QueryProgress",
    "info": {}
}
```

5.Device Response Field Descriptions
Parameter information(Note:`Optional`is optional):

Key	Type	Values	Description
operator		QueryProgress-Ack	Answer to batch add or modify progress query
messageId	String		Message id, used by the platform to distinguish each message.
code	String		Command execution error code 200-successes,see also 10.3 Batch Add or Modify Progress Query Error Codes
info	Object		Concrete content
facesluiceId	String		The clientID for connecting to the server,device ID/SN is used by default to ensure uniqueness
QueryType	String		Interface of the batch task currently being processed by the device
Status	String		Current status of the device 0:Idle, no tasks 1:Processing Batch Add Personnel Tasks 2:Processing Batch Modify Personnel Tasks 3:Processing Batch Add or Modify Personnel Tasks
result	String	"ok"/"fail"	Operating result
detail	String (optional)		Error message when Result is "Fail"

6.Device Response Example

```
{
    "messageId": "ID:localhost-637050272518414388:79346:87:1",
    "operator": "QueryProgress-Ack",
    "code": "200",
    "info": {
        "facesluiceId": "2864575",
        "QueryType": "EditPersonsNew",
        "Status": "3",
        "result": "ok"
    }
}
```

5.1.3 Deletion of Personnel Lists

5.1.3.1 Deletion of Single Personnel List

1.Description

This interface deletes the corresponding personnel according to customId, when deleting personnel in large quantities, you need to wait for the previous command to be executed successfully before deleting the next personnel, and it is recommended to use other interfaces for deleting personnel in full quantities (in large quantities).

2.API Description

Items	Description
Interface Name	Deletion of single personnel list
Data flow (active command side)	Platform->Device
Downlink data topic	mqtt/face/ID（Where ID refers to the device ID/SN，e.g,:1306612）
Uplink data topic	mqtt/face/ID/Ack（Where ID refers to the device ID/SN，e.g,:1306612）

3.Deletion of Single Personnel List Field Descriptions
Parameter information(Note:`Optional`is optional):

Key	Type	Values	Description
operator		DelPerson	Deletion of single personnel list
messageId	String		Message id, used by the platform to distinguish each message.
info	Object		Concrete content
customId	String	Length 48 characters (including terminator)	Platform generated id, uniquely identifies different personnel.It is recommended to use the identity card number, the incoming customId device already exists as a modified personnel, otherwise the increase in personnel

4.Example of Deletion of Single Personnel List

```
{
    "operator": "DelPerson",
    "messageId": "ID:localhost-637046811507388956:23952:65:48",
    "info": {
        "customId": "063c81e0fce184c696cdb7e049230f5e"
    }
}
```

5.Device Response Field Descriptions
Parameter information(Note:`Optional`is optional):

Key	Type	Value	Description
operator		DelPerson-Ack	Answer to deletion of single personnel list
messageId	String		Message id, used by the platform to distinguish each message.
code	String		Command execution error code 200-successes,see also 10.4 Deletion of Single Personnel List Error Codes
info	Object		Concrete content
facesluiceId	String		The clientID for connecting to the server,device ID/SN is used by default to ensure uniqueness
personId	String		Device database keyword self-incrementing ID
customId	String	Length 48 characters (including terminator)	Platform generated id, uniquely identifies different personnel.
result	String	"ok"/"fail"	Operating result
detail	String (optional)		Error message when Result is "Fail"

6.Device Response Example

```
{
    "messageId": "ID:localhost-637046811507388956:23952:65:48",
    "operator": "DelPerson-Ack",
    "code": "200",
    "info": {
        "facesluiceId": "1306612",
        "personId": "3",
        "customId": "063c81e0fce184c696cdb7e049230f5e",
        "result": "ok"
    }
}
```

5.1.3.2 Batch Deletion of Personnel Lists

1.Description

This interface deletes the corresponding personnel according to the customId array, this interface supports a maximum of 200 personnel lists to be deleted at one time, when the maximum number of personnel lists are deleted, it is necessary to wait for this command to return (usually it takes 7 seconds to complete) before you can issue other commands again or issue batch deletion of the personnel lists again, after this interface deletes the lists, the registration images of the corresponding control records are deleted, and do not delete the control record. Incoming customId will be executed successfully, incoming non-existing customId will be prompted in failure, deleting non-existing customId will not affect the platform use, if the platform doesn't care, it can't deal with the failure return message of deleting non-existing customId.

2.API Description

Items	**Description**
Interface Name	Batch deletion of personnel lists
Data flow (active command side)	Platform->Device
Downlink data topic	mqtt/face/ID（Where ID refers to the device ID/SN，e.g,:1306612）
Uplink data topic	mqtt/face/ID/Ack（Where ID refers to the device ID/SN，e.g,:1306612）

3.Batch Deletion of Personnel Lists Field Descriptions
Parameter information(Note:`Optional`is optional):

Key	**Type**	**Values**	**Description**
operator		DeletePersons	Batch deletion of personnel lists
messageId	String		Message id, used by the platform to distinguish each message.
PersonNum	int	0-200	The number of people, must be the same as the number of json corresponding to the person information.
DataBegin	String	Settled:BeginFlag	Packet start marker, detects packet integrity
info	Object		Concrete content
customId	Array	The length of each id is 48 characters (including terminator)	Platform generated id, uniquely identifies different personnel.It is recommended to use the identity card number, the incoming customId device already exists as a modified personnel, otherwise the increase in personnel
DataEnd	String	Settled:EndFlag	Packet end marker, detects packet integrity

4.Example of Batch Deletion of Personnel Lists

```
{
    "messageId": "2020-05-14 11:07:00",
    "DataBegin": "BeginFlag",
    "operator": "DeletePersons",
    "PersonNum": 200,
    "info": {
        "customId": [
            "063c81e0fce184c696cdb7e049230f5e23dfqwxc230000",
            /*198 data omitted here*/
            "063c81e0fce184c696cdb7e049230f5e23dfqwxc230199"
        ]
    },
    "DataEnd": "EndFlag"
}
```

5.Device Response Field Descriptions

Parameter information(Note:`Optional`is optional):

Key	Type	Values	Description
operator		DeletePersons-Ack	Answer to batch deletion of personnel lists
messageId	String		Message id, used by the platform to distinguish each message.
code	String		Command execution error code 200-successes,see also 10.5 Batch Deletion of Personnel Lists Error Codes
info	Object		Concrete content
facesluiceId	String		The clientID for connecting to the server,device ID/SN is used by default to ensure uniqueness
DelErrNum	String	0~200	The number of failed batch deletions of personnel. Deletions of personnel that do not exist on the device fail.
DelErrInfo	Array		Batch delete person list failure specific array information.
DelSucNum	String	0~200	Number of successful batch deletions.
DelErrInfo	Array		Batch delete person list success specific array information.
customId	String	Length 48 characters (including terminator)	Platform generated id, uniquely identifies different personnel.
result	String	"ok"/"fail"	Operating result
detail	String (optional)		Error message when Result is "Fail"

6.Device Response Example

```
{
    "messageId": "2020-05-14 11:07:00 DeletePersons",
    "operator": "DeletePersons-Ack",
    "code": "200",
    "info": {
        "facesluiceId": "1306612",
        "DelErrNum": "1",
        "DelSucNum": "199",
        "DelErrInfo": [{
```

```
            "customId": "063c81e0fce184c696cdb7e049230f5e23dfqwxc230199"
        }],
        "DelSucInfo": [{
                "customId": "063c81e0fce184c696cdb7e049230f5e23dfqwxc230000"
            },
            /*197 data omitted here*/
            ,
            {
                "customId": "063c81e0fce184c696cdb7e049230f5e23dfqwxc230198"
            }
        ],
        "result": "ok"
    }
}
```

5.1.3.3 All Lists Deleted

1.Description

All personnel list deletion interface at the same time will delete all personnel list **control records (authentication records) and can not be restored** , careful operation. Call to delete all personnel list success, face recognition all-in-one opportunity to **automatic restart!**

2.API Description

Items	Description
Interface Name	All lists deleted
Data flow (active command side)	Platform->Device
Downlink data topic	mqtt/face/ID (Where ID refers to the device ID/SN，e.g,:1306612)
Uplink data topic	mqtt/face/ID/Ack (Where ID refers to the device ID/SN，e.g,:1306612)

3.All Lists Deleted Field Descriptions

Parameter information(Note:`Optional`is optional):

Key	Type	Values	Description
operator		DeleteAllPerson	All lists deleted
messageId	String		Message id, used by the platform to distinguish each message.
info	Object		Concrete content
deleteall	int	0~1	Whether to delete all personnel lists 1: Confirm deletion

4.Example of All Lists Deleted

```
{
    "messageId": "ID:localhost-637046811507388956:23952:65:48",
    "operator": "DeleteAllPerson",
    "info": {
        "deleteall": 1
    }
}
```

5.Device Response Field Descriptions
Parameter information(Note:`Optional`is optional):

Key	Type	Values	Description
operator		DeleteAllPerson-Ack	Answer to all lists deleted
messageId	String		Message id, used by the platform to distinguish each message.
code	String		Command execution error code 200-successes,see also 10.6 All Lists Deleted Error Codes
info	Object		Concrete content
facesluiceId	String		The clientID for connecting to the server,device ID/SN is used by default to ensure uniqueness
result	String	"ok"/"fail"	Operating result
detail	String (optional)		Error message when Result is "Fail"

6.Device Response Example

```
{
    "messageId": "ID:localhost-637046811507388956:23952:65:48",
    "operator": "DeleteAllPerson-Ack",
    "code": "200",
    "info": {
        "facesluiceId": "1306612",
        "result": "ok"
    }
}

```

5.1.4 Query Personnel List Information

5.1.4.1 Query All Personnel customId

1.Description

This interface returns the customId key value for all people in the device, but excludes people added from the device web backend, etc., because the customId is only available to the platform in order to ensure uniqueness.

2.API Description

Items	Description
Interface Name	Query all personnel customId
Data flow (active command side)	Platform->Device
Downlink data topic	mqtt/face/ID（Where ID refers to the device ID/SN，e.g,:1306612）
Uplink data topic	mqtt/face/ID/Ack（Where ID refers to the device ID/SN，e.g,:1306612）

3.Query All Personnel customId Field Descriptions
Parameter information(Note:`Optional`is optional):

Key	Type	Values	Description
operator		QueryPerson	Query all personnel customId
messageId	String		Message id, used by the platform to distinguish each message.
info	Object		Concrete content

4.Example of Query All Personnel customId

```
{
    "operator": "QueryPerson",
    "messageId": "ID:localhost-637046811507388956:23952:65:48",
    "info": {}

}
```

5.Device Response Field Descriptions
Parameter information(Note:`Optional`is optional):

Key	Type	Values	Description
operator		QueryPerson-Ack	Answer to query all personnel customId
messageId	String		Message id, used by the platform to distinguish each message.
code	String		Command execution error code 200-successes,see also error cross-reference table
info	Object		Concrete content
facesluiceId	String		The clientID for connecting to the server,device ID/SN is used by default to ensure uniqueness
TotalPersonNum	String	1-N	Total number of device personnel (including lists with empty customId, e.g., personnel added by the equipment web backend)
QueryPersonNum	String	1-N	Total number of personnel with a customId present in the device (excluding personnel with a null customId)
customId	String	Length 48 characters (including terminator)	Platform generated id, uniquely identifies different personnel.
result	String	"ok"/"fail"	Operating result
detail	String (optional)		Error message when Result is "Fail"

6.Device Response Example

```
{
    "messageId": "ID:localhost-637046811507388956:23952:65:48",
    "operator": "QueryPerson-Ack",
    "code": "200",
    "info": {
        "facesluiceId": "1306612",
        "TotalPersonNum": "99",
        "QueryPersonNum": "98",
        "customId":
"063c81e0fce184c696cdb7e049230f5e23dfqwxc230000,063c81e0fce184c696cdb7e049230f5e23d
fqwxc230001,
        063c81e0fce184c696cdb7e049230f5e23dfqwxc230097,",
        "result": "ok"
    }
}
```


5.1.4.2 Query Single Personnel List Information

1.Description

This interface returns the details of the person corresponding to the customId key in the device.

2.API Description

Items	Description
Interface Name	Query Single Personnel List Information
Downlink data topic	mqtt/face/ID（Where ID refers to the device ID/SN，e.g,:1306612）
Uplink data topic	mqtt/face/ID/Ack（Where ID refers to the device ID/SN，e.g,:1306612）

3.Query Single Personnel List Information Field Descriptions
Parameter information(Note:`Optional`is optional):

Key	Type	Values	Description
operator		SearchPerson	Query single personnel list information
messageId	String		Message id, used by the platform to distinguish each message.
info	Object		Concrete content
customId	String	Length 48 characters (including terminator)	Platform generated id, uniquely identifies different personnel.
Picture	int	0~1(default 0)	Whether to include image information about people 0:List without image information 1:List with image information

4.Example of Query Single Personnel List Information

```
{
    "messageId": "ID:localhost-637046811507388956:23952:65:48",
    "operator": "SearchPerson",
    "info": {
        "customId": "063c81e0fce184c696cdb7e049230f5e",
        "Picture": 1
    }
}
```

5.Device Response Field Descriptions
Parameter information(Note:`Optional`is optional):

Key	Type	Values	Description
operator		SearchPerson-Ack	Answer to query single personnel list information
messageId	String		Message id, used by the platform to distinguish each message.
code	String		Command execution error code 200-successes,see also 10.7 Query Single Personnel List Information Error Codes
info	Object		Concrete content
facesluiceId	String		The clientID for connecting to the server,device ID/SN is used by default to ensure uniqueness.
personId	String		Device database keyword self-incrementing ID
customId	String	Length 48 characters (including terminator)	Platform generated id, uniquely identifies different personnel.
name	String	Length 32 characters (including terminator)	Name
gender	String	0~1	Genders 0: male 1: female
cardType	String (optional)	0	ID Type 0:ID
idCard	String		ID number,maximum length is 32 characters (including terminator)
birthday	String	1992-06-15	Birthday，YYYY-MM-DD
address	String		Address，maximum length is 72 characters (including terminator)
creatTime	String	2023-01-01T00:00:00	Personnel creation time.
telnum1	String		Telephone number ,maximum length is 32 characters (including terminator)

Key	Type	Values	Description
personType	String	0~1	0: white list 1: black list
nation	String	Reserve	Reserve
native	String		Native，maximum length is 32 characters (including terminator)
notes	String		Notes，maximum length is 64 characters (including terminator)
PersonalPassword	String (optional)		Personal password(six-digit access password)
tempCardType	String	0~3	List type 0: Permanent list 1: Temporary list 1 (starting and ending time periods) 2: Temporary list 2 (daily time slots) 3: Temporary list 3 (number of times valid) 4: Temporary List 4 (Valid for the same time period every day)
EffectNumber	String		Effective number of passes through Temporary List 3 or Temporary List 4
cardValidBegin	String	2020-04-18 00:00:00	Start time of the temporary list，YYYY-MM-DD hh:mm:ss Required if list type is temporary 1, 2, 4
cardValidEnd	String	2020-12-20 23:59:59	End time of the temporary list Required if list type is temporary 1, 2, 4
cardType2	String		Wiegand Card Number Generation Method 0: public number 1: automatic generation 2: manual input 3: do not use access card numbers
cardNum2	String		Wiegand Access Card Number(userid) Must be filled in when cardType2=2
CardMode	String	0~1	Component Access Card Number Model 0: Decimal Composition Card Number 1: Hexadecimal Composition Card Number **Default 0: Decimal Composition Card Number**

Key	Type	Values	Description
WGFacilityCode	int (optional)		Facility code Must be filled in when WiegandType= 6 or 7 use with cardNum2. Not required WiegandType=0 or 1
RFCardMode	unsigned int (optional)	0~1	Component RF (ID) Card Number Model， For built-in card machine type 0:Decimal Composition Card Number 1:Hexadecimal Composition Card Number Default 1: Hexadecimal Composition Card Number
RFIDCard	String (optional)		If RFCardMode=0,fill in the decimal string (“1369406761”) ; If RFCardMode=1,fill in the decimal string (“519F7D29”)
strategyNum	int	1~N	Access strategy number(return value)
strategyID	int	1~N	Access strategy ID(return value)
result	String	"ok"/"fail"	Operating result
detail	String (optional)		Error message when Result is "Fail"
pic	String (optional)		Personnel images (base64 encoded, no more than 1M))

6.Device Response Example


```
{
    "messageId": "ID:localhost-637046811507388956:23952:65:48",
    "operator": "SearchPerson-Ack",
    "code": "200",
    "info": {
        "facesluiceId": "1306612",
        "personId": "1",
        "customId": "063c81e0fce184c696cdb7e049230f5e",
        "name": "Test",
        "gender": "0",
        "idCard": "421381199504030001",
        "birthday": "1995-06-12",
        "address": "Shenzhe XXX ",
        "creatTime": "2023-01-01T00:00:00",
        "telnum1": "18888888888",
        "personType": "0",
```

```
        "nation": "1",
        "native": "GuangdongShenzhen",
        "notes": "note",
        "strategyNum": 2,
        "strategyID": [1, 2],
        "cardType2": "0",
        "cardNum2": "1",
        "CardMode": "0",
        "tempCardType": "0",
        "EffectNumber": "0",
        "cardValidBegin": "0000-00-00 00:00:00",
        "cardValidEnd": "0000-00-00 00:00:00",
        "result": "ok"
    },
    "pic": "*Data omitted here*/"
}
```

5.1.4.3 Query Multiple Personnel List Information

1.Description

This interface supports two methods for searching, currently does not support searching with image return.
Query Description:

1. Query by time: set the start time and the end time of the query, the name is empty.
2. Query by name or name fuzzy: set the start time and the end time of the query is empty, the name is not empty.

2.API Description

Items	Description
Interface Name	Query multiple personnel list information
Downlink data topic	mqtt/face/ID（Where ID refers to the device ID/SN，e.g,:1306612）
Uplink data topic	mqtt/face/ID/Ack（Where ID refers to the device ID/SN，e.g,:1306612）

3.Query Multiple Personnel List Information Field Descriptions
Parameter information(Note:`Optional`is optional):

Key	Type	Values	Description
operator		SearchPersonList	Query multiple personnel list information
messageId	String		Message id, used by the platform to distinguish each message.
info	Object		Concrete content
facesluiceId	String (optional)		The clientID for connecting to the server,device ID/SN is used by default to ensure uniqueness
personType	int (optional)	0~2（default 2）	List Type 0: White list 1: Black list 2:All
BeginTime	String (optional)	2018-08-13T00:00:00	Starting point of search
EndTime	String (optional)	2020-08-19T23:59:59	End point of search
gender	int (optional)	0~2	Genders 0:Male 1:Female 2:All
cardNum2	unsigned int (optional)		Access card number
name	String (optional)		Name
BeginNO	int (optional)	Default 0	Queries the starting position of the list, i.e., from which person
RequestCount	int (optional)	1~2000(Default 1000)	The total number of messages returned by the query list, the maximum number of messages returned is 2000, and only 2000 messages are returned even if the setting exceeds 2000

4.Example of Query Multiple Personnel List Information

4.1 Queries by Time

```
{
    "messageId": "ID:localhost-637046811507388956:23952:65:48",
    "operator": "SearchPersonList",
    "info": {
        "facesluiceId": "1306612",
        "personType": 0,
        "BeginTime": "2018-08-13T00:00:00",
        "EndTime": "2021-08-19T23:59:59",
        "gender": 2,
        "BeginNO": 0,
        "RequestCount": 100
    }
}
```

4.2 Queries by Name or Name Fuzzy

```
{
    "messageId": "ID:localhost-637046811507388956:23952:65:48",
    "operator": "SearchPersonList",
    "info": {
        "facesluiceId": "1306612",
        "personType": 0,
        "name": "xxx",
        "gender": 2,
        "BeginNO": 0,
        "RequestCount": 100
    }
}
```

4.3 Search All

```
{
    "messageId": "ID:localhost-637046811507388956:23952:65:48",
    "operator": "SearchPersonList",
    "info": {
        "facesluiceId": "1306612"
    }
}
```

5.Device Response Field Descriptions

Parameter information(Note:`Optional`is optional):

Key	Type	Values	Description
operator		SearchPersonList-Ack	Answer to query multiple personnel list information
messageId	String		Message id, used by the platform to distinguish each message.
code	int		Command execution error code 200-successes,see also 10.8 Query Multiple Personnel List Information Error Codes
info	Object		Concrete content
facesluiceId	String		The clientID for connecting to the server,device ID/SN is used by default to ensure uniqueness。
TotalPersonNum	int		Total number of matches in the device list database
PersonNum	int		Number of lists in the reply message
List	Object		Specific list information in the reply message
LibID	int		Device database keyword self-incrementing ID
customId	String	Length 48 characters (including terminator)	Platform generated id, uniquely identifies different personnel.
personType	int	0~1	0: white list 1: black list
name	String	Length 48 characters (including terminator)	Name
gender	int	0~1	Genders 0: male 1: female
nation	int	Reserve	Reserve
cardType	int	0	ID Type 0:ID
idCard	String		ID number,maximum length is 32 characters (including terminator)

Key	Type	Values	Description
birthday	String	1992-06-15	Birthday，YYYY-MM-DD
telnum1	String		Telephone number ,maximum length is 32 characters (including terminator)
native	String		Native，maximum length is 32 characters (including terminator)
address	String		Address，maximum length is 72 characters (including terminator))
notes	String		Notes，maximum length is 64 characters (including terminator)
cardType2	int	0~3	Wiegand Card Number Generation Method 0: public number 1: automatic generation 2: manual input 3: do not use access card numbers
tempCardType	int	0~3	List type 0: Permanent list 1: Temporary list 1 (starting and ending time periods) 2: Temporary list 2 (daily time slots) 3: Temporary list 3 (number of times valid) 4: Temporary List 4 (Valid for the same time period every day)
cardValidBegin	String	2020-04-18 00:00:00	Start time of the temporary list，YYYY-MM-DD hh:mm:ss Required if list type is temporary 1, 2, 4
cardValidEnd	String	2020-12-20 23:59:59	End time of the temporary list Required if list type is temporary 1, 2, 4
EffectNumber	int (optional)		Effective number of passes through Temporary List 3 or Temporary List 4
CardMode	String	0~1	Component Access Card Number Model 0: Decimal Composition Card Number 1: Hexadecimal Composition Card Number
cardNum2	int		Wiegand Access Card Number(userid) Must be filled in when cardType2=2

Key	Type	Values	Description
WGFacilityCode	int (optional)		Facility code Must be filled in when WiegandType= 6 or 7 use with cardNum2. Not required WiegandType=0 or 1
RFCardMode	String (optional)	0-1	Component RF (ID) Card Number Model，For built-in card machine type 0:Decimal Composition Card Number 1:Hexadecimal Composition Card Number Default 1: Hexadecimal Composition Card Number
RFIDCard	String (optional)		ID card number, the maximum length of the decimal length of 10 characters, for the built-in card machine models (including terminator)
strategyNum	int	0~N	Access strategy number(return value)
strategyID	Array	int array,1~N	Access strategy ID(return value)
result	String (optional)	"fail"	Operating result
detail	String (optional)		Error message when Result is "Fail"

6.Device Response Example

6.1 Example of Responses to Queries by Time Period


```
{
    "messageId": "ID:localhost-637046811507388956:23952:65:48",
    "operator": "SearchPersonList-Ack",
    "code": 200,
    "info": {
        "facesluiceId": "1306612",
        "TotalPersonNum": 2,
        "PersonNum": 2,
        "List": [{
                "LibID": 2,
                "personType": 0,
                "name": "xxx",
                "gender": 0,
                "nation": 1,
                "cardType": 0,
                "idCard": "421381199504030001",
                "birthday": "1995-06-12",
                "telnum1": "18888888888",
```

```
                "native": "sz",
                "address": " ",
                "notes": " ",
                "cardType2": 0,
                "strategyNum": 2,
                "strategyID": [1, 2],
                "cardNum2": 1,
                "tempCardType": 0,
                "customId": "210623022466",
                "cardValidBegin": "0000-00-00 00:00:00",
                "cardValidEnd": "0000-00-00 00:00:00"
            },
            {
                "LibID": 1,
                "personType": 0,
                "name": "cff",
                "gender": 0,
                "nation": 1,
                "cardType": 0,
                "idCard": "421381199504030001",
                "birthday": "1995-06-12",
                "telnum1": "18888888888",
                "native": "sz",
                "address": " ",
                "notes": " ",
                "cardType2": 0,
                "strategyNum": 2,
                "strategyID": [1, 2],
                "cardNum2": 1,
                "tempCardType": 0,
                "customId": "210623022456",
                "cardValidBegin": "0000-00-00 00:00:00",
                "cardValidEnd": "0000-00-00 00:00:00"
            }
        ]
    }
}
```


6.2Example of Responses to Queries by Name Fuzzy


```
{
    "messageId": "ID:localhost-637046811507388956:23952:65:48",
    "operator": "SearchPersonList-Ack",
    "code": 200,
    "info": {
        "facesluiceId": "1306612",
        "TotalPersonNum": 2,
        "PersonNum": 2,
        "List": [{
```

```
                "LibID": 2,
                "personType": 0,
                "name": "cff",
                "gender": 0,
                "nation": 1,
                "cardType": 0,
                "idCard": "421381199504030001",
                "birthday": "1995-06-12",
                "telnum1": "18888888888",
                "native": "sz",
                "address": " ",
                "notes": " ",
                "cardType2": 0,
                "strategyNum": 2,
                "strategyID": [1, 2],
                "cardNum2": 1,
                "tempCardType": 0,
                "customId": "210623022466",
                "cardValidBegin": "0000-00-00 00:00:00",
                "cardValidEnd": "0000-00-00 00:00:00"
            },
            {
                "LibID": 1,
                "personType": 0,
                "name": "cff",
                "gender": 0,
                "nation": 1,
                "cardType": 0,
                "idCard": "421381199504030001",
                "birthday": "1995-06-12",
                "telnum1": "18888888888",
                "native": "sz",
                "address": " ",
                "notes": " ",
                "cardType2": 0,
                "strategyNum": 2,
                "strategyID": [1, 2],
                "cardNum2": 1,
                "tempCardType": 0,
                "customId": "210623022456",
                "cardValidBegin": "0000-00-00 00:00:00",
                "cardValidEnd": "0000-00-00 00:00:00"
            }
        ]
    }
}
```

5.2 Access Strategy

Since the access time rules of the temporary list in the personnel list type can't satisfy the multiple time periods to control the access of personnel, this interface is developed to satisfy the time periods to control the access of personnel through the access strategy.

Note: (The various interfaces of the policy are mutually exclusive, in terms of processing, you need to receive the previous task feedback code=200 before executing the next one, otherwise the device will return the previous task busy to the platform; when prompted with the task busy it means that the previous task has not been processed yet, and you need to wait.)

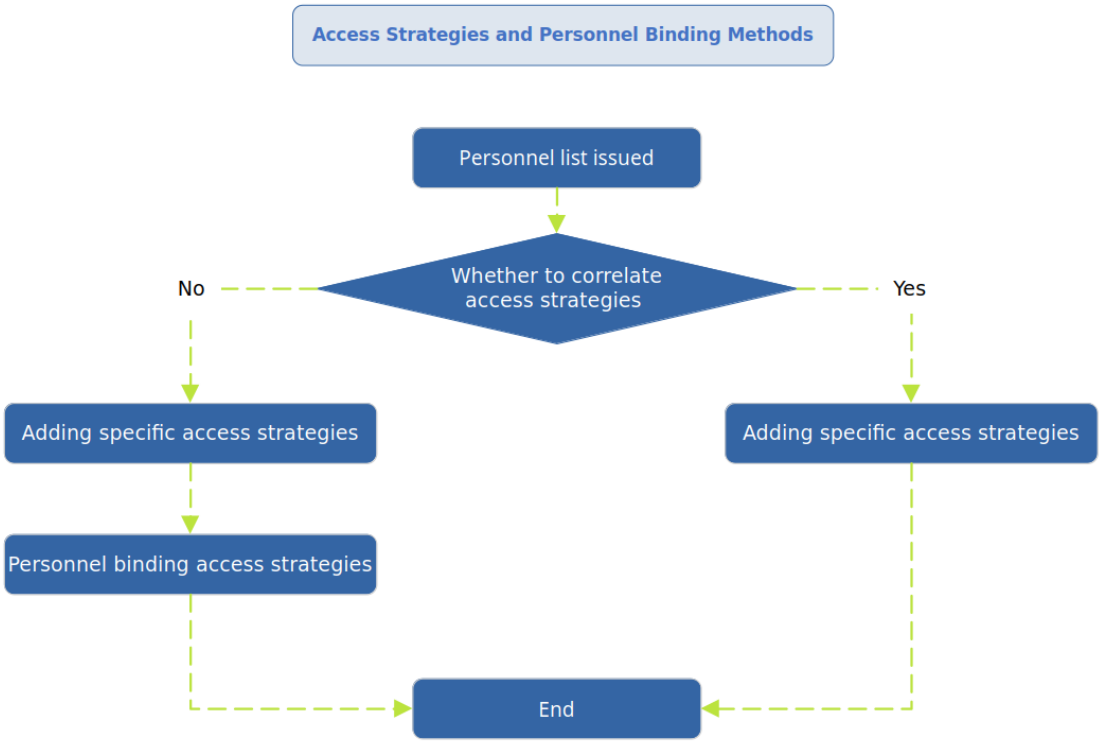
e.g. : While the previous PersonsBindStrategyID binding task is still processing, it needs to wait. The best way to handle this is to receive a successful return from this task before executing the next one.

```
{
  "messageId" : "ID:localhost-637050272518414388:79346:87:1",
  "operator" : "PersonsBindStrategyID-Ack",
  "code" : "410",
  "info" : {
    "facesluiceId" : "1888732",
    "result" : "fail",
    "detail" : "PersonsBindStrategyID is busy"
  }
}
```

5.2.1 Access Strategy

1. Each strategy is identified by a strategy ID (hereinafter referred to as `strategyID`), and each strategy consists of two parts, the regular strategy time (Monday to Sunday) and the holiday strategy time. 2.
2. Regular strategy time can be set for more than one time slot per day, for example: Monday can be set for more than one time slot
3. Holiday strategy time only supports adding one corresponding time slot, if there are more than one time slots for the same holiday, then you need to add more holidays.
4. The access strategy for the passable time period settings, if it is prohibited to pass the time, you need to take the passable time period of the opposite time, assuming that the passable time period of 9:00-18:00, if you want to prohibit the passage of time is also 9:00-18:00, issued to the face of the machine can be set for the passable time period for the 00:00-08:59 and 18:00-23:59.
5. The passable time period for the passable time period can be set. The personnel outside of the access strategy time policy will be restored to the passable state by default, assuming the expiration date of passable 2020-12-29T23:59:59, then the personnel bound to this strategy after this time will be restored to the passable state.
6. Restoring the factory operation by checking the Restore Personnel List will delete all the information of the access strategy.
7. A personnel list can be bound to more than one access strategy, and the final passable The time period is the concatenation of the time periods of multiple access strategy.
8. Try to use permanent lists for the issued list types, and then bind the access strategy, do not issue temporary lists and use the access strategy at the same time.

5.2.2 Access Strategies and Personnel Binding Methods



Here are two ways to bind a access strategy to a person, as shown above.

Mode 1: When issuing the personnel list, directly bring the corresponding associated access strategy information, and then add the corresponding access strategy information.

Mode 2: The list of personnel is issued without the corresponding associated access strategy information, then add the corresponding access strategy information, and finally bind the list of personnel to the corresponding access strategy through the personnel binding access strategy interface.

5.2.3 Access Strategies

5.2.3.1 Add/Modify Access Strategies

1.Description

Adding or modifying the time of a access strategy by means of the strategyID.

2.API Description

Items	Description
Interface Name	Add/modify access strategies
Data flow (active command side)	Platform->Device
Downlink data topic	mqtt/face/ID（Where ID refers to the device ID/SN，e.g,:1306612）
Uplink data topic	mqtt/face/ID/Ack（Where ID refers to the device ID/SN，e.g,:1306612）

3.Add/Modify Access Strategies Field Descriptions
Parameter information(Note: `Optional` is optional):

Key	Type	Values	Description
operator	String	AddAccessStrategy	Add/modify access strategies
meessageId	String		Message id, used by the platform to distinguish each message.
facesluiceId	String (optional)		The clientID for connecting to the server,device ID/SN is used by default to ensure uniqueness.
bIsInt	int	0,1	Returns whether the ack data strategy ID is a string or an int; 0-string; 1-int (default string, if this value is not passed)
info	Object		Concrete content
strategyID	int	1~N	Access strategy ID
strategyName	String (optional)		Access strategy name (64 bytes including terminator)
accessNumLimit	int		Limitations on the number of passes 0:Unlimited number of passes (default) 1:Restrictions on the number of passes
allowCnt	int (optional)		Number of passe Valid when accessNumLimit=1, required
startDate	String	YYYY-MM-DDTh h:mm:ss(2020-01-0 1T00:00:00)	Effective time of access strategy
endDate	String	YYYY-MM-DDTh h:mm:ss(2020-12-2 9T23:59:59)	Access strategy expiration time
monday	Array (optional)		Monday access strategy keywords
tuesday	Array (optional)		Tuesday access strategy keywords
wednesday	Array (optional)		Wednesday access strategy keywords
thursday	Array (optional)		Thursday access strategy keywords
friday	Array (optional)		Friday access strategy keywords

Key	Type	Values	Description
saturday	Array (optional)		Saturday access strategy keywords
sunday	Array (optional)		Sunday access strategy keywords
startTime	String	hh:mm:ss(08:00:00)	Multi-time period start time
endTime	String	hh:mm:ss(23:59:59)	Multi-time period end time
holidayInfo	Array		Holiday information keywords
holidayStartDate	String (optional)	YYYY-MM-DDTh h:mm:ss(2020-01-0 1T00:00:00)	Holiday start time
holidayEndDate	String (optional)	YYYY-MM-DDTh h:mm:ss(2020-12-2 9T23:59:59)	Holiday end time
holidayPeriod	Array		Holiday multi-time period keywords (Currently one holiday corresponds to one time period)

4.Example of Add/Modify Access Strategies

```
{
    "messageId": "ID:localhost-637050272518414388:79346:87:1",
    "operator": "AddAccessStrategy",
    "bIsInt":0,
    "info": {
        "strategyID": 1,
        "strategyName": "test1",
        "accessNumLimit": 0,
        "allowCnt": 0,
        "startDate": "2020-01-01T08:12:10",
        "endDate": "2020-12-09T08:08:12",
        "monday": [{
                "endTime": "01:09:00",
                "startTime": "00:00:00"
            },
            {
                "endTime": "03:41:00",
                "startTime": "02:02:00"
            }
```

```
        ],
        "tuesday": [{
                "endTime": "08:59:00",
                "startTime": "00:00:00"
            },
            {
                "endTime": "10:20:00",
                "startTime": "10:05:00"
            }
        ],
        "wednesday": [{
            "endTime": "02:59:00",
            "startTime": "00:00:00"
        }],
        "thursday": [{
            "endTime": "21:59:00",
            "startTime": "00:00:00"
        }],
        "friday": [{
            "endTime": "08:59:00",
            "startTime": "04:00:00"
        }],
        "saturday": [{
            "endTime": "15:43:00",
            "startTime": "01:00:00"
        }],
        "sunday": [{
            "endTime": "15:46:00",
            "startTime": "00:00:00"
        }],
        "holidayInfo": [{
            "holidayStartDate": "2020-10-10T00:00:00",
            "holidayEndDate": "2020-10-10T23:00:00",
            "holidayPeriod": [{
                "endTime": "12:43:00",
                "startTime": "01:00:00"
            }]
        }, {
            "holidayStartDate": "2020-10-10T00:00:00",
            "holidayEndDate": "2020-10-10T23:59:59",
            "holidayPeriod": [{
                "endTime": "23:45:07",
                "startTime": "17:00:00"
            }]
        }]

    }
}
```

5.Device Response Field Descriptions
Parameter information(Note:`Optional`is optional):

Key	Type	Values	Description
operator		AddAccessStrategy-Ack	Answer to add/modify access strategies
messageId	String		Message id, used by the platform to distinguish each message.
code	String		Command execution error code 200-successes,see also 10.9 Add/Modify Access Strategies Error Codes
info	Object		Concrete content
facesluiceId	String		The clientID for connecting to the server,device ID/SN is used by default to ensure uniqueness.
strategyID	String	1~N	Access strategy ID
result	String	"ok"/"fail"	Operating result
detail	String (optional)		Error message when Result is "Fail"

6.Device Response Example

```
{
    "messageId": "ID:localhost-637050272518414388:79346:87:1",
    "operator": "AddAccessStrategy-Ack",
    "code": "200",
    "info": {
        "facesluiceId": "2864575",
        "strategyID": "1",
        "result": "ok"
    }
}
```

5.2.3.2 Delete Access Strategies

1.Description

Deletion of the access strategy time is achieved through the access strategy ID (strategyID).

2.API Description

Items	Description
Interface Name	Delete access strategies
Data flow (active command side)	Platform->Device
Downlink data topic	mqtt/face/ID（Where ID refers to the device ID/SN，e.g,:1306612）
Uplink data topic	mqtt/face/ID/Ack（Where ID refers to the device ID/SN，e.g,:1306612）

3.Delete Access Strategies Field Descriptions
Parameter information(Note:`Optional`is optional):

Key	Type	Values	Description
operator	String	DelAccessStrategy	Delete access strategies
meessageId	String		Message id, used by the platform to distinguish each message.
facesluiceId	String (optional)		The clientID for connecting to the server,device ID/SN is used by default to ensure uniqueness
info	Object		Concrete content
strategyID	Array	int array，1~N	Access strategy ID

4.Example of Delete Access Strategies

```
{
    "messageId": "ID:localhost-637050272518414388:79346:87:1",
    "operator": "DelAccessStrategy",
    "info": {
        "strategyID": [1, 2]
    }
}
```

5.Device Response Field Descriptions
Parameter information(Note:`Optional`is optional):

Key	Type	Values	Description
operator		DelAccessStrategy-Ack	Answer to Delete access strategies
meessageId	String		Message id, used by the platform to distinguish each message.
code	String		Command execution error code 200-successes,see also 10.10 Delete Access Strategies Error Codes
info	Object		Concrete content
facesluiceId	String		The clientID for connecting to the server,device ID/SN is used by default to ensure uniqueness.
delAccessStrategyErrNum	String	0~N	Number of failed deletions of access strategies
delAccessStrategySucInfo	Array		Delete access strategies failure message keyword
delAccessStrategySucNum	String	0~N	Number of successful access strategies deletions
delAccessStrategySucInfo	Array		Delete access strategies success message keyword
JsonNum	String	0~N	strategyID index in Json array
result	String (optional)	"ok"/"fail"	Operating result
detail	String (optional)		Error message when Result is "Fail"

6.Device Response Example

```
{
    "messageId": "ID:localhost-637050272518414388:79346:87:1",
    "operator": "DelAccessStrategy-Ack",
    "code": "200",
    "info": {
        "facesluiceId": "2864575",
        "delAccessStrategyErrNum": "0",
        "delAccessStrategySucNum": "2",
        "delAccessStrategyErrInfo": [],
        "delAccessStrategySucInfo": [{
```

```
            "JsonNum": "0",
            "strategyID": "1"
        }, {
            "JsonNum": "1",
            "strategyID": "2"
        }],
        "result": "ok"
    }
}
```

5.2.3.3 Query All Access Strategies IDs and Names

1.Description

Queries all access strategies IDs in the device along with their names. (Maximum number of id groups to query at one time is 50)

2.API Description

Items	Description
Interface Name	Query all access strategies IDs and names
Data flow (active command side)	Platform->Device
Downlink data topic	mqtt/face/ID (Where ID refers to the device ID/SN，e.g,:1306612)
Uplink data topic	mqtt/face/ID/Ack (Where ID refers to the device ID/SN，e.g,:1306612)

3.Query All Access Strategies IDs and Names Field Descriptions
Parameter information(Note:`Optional`is optional):

Key	Type	Values	Description
operator	String	QueryAllStrategyID	Query all access strategies IDs and names
meessageId	String		Message id, used by the platform to distinguish each message.
facesluiceId	String (optional)		The clientID for connecting to the server,device ID/SN is used by default to ensure uniqueness
info	Object		Concrete content

4.Example of Query All Access Strategies IDs and Names

```
{
    "messageId": "ID:localhost-637050272518414388:79346:87:1",
    "operator": "QueryAllStrategyID",
    "info": {}
}
```

5.Device Response Field Descriptions

Parameter information(Note:`Optional`is optional):

Key	Type	Values	Description
operator		QueryAllStrategyID-Ack	Answer to query all access strategies IDs and names
meessageId	String		Message id, used by the platform to distinguish each message.
code	String		Command execution error code 200-successes,see also 10.11 Query All Access Strategies Error Codes
info	Object		Concrete content
facesluiceId	String		The clientID for connecting to the server,device ID/SN is used by default to ensure uniqueness
strategyNum	String	0~N	Number of access strategies
StrategyInfo	Array		Access strategy information keyword
StrateID	String	1~N	Access strategy ID
strategyName	String		Access strategy name
result	String (optional)	"ok"/"fail"	Operating result
detail	String (optional)		Error message when Result is "Fail"

6.Device Response Example

```
{
    "messageId": "ID:localhost-637050272518414388:79346:87:1",
    "operator": "QueryAllStrategyID-Ack",
    "code": "200",
    "info": {
        "facesluiceId": "2864575",
        "strategyNum": "1",
        "strategyInfo": [{
```

```
            "strategyID": "1",
            "strategyName": "test1"
        }],
        "result": "ok"
    }
}
```

5.2.3.4 Query Access Strategy Details

1.Description
Query access strategy details by access strategy ID.

2.API Description

Items	Description
Interface Name	Query access strategy Details
Data flow (active command side)	Platform->Device
Downlink data topic	mqtt/face/ID（Where ID refers to the device ID/SN，e.g,:1306612）
Uplink data topic	mqtt/face/ID/Ack（Where ID refers to the device ID/SN，e.g,:1306612）

3.Query Access Strategy Details Field Descriptions
Parameter information(Note:`Optional`is optional):

Key	Type	Values	Description
operator	String	AccessStrategyIDQueryStrategy	Query Access Strategy Details
meessageId	String		Message id, used by the platform to distinguish each message.
facesluiceId	String (optional)		The clientID for connecting to the server,device ID/SN is used by default to ensure uniqueness
info	Object		Concrete content
strategyID	Array	int array，1~N	Access strategy ID

4.Example of Query Access Strategy Details

```
{
    "messageId": "ID:localhost-637050272518414388:79346:87:1",
    "operator": "AccessStrategyIDQueryStrategy",
    "info": {
        "strategyID": [1, 2, 3]
    }
}
```

5.Device Response Field Descriptions

Parameter information(Note: `Optional` is optional):

Key	Type	Values	Description
operator		AccessStrategyIDQueryStrategy-Ack	Answer to query access strategy details
meessageId	String		Message id, used by the platform to distinguish each message.
code	String		Command execution error code 200-successes,see also 10.13 Query Access Strategy Details Error Codes
info	Object		Concrete content
facesluiceId	String		The clientID for connecting to the server,device ID/SN is used by default to ensure uniqueness
strategyNum	String	0~N	Number of access strategies
StrategyInfo	Array		Access strategy information keyword
strategyID	int	1~N	Access strategy ID
strategyName	String		Access strategy name
accessNumLimit	int		Limitations on the number of passes 0:Unlimited number of passes (default) 1:Restrictions on the number of passes
allowCnt	int (optional)		Number of passe Valid when accessNumLimit=1, required
cTimeDate	String	YYYY-M M-DDThh:mm:ss(202 0-01-01T00 :00:00)	Access strategy creation time
startDate	String	YYYY-M M-DDThh:mm:ss(2020-01-01T00 :00:00)	Effective time of access strategy
endDate	String	YYYY-M M-DDThh:mm:ss(2020-12-29T23 :59:59)	Access strategy expiration time

Key	Type	Values	Description
monday	Array (optional)		Monday access strategy keywords
tuesday	Array (optional)		Tuesday access strategy keywords
wednesday	Array (optional)		Wednesday access strategy keywords
thursday	Array (optional)		Thursday access strategy keywords
friday	Array (optional)		Friday access strategy keywords
saturday	Array (optional)		Saturday access strategy keywords
sunday	Array (optional)		Sunday access strategy keywords
everyday	Array (optional)		All day access strategy keywords
startTime	String	hh:mm:ss(08:00:00)	Multi-time period start time
endTime	String	hh:mm:ss(23:59:59)	Multi-time period end time
holidayInfo	Array		Holiday information keywords
holidayStartDate	String (optional)	YYYY-MM-DDThh:mm:ss(2020-10-1T00:00:00)	Holiday start time
holidayEndDate	String (optional)	YYYY-MM-DDThh:mm:ss(2020-10-1T23:59:59)	Holiday end time
holidayPeriod	String		Holiday multi-time period keywords (Currently one holiday corresponds to one time period)

Key	Type	Values	Description
result	String (optional)	"ok"/"fail"	Operating result
detail	String (optional)		Error message when Result is "Fail"

6.Device Response Example


```
{
    "messageId": "ID:localhost-637050272518414388:79346:87:1",
    "operator": "AccessStrategyIDQueryStrategy-Ack",
    "code": "200",
    "info": {
        "facesluiceId": "1924537",
        "strategyNum": 3,
        "strategyInfo": [
            {
                "strategyID": 1,
                "strategyName": "strategyName_1",
                "accessNumLimit": 0,
                "allowCnt": 0,
                "startDate": "2023-01-01T00:00:00",
                "endDate": "2037-01-01T23:59:59",
                "monday": [
                    {
                        "startTime": "07:00:01",
                        "endTime": "09:00:01"
                    },
                    {
                        "startTime": "18:00:01",
                        "endTime": "20:00:01"
                    }
                ],
                "tuesday": [
                    {
                        "startTime": "07:00:02",
                        "endTime": "09:00:02"
                    },
                    {
                        "startTime": "18:00:02",
                        "endTime": "20:00:02"
                    }
                ],
                "wednesday": [
```

```
                    {
                        "startTime": "07:00:03",
                        "endTime": "09:00:03"
                    },
                    {
                        "startTime": "18:00:03",
                        "endTime": "20:00:03"
                    }
                ],
                "thursday": [
                    {
                        "startTime": "07:00:04",
                        "endTime": "09:00:04"
                    },
                    {
                        "startTime": "18:00:04",
                        "endTime": "20:00:04"
                    }
                ],
                "friday": [
                    {
                        "startTime": "07:00:05",
                        "endTime": "09:00:05"
                    },
                    {
                        "startTime": "18:00:05",
                        "endTime": "20:00:05"
                    }
                ],
                "saturday": [
                    {
                        "startTime": "07:00:06",
                        "endTime": "09:00:06"
                    },
                    {
                        "startTime": "18:00:06",
                        "endTime": "20:00:06"
                    }
                ],
                "sunday": [
                    {
                        "startTime": "07:00:07",
                        "endTime": "09:00:07"
                    },
                    {
                        "startTime": "18:00:07",
                        "endTime": "20:00:07"
                    }
                ],
                "holidayInfo": [
```

```
                    {
                        "holidayStartDate": "1999-01-01T00:00:00",
                        "holidayEndDate": "2037-12-31T23:59:59",
                        "holidayPeriod": [
                            {
                                "startTime": "00:00:00",
                                "endTime": "23:59:59"
                            }
                        ]
                    },
                    {
                        "holidayStartDate": "2024-11-19T17:50:59",
                        "holidayEndDate": "2037-12-31T23:59:59",
                        "holidayPeriod": [
                            {
                                "startTime": "00:00:00",
                                "endTime": "23:59:59"
                            }
                        ]
                    }
                ]
            },
            {
                "strategyID": 175,
                "strategyName": "4",
                "accessNumLimit": 0,
                "allowCnt": 0,
                "startDate": "2023-01-01T00:00:24",
                "endDate": "2037-01-01T23:59:59",
                "everyday": [
                    {
                        "startTime": "00:00:00",
                        "endTime": "23:59:59"
                    }
                ]
            }
        ],
        "result": "ok"
    }
}
```


5.2.3.5 Query All Associated Users by Access Strategy ID

1.Description

Querying all users associated with the access strategy is accomplished by using the access strategy ID (strategyID).

2.API Description

Items	Description
Interface Name	Query all associated users
Data flow (active command side)	Platform->Device
Downlink data topic	mqtt/face/ID（Where ID refers to the device ID/SN，e.g,:1306612）
Uplink data topic	mqtt/face/ID/Ack（Where ID refers to the device ID/SN，e.g,:1306612）

3.Query All Associated Users by Access Strategy ID Field Descriptions
Parameter information(Note:`Optional`is optional):

Key	Type	Values	Description
operator	String	AccessStrategyIDQueryPersons	Query all associated users by access strategy ID
meessageId	String		Message id, used by the platform to distinguish each message.
facesluiceId	String (optional)		The clientID for connecting to the server,device ID/SN is used by default to ensure uniqueness
info	Object		Concrete content
strategyID	int	1~N	Access strategy ID

4.Example of Query All Associated Users by Access Strategy ID

```
{
    "messageId": "ID:localhost-637050272518414388:79346:87:1",
    "operator": "AccessStrategyIDQueryPersons",
    "info": {
        "strategyID": 1
    }
}
```

5.Device Response Field Descriptions
Parameter information(Note:`Optional`is optional):

Key	Type	Values	Description
operator		AccessStrategyIDQueryPersons-Ack	Answer to query all associated users by access strategy ID
messageId	String		Message id, used by the platform to distinguish each message.
code	String		Command execution error code 200-successes,see also 10.12 Query All Associated Users by Access Strategy ID Error Codes
info	Object		Concrete content
facesluiceId	String		The clientID for connecting to the server,device ID/SN is used by default to ensure uniqueness
personNum	String	0-N	Number of people matching the access strategy
personInfo	Array		The json array of the queried person information
customId	String	Length 48 characters (including terminator)	Platform generated id, uniquely identifies different personnel.
name	String	Reserved	Reserve
result	String	"ok"/"fail"	Operating result
detail	String (optional)		Error message when Result is "Fail"

6.Device Response Example

```
{
    "messageId": "ID:localhost-637050272518414388:79346:87:1",
    "operator": "AccessStrategyIDQueryPersons-Ack",
    "code": "200",
    "info": {
        "facesluiceId": "2864575",
        "personNum": "1",
        "personInfo": [{
            "customId": "063c81e0fce184c696cdb7e049230f5d",
            "name": "Reserved"
        }],
        "result": "ok"
    }
```

```
14  }
```

5.2.3.6 Personnel Binding Access Strategy

1.Description

This interface is mainly to realize that when issuing personnel, you can not bind the personnel first, but only issue the personnel list information, and after the list is issued, you can bind the corresponding access strategy. However, it should be noted that the binding relationship between the access strategy information in the list distribution and the personnel list in this interface is **a full override operation**. For example, the first time the personnel is bound to the access strategy group 1 and group 2, the second time the personnel is bound to the access strategy group 1, then the final personnel is bound to the access strategy group 1 only.（Maximum of 64 strategy id groups for single person binding）

2.API Description

Items	Description
Interface Name	Personnel binding access strategy
Data flow (active command side)	Platform->Device
Downlink data topic	mqtt/face/ID（Where ID refers to the device ID/SN，e.g,:1306612）
Uplink data topic	mqtt/face/ID/Ack（Where ID refers to the device ID/SN，e.g,:1306612）

3.Personnel Binding Access Strategy Field Descriptions
Parameter information(Note:`Optional`is optional):

Key	Type	Values	Description
operator	String	PersonsBindStrategyID	Personnel binding access strategy
meessageId	String		Message id, used by the platform to distinguish each message.
facesluiceId	String (optional)		The clientID for connecting to the server,device ID/SN is used by default to ensure uniqueness
info	Object		Concrete content
personsInfo	Array		Person and strategy information keywords
customId	String	Length 48 characters (including terminator)	Platform generated id, uniquely identifies different personnel.
strategyID	Array	1~N	Access strategy ID

4.Example of Personnel Binding Access Strategy

```
{
    "messageId": "ID:localhost-637050272518414388:79346:87:1",
    "operator": "PersonsBindStrategyID",
    "info": {
        "personsInfo": [{
                "customId": "063c81e0fce184c696cdb7e049230f5d",
                "strategyID": [1, 2]
            },
            {
                "customId": "063c81e0fce184c696cdb7e049230f5e",
                "strategyID": [1, 2, 3, 4]
            }
        ]
    }
}
```

5.Device Response Field Descriptions

Parameter information(Note: `Optional` is optional):

Key	Type	Values	Description
operator		PersonsBindStrategyID-Ack	Answer to personnel binding access strategy
messageId	String		Message id, used by the platform to distinguish each message.
code	String		Command execution error code 200-successes,see also 10.14 Personnel Binding Access Strategy Error Codes
info	Object		Concrete content
facesluiceId	String		The clientID for connecting to the server,device ID/SN is used by default to ensure uniqueness
bindErrNum	String	0~N	Number of failed personnel bind access strategy
bindErrInfo	Array		Personnel binding access strategy failure message keyword
customId	String	Length 48 characters (including terminator)	Platform generated id, uniquely identifies different personnel.
errcode	String	1-N	Error Code
bindSucNum	String	0~N	Number of successful personnel binding access strategy
bindSucInfo	Array		Personnel binding access strategy success message keyword
strategyID	Array	1~N	Access strategy ID
result	String	"ok"/"fail"	Operating result
detail	String (optional)		Error message when Result is "Fail"

6.Device Response Example

```
{
    "messageId": "ID:localhost-637050272518414388:79346:87:1",
    "operator": "PersonsBindStrategyID-Ack",
    "code": "200",
    "info": {
        "facesluiceId": "2864575",
        "bindErrNum": "1",
```

```
        "bindSucNum": "1",
        "bindErrInfo": [{
            "customId": "063c81e0fce184c696cdb7e049230f5e",
            "errcode": "461"
        }],
        "bindSucInfo": [{
            "customId": "063c81e0fce184c696cdb7e049230f5d",
            "strategyID": [1, 2]
        }],
        "result": "ok"
    }
}
```

5.2.3.7 Personnel Unbinding Access Strategy

1.Description

The Person Unbinding Access Strategy mainly implements the unbinding of a person from a access strategy, if this person is itself on a permanent list, then this person will become **passable**.

2.API Description

Items	Description
Interface Name	Personnel unbinding access strategy
Data flow (active command side)	Platform->Device
Downlink data topic	mqtt/face/ID（Where ID refers to the device ID/SN，e.g,:1306612）
Uplink data topic	mqtt/face/ID/Ack（Where ID refers to the device ID/SN，e.g,:1306612）

3.Personnel Unbinding Access Strategy Field Descriptions
Parameter information(Note:`Optional`is optional):

Key	Type	Values	Description
operator	String	PersonsUnbindStrategyID	Personnel unbinding access strategy
meessageId	String		Message id, used by the platform to distinguish each message.
facesluiceId	String (optional)		The clientID for connecting to the server,device ID/SN is used by default to ensure uniqueness
info	Object		Concrete content
personsInfo	Array		Person and strategy information keywords
customId	String	Length 48 characters (including terminator)	Platform generated id, uniquely identifies different personnel.
strategyID	Array	1-N	Access strategy ID

4.Example of Personnel Unbinding Access Strategy

```
{
    "messageId": "ID:localhost-637050272518414388:79346:87:1",
    "operator": "PersonsUnbindStrategyID",
    "info": {
        "personsInfo": [{
                "customId": "063c81e0fce184c696cdb7e049230f5d",
                "strategyID": [1, 2, 3]
            },
            {
                "customId": "063c81e0fce184c696cdb7e049230f5e",
                "strategyID": [1, 2, 3, 4]
            }
            ]
    }
}
```

5.Device Response Field Descriptions
Parameter information(Note: `Optional` is optional):

Key	Type	Values	Description
operator		PersonsUnbindStrategyID-Ack	Answer to personnel unbinding access strategy
messageId	String		Message id, used by the platform to distinguish each message.
code	String		Command execution error code 200-successes,see also 10.15 Personnel Unbinding Access Strategy Error Codes
info	Object		Concrete content
facesluiceId	String		he clientID for connecting to the server,device ID/SN is used by default to ensure uniqueness.
unBindErrNum	String	0~N	Number of failed personnel unbind access strategy
unBindErrInfo	Array		Personnel unbinding access strategy failure message keyword
customId	String	Length 48 characters (including terminator)	Platform generated id, uniquely identifies different personnel.
errcode	String	1-N	Error code
unBindSucNum	String	0~N	Number of successful personnel unbinding access strategy
unBindSucInfo	Array		Personnel binding access strategy success message keyword
strategyID	Array	1-N	Access strategy ID
result	String	"ok"/"fail"	Operating result
detail	String (optional)		Error message when Result is "Fail"

6.Device Response Example

```
{
    "messageId": "ID:localhost-637050272518414388:79346:87:1",
    "operator": "PersonsUnbindStrategyID-Ack",
    "code": "200",
    "info": {
        "facesluiceId": "2864575",
```

```
        "unBindErrNum": "1",
        "unBindSucNum": "1",
        "unBindErrInfo": [{
            "customId": "063c81e0fce184c696cdb7e049230f5e",
            "errcode": "461"
        }],
        "unBindSucInfo": [{
            "customId": "063c81e0fce184c696cdb7e049230f5d",
            "strategyID": [1, 2, 3]
        }],
        "result": "ok"
    }
}
```

5.2.3.8 Batch Downstream Access Strategy

1.Description

Enable batch add or modify access strategy time.（**Suggested bulk distribution of 100 items at a time**）

2.API Description

Items	Description
Interface Name	Batch sownstream access strategy
Data flow (active command side)	Platform->Device
Downlink data topic	mqtt/face/ID（Where ID refers to the device ID/SN，e.g,:1306612）
Uplink data topic	mqtt/face/ID/Ack（Where ID refers to the device ID/SN，e.g,:1306612）

3.Batch Downstream Access Strategy Field Descriptions
Parameter information(Note:`Optional`is optional):

Key	Type	Values	Description
operator	String	AddAccessStrategys	Batch downstream access strategy
meessageId	String		Message id, used by the platform to distinguish each message.
facesluiceId	String (optional)		The clientID for connecting to the server,device ID/SN is used by default to ensure uniqueness.
bIsInt	int	0,1	Returns whether the ack data strategy ID is a string or an int; 0-string; 1-int (default string, if this value is not passed)
info	Object		Concrete content
strategyID	int	1~N	Access strategy ID
strategyName	String (optional)		Access strategy name (64 bytes including terminator)
accessNumLimit	int		Limitations on the number of passes 0:Unlimited number of passes (default) 1:Restrictions on the number of passes
allowCnt	int (optional)		Number of passe Valid when accessNumLimit=1, required
startDate	String	YYYY-MM-DDTh h:mm:ss(2020-01-0 1T00:00:00)	Effective time of access strategy
endDate	String	YYYY-MM-DDTh h:mm:ss(2020-12-2 9T23:59:59)	Access strategy expiration time
monday	Array (optional)		Monday access strategy keywords
tuesday	Array (optional)		Tuesday access strategy keywords
wednesday	Array (optional)		Wednesday access strategy keywords
thursday	Array (optional)		Thursday access strategy keywords
friday	Array (optional)		Friday access strategy keywords

Key	Type	Values	Description
saturday	Array (optional)		Saturday access strategy keywords
sunday	Array (optional)		Sunday access strategy keywords
startTime	String	hh:mm:ss(08:00:00)	Multi-time period start time
endTime	String	hh:mm:ss(23:59:59)	Multi-time period end time
holidayInfo	Array		Holiday information keywords
holidayStartDate	String (optional)	YYYY-MM-DDTh h:mm:ss(2020-01-0 1T00:00:00)	Holiday start time
holidayEndDate	String (optional)	YYYY-MM-DDTh h:mm:ss(2020-12-2 9T23:59:59)	Holiday end time
holidayPeriod	Array		Holiday multi-time period keywords (Currently one holiday corresponds to one time period)

4.Example of Batch Downstream Access Strategy

```
{
    "messageId": "ID:localhost-637050272518414388:79346:87:1",
    "operator":"AddAccessStrategys",
    "info":[
            {
                "monday":[{"startTime":"15:00:00","endTime":"21:00:00"},{"startTime":"15:00:00","endTime":"20:00:00"}],
                "tuesday":[{"startTime":"05:00:00","endTime":"07:00:00"},{"startTime":"9:00:00","endTime":"10:00:00"}],
                "wednesday":[{"startTime":"12:00:00","endTime":"16:00:00"},{"startTime":"16:01:00","endTime":"23:00:00"}],
                "thursday":[{"startTime":"8:00:00","endTime":"20:00:00"},{"startTime":"08:30:00","endTime":"20:00:00"}],
                "friday":[{"startTime":"06:00:00","endTime":"23:00:00"},{"startTime":"13:00:00","endTime":"14:00:00"}],
                "saturday":[{"startTime":"01:00:00","endTime":"23:00:00"},{"startTime":"02:00:00","endTime":"22:00:00"}],
                "sunday":[{"startTime":"01:00:00","endTime":"23:00:00"}],
                "holidayInfo":[
                    {"holidayStartDate":"2024-10-10T00:00:00","holidayEndDate":"2024-10-11T23:00:00","holidayPeriod":[{"startTime":"01:00:00","endTime":"12:43:00"}]},
```

```
                    {"holidayStartDate":"2023-10-
10T00:00:00","holidayEndDate":"2023-10-10T23:59:59","holidayPeriod":
[{"startTime":"09:00:00","endTime":"23:45:00"}]}
                ],
                "strategyID":75,
                "strategyName":"xxx leave passage time",
                "startDate":"2023-01-01T00:00:00",
                "endDate":"2099-01-01T23:59:59"
            },
            {
                "monday":[{"startTime":"01:00:00","endTime":"23:00:00"}],
                "tuesday":[],"wednesday":[],"thursday":[],"friday":[],"saturday":
[],"sunday":[],
                "strategyID":175,"strategyName":"4",
                "startDate":"2023-01-01T00:00:500","endDate":"2099-01-01T23:59:59"
            }
        ]
}
```

5.Device Response Field Descriptions

Parameter information(Note: `Optional` is optional):

Key	Type	Values	Description
operator		AddAccessStrategys-Ack	Answer to batch downstream access strategy
messageId	String		Message id, used by the platform to distinguish each message.
code	String		Command execution error code 200-successes,see also 10.9 Add/Modify Access Strategies Error Codes
info	Object		Concrete content
facesluiceId	String		The clientID for connecting to the server,device ID/SN is used by default to ensure uniqueness.
strategyID	String	1~N	Access strategy ID
AddErrNum	int		Number of failed add access strategies
AddSucNum	int		Number of successful add access strategies
AddErrInfo	Array		Add access strategies failure id data
AddSucInfo	Array		Add access strategies success id data
Result	String	"OK"/"fail"	Operating result
detail	String (optional)		Error message when Result is "Fail"

6.Device Response Example

```
{
    "messageId": "ID:localhost-637050272518414388:79346:87:1",
    "operator": "AddAccessStrategys-Ack",
    "code": 200,
    "info": {
        "facesluiceId": "1924537",
        "AddErrNum": 0,
        "AddSucNum": 2,
        "AddErrInfo": [
        ],
        "AddSucInfo": [
            75,
            175
        ],
        "Result": "OK"
    }
}
```

5.3 Photo Ads

The version of Face Machine software with (AD) in the version number supports polling time to play the advertisement picture, the picture format supports png/jpg/bmp format picture, and currently supports up to 10 advertisement pictures. Default advertisement effect: Stranger advertisement picture does not disappear, the UI of the face machine will show the main video picture when the personnel passes the state, continue to verify the reset time, and switch back to the advertisement picture.The pixel size of the advertisement picture is 600*1024 for 7-inch model and 800*1280 for 8-inch model. Other sizes of models do not support the advertisement function for the time being.

5.3.1 Upload Ads

1.Description
Add or modify advertisements.

2.API Description

Items	Description
Interface Name	Add or modify advertisements.
Data flow (active command side)	Platform->Device
Downlink data topic	mqtt/face/ID（Where ID refers to the device ID/SN，e.g,:1306612）
Uplink data topic	mqtt/face/ID/Ack（Where ID refers to the device ID/SN，e.g,:1306612）

3.Upload Ads Field Descriptions
Parameter information(Note:`Optional`is optional):

Key	Type	Values	Description
operator	String	EditAD	Add or modify advertisements.
messageId	String		Message id, used by the platform to distinguish each message.
info	Object		Concrete content
facesluiceId	String		The clientID for connecting to the server,device ID/SN is used by default to ensure uniqueness
adslot	int	0~9	Advertising Slot
path	String		Image download path
adid	String (optional)	Operation ID	Reserve
polltime	int (optional)	Default 10 seconds	Length of time each advertisement is polled.

4.Example of Upload Ads

```
{
    "operator": "EditAD",
    "messageId": "ID:localhost-637050888589478689:44009:24:40",
    "info": {
        "adid": "",
        "path": "http://xxxxxxxx.jpg",
        "adslot": 0,
        "polltime": 10
    }
}
```

5.Device Response Field Descriptions

Parameter information(Note: `Optional` is optional):

Key	Type	Values	Description
operator		EditAD-Ack	Answer to add or modify advertisements.
messageId	String		Message id, used by the platform to distinguish each message.
code	String		Command execution error code 200-successes,see also 10.16 Upload Ads Error Codes
info	Object		Concrete content
facesluiceId	String		The clientID for connecting to the server,device ID/SN is used by default to ensure uniqueness
adid	String	Operation ID	Reserve
result	String	"ok"/"fail"	Operating result
detail	String (optional)		Error message when Result is "Fail"

6.Device Response Example

```
{
    "messageId": "ID:localhost-637050888589478689:44009:24:40",
    "operator": "EditAD-Ack",
    "code": "200",
    "info": {
        "facesluiceId": "1306612",
        "adid": "",
        "result": "ok"
    }
}
```

5.3.2 Drop Ads

1.Description

Delete the advertisement picture of the existing advertisement slot of the device, if the deleted advertisement is in the middle slot, it is recommended to re-sort the advertisement and send it out, and it is not recommended to leave the middle slot empty.

2.API Description

Items	Description
Interface Name	Drop Ads
Downlink data topic	mqtt/face/ID（Where ID refers to the device ID/SN，e.g,:1306612）
Uplink data topic	mqtt/face/ID/Ack（Where ID refers to the device ID/SN，e.g,:1306612）

3.Drop Ads Field Descriptions
Parameter information(Note:`Optional`is optional):

Key	Type	Values	Description
operator		DelAD	Drop Ads
messageId	String		Message id, used by the platform to distinguish each message.
info	Object		Concrete content
facesluiceId	String(optional)		The clientID for connecting to the server,device ID/SN is used by default to ensure uniqueness.
adid	String(optional)	Operation ID	Reserve
adslot	int	0~9	Advertising Slot

4.Example of Drop Ads

```
{
    "operator": "DelAD",
    "messageId": "ID:localhost-637050888589478689:44009:24:50",
    "info": {
        "adid": "",
        "adslot": 0
    }
}
```

5.Device Response Field Descriptions
Parameter information(Note:`Optional`is optional):

Key	Type	Values	Description
operator		DelAD-Ack	Answer to drop ads
messageId	String		Message id, used by the platform to distinguish each message.
code	String		Command execution error code 200-successes,see also 10.17 Drop Ads Error Codes
info	Object		Concrete content
facesluiceId	String		The clientID for connecting to the server,device ID/SN is used by default to ensure uniqueness
adid	String (optional)	Operation ID	Reserve
result	String		Operating result
detail	String (optional)		Error message when Result is "Fail"

6.Device Response Example

```
{
    "messageId": "ID:localhost-637050888589478689:44009:24:50",
    "operator": "DelAD-Ack",
    "code": "200",
    "info": {
        "facesluiceId": "1306612",
        "adid": "",
        "result": "ok"
    }
}
```

5.4 QR Code Display

1.Description

The QR code display part mainly involves displaying the QR code in the UI interface of the face machine. For machines with different screen sizes, you need to calculate the starting position and the image size to avoid out-of-bounds. If the image is sent as base64 data, you need to specify the original real image format of the image, otherwise the image data can not be displayed normally.There are two ways to display the QR code:

1. Send the raw data of the QR code, and the device will generate the QR code image;
2. Send the base64 data of the QR code image, and the device will directly display the image data.

2.API Description

Items	Description
Interface Name	QR code display
Data flow (active command side)	Platform->Device
Downlink data topic	mqtt/face/ID（Where ID refers to the device ID/SN，e.g,:1306612）
Uplink data topic	mqtt/face/ID/Ack（Where ID refers to the device ID/SN，e.g,:1306612）

3.QR Code Display Field Descriptions
Parameter information(Note: `Optional` is optional):

Key	Type	Values	Description
operator	String	ShowQRCode	QR code display
messageId	String		Message id, used by the platform to distinguish each message.
info	Object		Concrete content
facesluiceId	String		The clientID for connecting to the server,device ID/SN is used by default to ensure uniqueness
ImageType	int (optional)		Display Image Type 1:String of QR code images to be displayed 2:The format of the image to be displayed is png 3:The format of the image to be displayed is jpg 4:The format of the image to be displayed is bmp Default 1:String of QR code images to be displayed.
AbsX	int	10~Maximum value for different UI heights Default:10	Display image based on UI start position X-value
AbsY	int	110~Maximum value for different UI widths Default:10	Display image based on UI start position Y-value
ImageW	int	10~400(Default:200)	Display image width
ImageH	int	10~400(Default:200)	Display image height
QRCodeData	String	32KBytes(including terminator)	String data for the QR code image to be encoded when ImageType = 1 BASE64 data for the png image to be displayed when ImageType = 2 BASE64 data for the jpg image to be displayed when ImageType = 3 BASE64 data for the bmp image to be displayed when ImageType = 4
ShowStatus	int (optional)	0-1 (Default 1)	Cancel QR code image display. 0:Cancel display

4.Example of QR Code Display

4.1 QR code data is the original data

```
{
    "messageId": "XXXXXXXXXXXXXXXXXXXXXXXXXXXXXX",
    "operator": "ShowQRCode",
    "info": {
        "ImageType": 1,
        "AbsX": 10,
        "AbsY": 10,
        "ImageW": 200,
        "ImageH": 200,
        "QRCodeData": "A1B2C3D4E5r6t7y8u9Pn"
        }
}
```

4.2 QR code data as base64 data of png format image

```
{
    "messageId": "XXXXXXXXXXXXXXXXXXXXXXXXXXXXXX",
    "operator": "ShowQRCode",
    "info": {
        "ImageType": 2,
        "AbsX": 10,
        "AbsY": 10,
        "ImageW": 200,
        "ImageH": 200,
        "QRCodeData":
"*Data
omitted here*/"
    }
}
```

5.Device Response Field Descriptions

Parameter information(Note:`Optional`is optional):

Key	Type	Values	Description
operator		ShowQRCode-Ack	Answer to QR code display
messageId	String		Message id, used by the platform to distinguish each message.
code	String		Command execution error code 200-successes,see also 10.18 QR Code Display Error Codes
info	Object		Concrete content
facesluiceId	String		The clientID for connecting to the server,device ID/SN is used by default to ensure uniqueness
result	String	“ok”/“fail”	Operating result
detail	String (optional)		Error message when Result is "Fail"

6.Device Response Example

```
{
    "messageId": "XXXXXXXXXXXXXXXXXXXXXXXXXXXXXX",
    "operator": "ShowQRCode-Ack",
    "code": "200",
    "info": {
        "facesluiceId": "1333206",
        "result": "ok"
    }
}
```

5.5 Manual Snapshot

1.Description

Call the interface to get a capture scene image of the device, this interface is frequently called will likely have an impact on the recognition of the subject thread.

⚠By default, the stranger capture/authentication (control) logs reported by the device is a small image of a human face, and the face image of the stranger capture/authentication (control) logs of some models devices can be set to be reported as a capture scene image.

2.API Description

Items	Description
Interface Name	Manual snapshot
Data flow (active command side)	Platform->Device
Downlink data topic	mqtt/face/ID（Where ID refers to the device ID/SN，e.g,:1306612）
Uplink data topic	mqtt/face/ID/Ack（Where ID refers to the device ID/SN，e.g,:1306612）

3.Manual Snapshot Field Descriptions
Parameter information(Note:`Optional`is optional):

Key	Type	Values	Description
operator	String	GetSceneSnap	Manual snapshot
messageId	String		Message id, used by the platform to distinguish each message.
info	Object		Concrete content
facesluiceId	String		The clientID for connecting to the server,device ID/SN is used by default to ensure uniqueness
ImgType	int	1~2 （Default：2）	The data type of the returned captured image. 1: bmp 2: jpeg
ImgQuality	int (optional)	55~100 (Default:55)	Required for ImgType=2, compressed image quality

4.Example of Manual Snapshot

```
{
    "operator": "GetSceneSnap",
    "messageId": "ID:localhost-637050888589478689:44009:24:40",
    "info": {
        "facesluiceId": "1379743",
        "ImgType": 2,
        "ImgQuality": 55
    }
}
```

5.Device Response Field Descriptions
Parameter information(Note:`Optional`is optional):

Key	Type	Values	Description
operator		GetSceneSnap-Ack	Manual snapshot
messageId	String		Message id, used by the platform to distinguish each message.
code	String		Command execution error code 200-successes,see also 10.19 Manual Snapshot Error Codes
info	Object		Concrete content
facesluiceId	String		The clientID for connecting to the server,device ID/SN is used by default to ensure uniqueness.
ImgType	String	1~2（Default：2）	The data type of the returned captured image. 1: bmp, 2: jpeg
ImgData	String		Returns the Base64 data of the captured image
result	String	“ok”/“fail”	Operating result
detail	String (optional)		Error message when Result is "Fail"

6.Device Response Example

```
{
    "messageId": "ID:localhost-637050888589478689:44009:24:40",
    "operator": "GetSceneSnap-Ack",
    "code": "200",
    "info": {
        "facesluiceId": "1379743",
        "ImgType": "2",
        "ImgData": "*Data omitted here*/caVY3PMltHu/vKNp/MVg6Jq940svm3ltqFovCvGQJBh1BpVaJcmNwmeuRnNRNpRbZnVkowblsf/9k=",
        "result": "ok"
    }
}
```

5.6 Photo Services

5.6.1 Image Similarity Comparison

1.Description

This interface provides for comparing the similarity of two images.

2.API Description

Items	Description
Interface Name	Image similarity comparison
Data flow (active command side)	Platform->Device
Downlink data topic	mqtt/face/ID（Where ID refers to the device ID/SN，e.g,:1306612）
Uplink data topic	mqtt/face/ID/Ack（Where ID refers to the device ID/SN，e.g,:1306612）

3.Image Similarity Comparison Field Descriptions
Parameter information(Note: `Optional` is optional):

Key	Type	Values	Description
operator	String	GetPictureSimilarity	Image similarity comparison
messageId	String		Message id, used by the platform to distinguish each message.
info	Object		Concrete content
facesluiceId	String		The clientID for connecting to the server,device ID/SN is used by default to ensure uniqueness
picinfo1	String		Base64 encoded data of face image (no more than 1M)
picinfo2	String		Base64 encoded data of face image (no more than 1M)

4.Example of Image Similarity Comparison

```
{
    "operator": "GetPictureSimilarity",
    "messageId": "ID:localhost-637050888589478689:44009:24:40",
    "info": {
        "facesluiceId": "1379743",
        "picinfo1": "*Data omitted here*/caowl/9k=",
        "picinfo2": "*Data omitted here*/wblsf/9k="
    }
}
```

5.Device Response Field Descriptions

Parameter information(Note: `Optional` is optional):

Key	Type	Values	Description
operator		GetPictureSimilarity-Ack	Answer to image similarity comparison
messageId	String		Message id, used by the platform to distinguish each message.
code	String		Command execution error code 200-successes,see also 10.20 Image Similarity Comparison Error Codes
info	Object		Concrete content
facesluiceId	String		The clientID for connecting to the server,device ID/SN is used by default to ensure uniqueness
Similarity	String		Face Image Similarity.
result	String	“ok”/“fail”	Operating result
detail	String (optional)		Error message when Result is "Fail"

6.Device Response Example

```
{
    "messageId": "ID:localhost-637050888589478689:44009:24:40",
    "operator": "GetPictureSimilarity-Ack",
    "code": "200",
    "info": {
        "facesluiceId": "1379743",
        "Similarity": "94.0",
        "result": "ok"
    }
}
```

5.6.2 Search Local Face Database by Image

1.Description

The interface provides the comparison with the picture and the local face database of the face machine to get the corresponding results.

2.API Description

Items	Description
Interface Name	Search local face database by image
Data flow (active command side)	Platform->Device
Downlink data topic	mqtt/face/ID（Where ID refers to the device ID/SN，e.g,:1306612）
Uplink data topic	mqtt/face/ID/Ack（Where ID refers to the device ID/SN，e.g,:1306612）

3.Search Local Face Database by Image Field Descriptions
Parameter information(Note:`Optional`is optional):

Key	Type	Values	Description
operator	String	GetPictureSearch	Search local face database by image
messageId	String		Message id, used by the platform to distinguish each message.
info	Object		Concrete content
facesluiceId	String		The clientID for connecting to the server,device ID/SN is used by default to ensure uniqueness
MaxSimilarity	float		Similarity threshold
MaxNum	int	1	Maximum number
picinfo	String		Base64 encoded data of face images (no more than 1M)

4.Example of Search Local Face Database by Image

```
{
    "operator": "GetPictureSearch",
    "messageId": "ID:localhost-637050888589478689:44009:24:40",
    "info": {
        "facesluiceId": "1379743",
        "MaxSimilarity": 80,
        "MaxNum": 1
    },
    "picinfo": "......"
}
```

5.Device Response Field Descriptions
Parameter information(Note: `Optional` is optional):

Key	Type	Values	Description
operator		GetPictureSearch-Ack	Answer to search local face database by image
messageId	String		Message id, used by the platform to distinguish each message.
code	String		Command execution error code 200-successes,see also 10.21 Search Local Face Database by Image Error Codes
info	Object		Concrete content
facesluiceId	String		The clientID for connecting to the server,device ID/SN is used by default to ensure uniqueness
TotalNum	int		Total number of lists returned from search results (up to 20)
SearchInfo	array		Matching personnel information content
personType	String	0~1	0: white list 1: black list
name	String	Length 32 characters (including terminator)	Name
cardType2	String	0~3	Wiegand Card Number Generation Method 0: public number 1: automatic generation 2: manual input 3: do not use access card numbers
cardNum2	String		Access card number
gender	String	0~1	Genders 0: male 1: female
nation	String	Reserve	Reserve
idCard	String		Identity card number (not more than 32 bytes)
birthday	String	1992-06-15	Brithday，YYYY-MM-DD
telnum1	String		Telephone number，Length 32 characters (including terminator)
native	String		Native，Length 32 characters (including terminator)
address	String		Address,Length 72 characters (including terminator)

Key	Type	Values	Description
notes	String		Notes,Length 64 characters (including terminator)
customId	String	Length 48 characters (including terminator)	Platform generated id, uniquely identifies different personnel.
Time	String	2022-01-10/12:10:10	List creation time
result	String	"ok"/"fail"	Operating result
detail	String (optional)		Error message when Result is "Fail"

6.Device Response Example

```
{
"messageId":"ID:localhost-637050888589478689:44009:24:40",
"operator": "GetPictureSearch-Ack",
"code":"200",
"info": {
"facesluiceId":"1783089",
"TotalNum": 1,
"SearchInfo": [{
"personType": "0",
"name": "xxx",
"cardType2":"0",
"cardNum2": "1",
"gender": "0",
"nation": "1",
"idCard": " ",
"birthday": "2020-12-10",
"telnum1": " ",
"native": " ",
"address": " ",
"notes": " ",
"customId": " ",
"Time": "2021-12-10/11:05:11"
}]},
"result":"ok"
}
```

5.6.3 Detecting Faces in Image

1.Description

This interface provides rough detection of whether a face can be extracted from an image.

2.API Description

Items	Description
Interface Name	Detecting faces in image
Data flow (active command side)	Platform->Device
Downlink data topic	mqtt/face/ID（Where ID refers to the device ID/SN，e.g,:1306612）
Uplink data topic	mqtt/face/ID/Ack（Where ID refers to the device ID/SN，e.g,:1306612）

3.Detecting Faces in Image Field Descriptions
Parameter information(Note:`Optional`is optional):

Key	Type	Values	Description
operator	String	DetectFaceFromPic	Detecting faces in image
messageId	String		Message id, used by the platform to distinguish each message.
info	Object		Concrete content
facesluiceId	String		The clientID for connecting to the server,device ID/SN is used by default to ensure uniqueness
picinfo	String		Base64 encoded data of face images (no more than 1M)

4.Example of Detecting Faces in Image

```
{
    "operator": "DetectFaceFromPic",
    "messageId": "ID:localhost-637050888589478689:44009:24:40",
    "info": {
        "facesluiceId": "1379743",
        "picinfo": "......"
    }
}
```

5.Device Response Field Descriptions

Key	Type	Values	Description
operator		DetectFaceFromPic-Ack	Answer to detecting faces in image
messageId	String		Message id, used by the platform to distinguish each message.
code	String		Command execution error code 200-successes,see also 10.22 Detecting Faces in Image Error Codes
info	Object		Concrete content
faceluiceId	String		The clientID for connecting to the server,device ID/SN is used by default to ensure uniqueness
DetectFace	String	0~1	Detection result 0: No face detected 1: Face detected
result	String	"ok"/"fail"	Operating result
detail	String (optional)		Error message when Result is "Fail"

6.Device Response Example

```
{
    "messageId": "ID:localhost-637050888589478689:44009:24:40",
    "operator": "DetectFaceFromPic-Ack",
    "code" : "200",
    "info": {
        "facesluiceId": "1379743",
        "DetectFace": "1",
        "result": "ok"
    }
}
```

5.7 Playing Voice Files

The device supports playing the original voice files of the device, as well as customer-defined voice files. Customer-defined voice files need to meet the specific voice file format. All voice files need to be single-channel, with a sampling rate of 16KHz, ending with ".wav" format. If it is a customer-defined voice file, it needs to end with "_C.wav" format, you can upgrade the voice file in the web page file upgrade interface.

⚠ Later, this interface function will be directly realized in the data format of remote door opening, or door opening mode selection for face verification + remote door opening mode reply, this interface can be called without separate.

5.7.1 Getting Audio Files

1.Description

Gets the name of an existing voice file for the device.

2.API Description

Items	Description
Interface Name	Gets the name of an existing voice file for the device.
Data flow (active command side)	Platform->Device
Downlink data topic	mqtt/face/ID（Where ID refers to the device ID/SN，e.g,:1306612）
Uplink data topic	mqtt/face/ID/Ack（Where ID refers to the device ID/SN，e.g,:1306612）

3.Getting Audio Files Field Descriptions
Parameter information(Note:`Optional`is optional):

Key	Type	Values	Description
operator	String	GetAudio	Gets the name of an existing voice file for the device.
messageId	String		Message id, used by the platform to distinguish each message.
info	Object		Concrete content

4.Example of Getting Audio Files

```
{
    "operator":"GetAudio",
    "messageId":"ID:localhost-637050888589478689:44009:24:40",
    "info":{

    }
}
```

5.Device Response Field Descriptions

Key	Type	Values	Description
operator		GetAudio-Ack	Answer to get the name of an existing voice file for the device.
messageId	String		Message id, used by the platform to distinguish each message.
code	String		Command execution error code 200-successes,see also 10.23 Getting Audio Files Error Codes
info	Object		Concrete content
facesluiceId	String		The clientID for connecting to the server,device ID/SN is used by default to ensure uniqueness
AudioNum	String		Number of original sound files on the device
AudioName	Array		Collection of original sound file names of the device
AudioCustomerNum	String		Number of customer-defined sound files on the device
AudioCustomerName	Array		Collection of customer-defined sound file names of the device
result	String	"ok"/"fail"	Operating result
detail	String (optional)		Error message when Result is "Fail"

6.Device Response Example

```
{
    "messageId": "ID:localhost-637050888589478689:44009:24:40",
    "operator": "GetAudio-Ack",
    "code" : "200",
    "info": {
        "facesluiceId": "1306612",
        "result": "ok",
        "AudioNum": "3",
        "AudioName": ["welcome.wav", "unRecorded.wav", "sorry.wav"],
        "AudioCustomerNum": "1",
        "AudioCustomerName": ["CustomAudioFile_C.wav"]
    }
}
```

5.7.2 Playing Audio Files

1.Description
Playing a voice file from a device

2.API Description

Items	Description
Interface Name	Playing audio files
Data flow (active command side)	Platform->Device
Downlink data topic	mqtt/face/ID（Where ID refers to the device ID/SN，e.g,:1306612）
Uplink data topic	mqtt/face/ID/Ack（Where ID refers to the device ID/SN，e.g,:1306612）

3.Playing Audio Files Field Descriptions
Parameter information(Note:`Optional`is optional):

Key	Type	Values	Description
operator	String	voiceplay	Playing audio files
messageId	String		Message id, used by the platform to distinguish each message.
info	Object		Concrete content
facesluiceId	String		The clientID for connecting to the server,device ID/SN is used by default to ensure uniqueness.
UsrType	int	1~2 (Default 1)	1: Play the original sound file in the device 2: Play the customer-defined voice file in the device
voicetype	String	e.g.:welcome.wav	Name of the sound file to be played.

4.Example of Playing Audio Files

```
{
    "operator": "voiceplay",
    "messageId": "ID:localhost-637050888589478689:44009:24:40",
    "info": {
        "UsrType": 1,
        "voicetype": "welcome.wav"
    }
}
```

5.Device Response Field Descriptions

Key	Type	Values	Description
operator		voiceplay-Ack	Answer to playing audio files
messageId	String		Message id, used by the platform to distinguish each message.
code	String		Command execution error code 200-successes,see also 10.24 Playing Audio Files Error Codes
info	Object		Concrete content
facesluiceId	String		The clientID for connecting to the server,device ID/SN is used by default to ensure uniqueness
result	String		Operating result.
detail	String (optional)		Error message when Result is "Fail"

6.Device Response Example

```
{
    "messageId": "XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX",
    "operator": "voiceplay-Ack",
    "code": "200",
    "info": {
        "facesluiceId": "2864575",
        "result": "ok"
    }
}
```

6.Logs Reporting

Logs reporting section mainly introduces control logs reporting and stranger capture reporting, you need to pay attention to whether the device is enabled to `Continue Transmitting After Disconnection` , under this mode, the platform needs to reply to the device in accordance with the format after receiving the reported data.Manual Repush Control Record and Manual Repush Stranger Capture Record will realize the historical record data to be pushed to the server again through the interface.

6.1 Continue Transmitting After Disconnection

1.Description

The authentication (control) logs and **stranger capture logs** generated by the device have the difference of enabling and disabling the function of `Continue Transmitting After Disconnection` in the reporting mode. The platform only needs to receive and process the data when the data is reported without enabling the function of `Continue Transmitting After Disconnection`, but **there is a risk of data loss in case of network abnormality**; when the data is reported with the function of `Continue Transmitting After Disconnection` enabled, the platform needs to reply to the device according to the following format within **10 seconds** after receiving the data, and if there is no reply to the device or the reply to the device is incorrect, the Face Recognition All-in-One Machine will push **the same record data** all the time.

2.API Description

Items	Description
Interface Name	Logs `Continue Transmitting After Disconnection` responses
Uplink data topic	mqtt/face/ID/Rec（Where ID refers to the device ID/SN，e.g,:1306612）
Downlink data topic	mqtt/face/ID（Where ID refers to the device ID/SN，e.g,:1306612）

3.`Continue Transmitting After Disconnection` Responses Field Descriptions

Parameter information(Note:`Optional` is optional):

Key	Type	Values	Description
operator		PushAck	Response to the Platform's receipt of logs uploads for the device `Continue Transmitting After Disconnection` mode
info	Object		Concrete content
PushAckType	int	1~2	Logs type 1:Reply to receive stranger capture logs 2:Reply to receive authentication (control) logs
SnapOrRecordID	int		When PushAckType=1 fill in the SnapID of the received stranger log library When PushAckType=2 fill in the RecordID of the received control log library
openDoor	int (optional)	0~1	Open door action 0: Do not open door 1: Open door
showInfo	String (optional)	64bytes (including terminator)	UI tips

4.Example of `Continue Transmitting After Disconnection` Responses

4.1.Example of Control Logs `Continue Transmitting After Disconnection` Responses

⚠ Assuming the RecordID of the control log received by the platform is 381

```
{
    "messageId": "ID:localhost-637046811507388956:23952:65:48",
    "operator": "PushAck",
    "info": {
        "PushAckType": 2,
        "SnapOrRecordID": 381
    }
}
```

4.2.Example of Captured Logs `Continue Transmitting After Disconnection` Responses

⚠ Assuming the SnapID of the stranger captured log received by the platform is 867

```
{
    "operator": "PushAck",
    "messageId": "ID:localhost-637046811507388956:23952:65:49",
    "info": {
        "PushAckType": 1,
        "SnapOrRecordID": 867
    }
}
```

5.Device Response Field Descriptions
Parameter information(Note:`Optional` is optional):

Key	Type	Values	Description
operator		PushAck-Ack	Answer to response to the Platform's receipt of logs uploads for the device `Continue Transmitting After Disconnection` mode
messageId	String		Message id, used by the platform to distinguish each message.
code	String		Command execution error code 200-successes,see also 10.25 Continue Transmitting After Disconnection Error Codes
info	Object		Concrete content
facesluiceId	String		The clientID for connecting to the server,device ID/SN is used by default to ensure uniqueness
PushAckType	String	1~2	Logs type 1:Reply to receive stranger capture logs 2:Reply to receive authentication (control) logs
SnapOrRecordID	String		When PushAckType=1 fill in the SnapID of the received stranger log library When PushAckType=2 fill in the RecordID of the received control log library
result	String	"ok"/"fail"	Operating result
detail	String (optional)		Error message when Result is "Fail"

6.Device Response Example

6.1 Example of Device ReplyControl Logs `Continue Transmitting After Disconnection` Responses

```
{
    "messageId": "ID:localhost-637046811507388956:23952:65:48",
    "operator": "PushAck-Ack",
    "code" : "200",
    "info": {
        "facesluiceId": "1305132",
        "result": "ok",
        "PushAckType": "2",
        "SnapOrRecordID": "381"
    }
}
```

6.2Example of Example of Device ReplyCaptured Logs`Continue Transmitting After Disconnection` Responses

```
{
    "messageId": "ID:localhost-637046811507388956:23952:65:48",
    "operator": "PushAck-Ack",
    "info": {
        "facesluiceId": "1305132",
        "result": "ok",
        "PushAckType": "1",
        "SnapOrRecordID": "381"
    }
}
```

6.2 Reporting of Certification Results

1.Description

When the device turns on authentication logs subscription, all personnel identification logs (control logs) generated are reported to the subscribed URL, control records include (but are not limited to)

1)Regular personnel identification logs

2)Control logs generated by remote door opening commands

3)Unauthorized access logs generated under a special version (access strategy)(“VerifyStatus”:24)

4)Logs of blacklisted denials of passage(“VerifyStatus”:2)

⚠Note:If the device has enabled the `Continue Transmitting After Disconnection` function, the subscription platform needs to reply to the data reported by the device, otherwise the device will send the same stranger capture log; refer to the description in the 4、HTTP Subscription `Continue Transmitting After Disconnection` Function section for the introduction of the specific reply, and note that after the `Continue Transmitting After Disconnection` function is enabled, it is necessary to reply to all the stranger capture logs reported by the device.

2.API Description

Items	Description
Interface Name	Reporting of control logs
Uplink data topic	mqtt/face/ID/Rec（Where ID refers to the device ID/SN，e.g,:1306612）
Downlink data topic	mqtt/face/ID（Where ID refers to the device ID/SN，e.g,:1306612）

3.Reporting of Certification Results Field Description
Parameter information(Note:`Optional`is optional):

Key	Type	Values	Description
operator		RecPush	Reporting of control logs
info	Object		Concrete content
customId	String	Length 48 characters (including terminator)	Platform generated id, uniquely identifies different personnel.
personId	String	0-N	Device personnel list database primary key self-incrementing ID, non-whitelisted mode value is 0
RecordID	String	1-N	Device control log library self-increment ID,for Continue Transmitting After Disconnection mode reply to face machine
VerifyStatus	String	0~3 or 22、24	Authentication results 0: None 1: Allowed 2: Denied 3: Not yet registered 22:To be verified (control record for door opening method 3: face verification + remote door opening method) 24:No permission (control record for special version of non-pass time period)
PersonType	String	0~1	List Type 0: White list 1: Black list
similarity1	float	0~100.0	Black and white list comparison similarity
similarity2	float	0~100.0	ID image comparison similarity
Sendintime	Int	0~1	Whether information is pushed in a timely manner 0:Not pushed in a timely manner (push time is greater than the authentication time by 10 seconds) 1:Pushed in a timely manner
direction	String		Entrance/Exit Direction Entrance:"entr" Exit:"exit" Unidirectional: "unknow"

Key	Type	Values	Description
otype	String	Authentication Type 1: Whitelist Verification 2: ID Verification 3: Whitelist + ID Verification 7:MQTT Remote Door Opening 21:RF Card Verification(**Built-in card swiping model**) 22:RF Card Verification + Whitelist Verification(**Built-in card swiping model**) 24:Wergand Card Verification 25. Wigand Card + Whitelist Verification 27:HTTP Remote Door Opening	1:Universal model 1~3 or 21~22 or 24~25 or 27； 2:Special model: **Support for mask detection** or (|) operation 0x100 indicates a pass with mask detection; **Supports body temperature detection** or (|) operation 0x200 indicates a pass with temperature detection； **Support mask + body temperature detection** Or (|) operation 0x300 means with mask + temperature detection pass; e.g.Masks + whitelist verification（0x101） Body Temperature + Whitelist Verification（0x201） Mask + Body Temperature + Wiegand Swipe + Whitelist Verification（0x319））
personName	String		Name
facesluiceId	String		The clientID for connecting to the server,device ID/SN is used by default to ensure uniqueness。
facesluiceName	String	64 Bytes	Device name
idCard	String		ID number
telnum	String		Telephone Number
left	String	0-N	Coordinates to the left of the face rectangle box
top	String	0-N	Coordinates to the top of the face rectangle box

Key	Type	Values	Description
right	String	0-N	Coordinates to the right of the face rectangle box
bottom	String	0-N	Coordinates to the bottom of the face rectangle box
time	String	0-N	Creation time of the control log. YYYY-MM-DD hh-mm-ss
PushType	String	0-2	Push type 0:Reserved 1:Reserved 2: Manual Push Control Logging data return
OpendoorWay	String	0~3	Open the door 0:Face open the door 1:Remote open the door 2:Remote open the door or face 3:Local verification remote open the door
cardNum2	String	0-N	Wiegand access card number.
WGFacilityCode	String (optional)	0-N	Facility code (Valid when Wiegand protocol is 8+16 or 8+24) is used with cardNum2
RFIDCard	String (optional)		RFID card number, maximum length of decimal 10 (hexadecimal 8) characters (Built-in card models)
isNoMask	String	0~2	Whether to wear a mask **(Note: Mask model support)** 0:Passage with mask 1:Passage without mask 2:Passage without mask (open door condition checkbox allows passage without mask))
temperature	float (optional)	37.0	Real-time detection of face temperature, return to detect temperature only when the door opening mode with temperature detection mode**(Temperature measurement model support)**

Key	Type	Values	Description
temperatureAlarm	String (optional)	0-1	Real-time detection of whether the face temperature exceeds the threshold value **（Temperature measurement model support）** 0:Not exceeded; 1:Exceeded
temperatureMode	String (optional)	0-1	1:Fahrenheit 0:Celsius （Supported by temperature measurement models only）
GpsLongitude	String (optional)	0-N	GPS longitude data (Models with GPS module support)
GpsLatitude	String (optional)	0-N	GPS latitude data (Models with GPS module support)
GpsTime	String (optional)		The time format is YYYY-MM-DD hh-mm-ss (Models with GPS module support)
dwFileIndex	String (optional)	0-N	File index of of image captured in control log
dwFilePos	String (optional)	0-N	Location of image captured in control log
pic	String (optional)		Base64 encoding of captured images (up to 1M)

4.Example of Reporting of Certification Results

```
{
    "operator" : "RecPush",
    "info" : {
        "customId": "063c81e0fce184c696cdb7e049230f5e",
        "personId" : "3",
        "RecordID" : "2",
        "VerifyStatus" : "1",
        "PersonType" : "0",
        "similarity1" : "89.900002",
        "similarity2" : "0.000000",
        "Sendintime" : 1,
        "direction" : "unknow",
        "otype" : "1",
        "personName" : "test",
        "facesluiceId" : "1787156",
```

```
        "facesluiceName" : "Face1",
        "idCard" : " ",
        "telnum" : " ",
        "left" : "158",
        "top" : "426",
        "right" : "672",
        "bottom" : "946",
        "time" : "2023-01-01 00:00:00",
        "PushType" : "0",
        "OpendoorWay" : "0",
        "cardNum2" : "1",
        "RFIDCard" : "0",
        "szQrCodeData" : "",
        "isNoMask" : "0",
        "pic": "*Data omitted here*/"
    }
}
```

5.Description of Platform Response Fields

⚠ **Response required for reporting data when set to `Continue Transmitting After Disconnection` mode only.**

Parameter information(Note:`Optional` is optional):

Key	Type	Values	Description
operator		PushAck	Platform replies to received control logs (only in Continue Transmitting After Disconnection mode)
info	Object		Concrete content
PushAckType	String	1-2	Logs type 1:Reply to receive stranger capture logs 2:Reply to receive authentication (control) logs
SnapOrRecordID	String	0-N	When PushAckType=1 fill in the SnapID of the received stranger log library When PushAckType=2 fill in the RecordID of the received control log library
result			Operating result
detail			Error message when Result is "Fail"

6.Example of Platform Response

```
{
    "messageId": "ID:localhost-637046811507388956:23952:65:48",
    "operator": "PushAck",
    "info": {
        "PushAckType": "2",
        "SnapOrRecordID": "2"
    }
}
```

6.3 Stranger Captured Logs

1.Description

Logs of strangers who are not registered at the device or people whose comparison similarity does not reach the threshold will be saved in the stranger capture log of the device, and setting up the report stranger log will get the corresponding data.

⚠Note:If the device has enabled the `Continue Transmitting After Disconnection` function, the subscription platform needs to reply to the data reported by the device, otherwise the device will send the same stranger capture log; refer to the description `Continue Transmitting After Disconnection` Function section for the introduction of the specific reply, and note that after the `Continue Transmitting After Disconnection` function is enabled, it is necessary to reply to all the stranger capture logs reported by the device.

2.API Description

Items	Description
Interface Name	Stranger snap push
Uplink data topic	mqtt/face/ID/Snap（Where ID refers to the device ID/SN，e.g,:1306612）
Downlink data topic	mqtt/face/ID（Where ID refers to the device ID/SN，e.g,:1306612）

3.Description of reported fields
Parameter information(Note:`Optional`is optional):

Key	Type	Values	Description
operator		StrSnapPush	Stranger snap push
info	Object		Concrete content
SnapID	String	1-N	Device control log library self-increment ID,for Continue Transmitting After Disconnection mode reply to face machine
facesluiceId	String		The clientID for connecting to the server,device ID/SN is used by default to ensure uniqueness
facesluiceName	String		Device name
time	String		Stranger capture recording time.YYYY-MM-DD hh-mm-ss
direction	String	Entrance:"entr" Exit:"exit" Unidirectional: "unknow"	Entrance/Exit Direction
left	int	0-N	Coordinates to the left of the face rectangle box
top	int	0-N	Coordinates to the top of the face rectangle box
right	int	0-N	Coordinates to the right of the face rectangle box
bottom	int	0-N	Coordinates to the bottom of the face rectangle box
PushType	int	0-2	Push type 0:Reserved 1:Reserved 2: Manual Push Control Logging data return.
isNoMask	int	0~2	Whether to wear a mask **(Note: Mask model support)** 0:Passage with mask 1:Passage without mask 2:Passage without mask (open door condition checkbox allows passage without mask)
temperature	float (optional)	37.3	Real-time detection of face temperature, return to detect temperature only when the door opening mode with temperature detection mode **(Temperature measurement model support)**

Key	Type	Values	Description
temperatureAlarm	String (optional)		Real-time detection of whether the face temperature exceeds the threshold value **(Temperature measurement model support)**
temperatureMode	String (optional)	0-1	Temperature display mode **(Temperature measurement model support)** 0: Celsius temperature 1: Fahrenheit temperature
GpsLongitude	String (optional)	0-N	GPS longitude data (Models with GPS module support)
GpsLatitude	String (optional)	0-N	GPS latitude data (Models with GPS module support)
GpsTime	String (optional)		The time format is YYYY-MM-DD hh-mm-ss (Models with GPS module support)
pic	String (optional)		Base64 encoding of captured images (up to 1M)

4.Reporting Example

```
{
    "operator" : "StrSnapPush",
    "info" : {
        "SnapID" : "12",
        "facesluiceId" : "1787156",
        "facesluiceName" : "Face1",
        "direction" : "unknow",
        "left" : "156",
        "top" : "354",
        "right" : "600",
        "bottom" : "802",
        "time" : "2023-01-01 00:00:00",
        "PushType" : "0",
        "isNoMask" : "0",
        "pic":
"......"
    }
}
```

5.Description of Platform Response Fields

Response required for reporting data when set to `Continue Transmitting After Disconnection` mode only.

Parameter information(Note: `Optional` is optional):

Key	Type	Values	Description
operator		PushAck	Platform replies to received control logs (only in Continue Transmitting After Disconnection mode)
info			Concrete content
PushAckType	String	1-2	Logs type 1:Reply to receive stranger capture logs 2:Reply to receive authentication (control) logs
SnapOrRecordID	String	0-N	When PushAckType=1 fill in the SnapID of the received stranger log library When PushAckType=2 fill in the RecordID of the received control log library
result			Operating result
detail			Error message when Result is "Fail"

6.Example of Platform Response Fields

```
{
    "messageId": "ID:localhost-637046811507388956:23952:65:23",
    "operator": "PushAck",
    "info": {
        "PushAckType": "1",
        "SnapOrRecordID": "12"
    }
}
```

6.4 Manual Push Control Logging

1.Description

This interface is mainly used to re-push historical control logs. After calling this interface the device actively uploads the control logs of the time period to the MQTT server via the "Identification Log" subscription topic mqtt/face/ID/Rec (where ID refers to the ID of this device, e.g. 1305433). This interface requires that the device is connected to the server and that "Identification Log" subscription is enabled. Generally, due to some special reasons, the platform does not receive a certain part of the historical data. The “**PushType**” field of the reported data can be used to determine the type of data to be pushed, and the platform needs to de-emphasize the data reported by this interface.

⚠ For control log reporting field parameters see section 6.2 Reporting of Certification Results for parameter descriptions.

2.API Description

Items	Description
Interface Name	Manual push control logging
Uplink data topic	mqtt/face/ID/Rec (Where ID refers to the device ID/SN，e.g,:1306612)
Downlink data topic	mqtt/face/ID (Where ID refers to the device ID/SN，e.g,:1306612)

3.Manual Push Control Logging Field Descriptions
Parameter information(Note: `Optional` is optional):

Key	Type	Values	Description
operator	String	ManualPushRecords	Manual push control logging
info	String		Concrete content
facesluiceId	String (optional)		The clientID for connecting to the server,device ID/SN is used by default to ensure uniqueness
TimeS	string	YYYY-MM-DDThh:mm:ss	Control log start time that needs to be pushed. e.g. 2020-07-29T21:50:05
TimeE	string	YYYY-MM-DDThh:mm:ss	Control log end time that needs to be pushed. e.g. 2020-07-29T21:50:05

4.Example of Manual Push Control Logging

⚠ Note the format of the time, which requires the character T between the date and the hour.

```
{
    "operator": "ManualPushRecords",
    "messageId": "ID:localhost-637050888589478689:44009:24:40",
    "info": {
        "facesluiceId": "1306612",
        "TimeS": "2020-07-28T6:00:00",
        "TimeE": "2020-08-12T23:00:00"
    }
}
```

5.Device Response Field Descriptions
Parameter information(Note: `Optional` is optional):

Key	Type	Values	Description
operator		ManualPushRecords-Ack	Answer to manual push control logging
messageId	String		Message id, used by the platform to distinguish each message.
code	String		Command execution error code 200-successes,see also 10.26 Manual Push Control Logging Error Codes
info	Object		Concrete content
RecordNum	String	0-N	Total number of matching control logs (return value) 0:No data searched
result			Operating result
detail			Error message when Result is "Fail"

6.Device Response Example

```
{
    "messageId": "ID:localhost-637050888589478689:44009:24:40",
    "operator": "ManualPushRecords-Ack",
    "code": "200",
    "info": {
        "facesluiceId": "1306612",
        "RecordNum": "165",
        "result": "ok"
    }
}
```

6.5 Manual Push Stranger Logging

1.Description

This interface is mainly used to re-push historical stranger capture logs. After calling this interface the device actively uploads the stranger capture logs of the time period to the MQTT server via the "Stranger" subscription topic mqtt/face/ID/Snap (where ID refers to the ID of this device, e.g. 1305433). This interface requires that the device is connected to the server and that "Stranger" subscription is enabled. Generally, due to some special reasons, the platform does not receive a certain part of the historical data. The “**PushType**” field of the reported data can be used to determine the type of data to be pushed, and the platform needs to de-emphasize the data reported by this interface.

⚠ For stranger capture log reporting field parameters see section 6.3 Stranger Captured Logs for parameter descriptions

2.API Description

Items	Description
Interface Name	Manual push stranger logging
Uplink data topic	mqtt/face/ID/Snap（Where ID refers to the device ID/SN，e.g,:1306612）
Downlink data topic	mqtt/face/ID（Where ID refers to the device ID/SN，e.g,:1306612）

3.Manual Push Stranger Logging Field Descriptions
Parameter information(Note:`Optional`is optional):

Key	Type	Values	Description
operator	String	ManualPushSnaps	Manual push stranger logging
info	String		Concrete content
faceluiceId	String		The clientID for connecting to the server,device ID/SN is used by default to ensure uniqueness。
TimeS	string	YYYY-MM-DDThh:mm:ss	Stranger capture log start time that needs to be pushed. e.g.2020-07-29T21:50:05
TimeE	string	YYYY-MM-DDThh:mm:ss	Stranger capture log start time that needs to be pushed. e.g. 2020-07-29T21:50:05

4.Example of Manual Push Stranger Logging

 Note the format of the time, which requires the character T between the date and the hour.

```
{
    "operator": "ManualPushSnaps",
    "messageId": "ID:localhost-637050888589478689:44009:24:41",
    "info": {
        "facesluiceId": "1306612",
        "TimeS": "2020-07-28T6:00:00",
        "TimeE": "2020-08-12T23:00:00"
    }
}
```

5.Device Response Field Descriptions
Parameter information(Note:`Optional`is optional):

Key	Type	Values	Description
operator		ManualPushSnaps-Ack	Answer to Manual push stranger logging.
messageId	String		Message id, used by the platform to distinguish each message.
code	String		Command execution error code 200-successes,see also 10.27 Manual Push Stranger Logging Error Codes
info	Object		Concrete content
RecordNum	String	0-N	Total number of matching stranger capture logs （return value） 0:No data searched
result			Operating result
detail			Error message when Result is "Fail"

6.Device Response Example

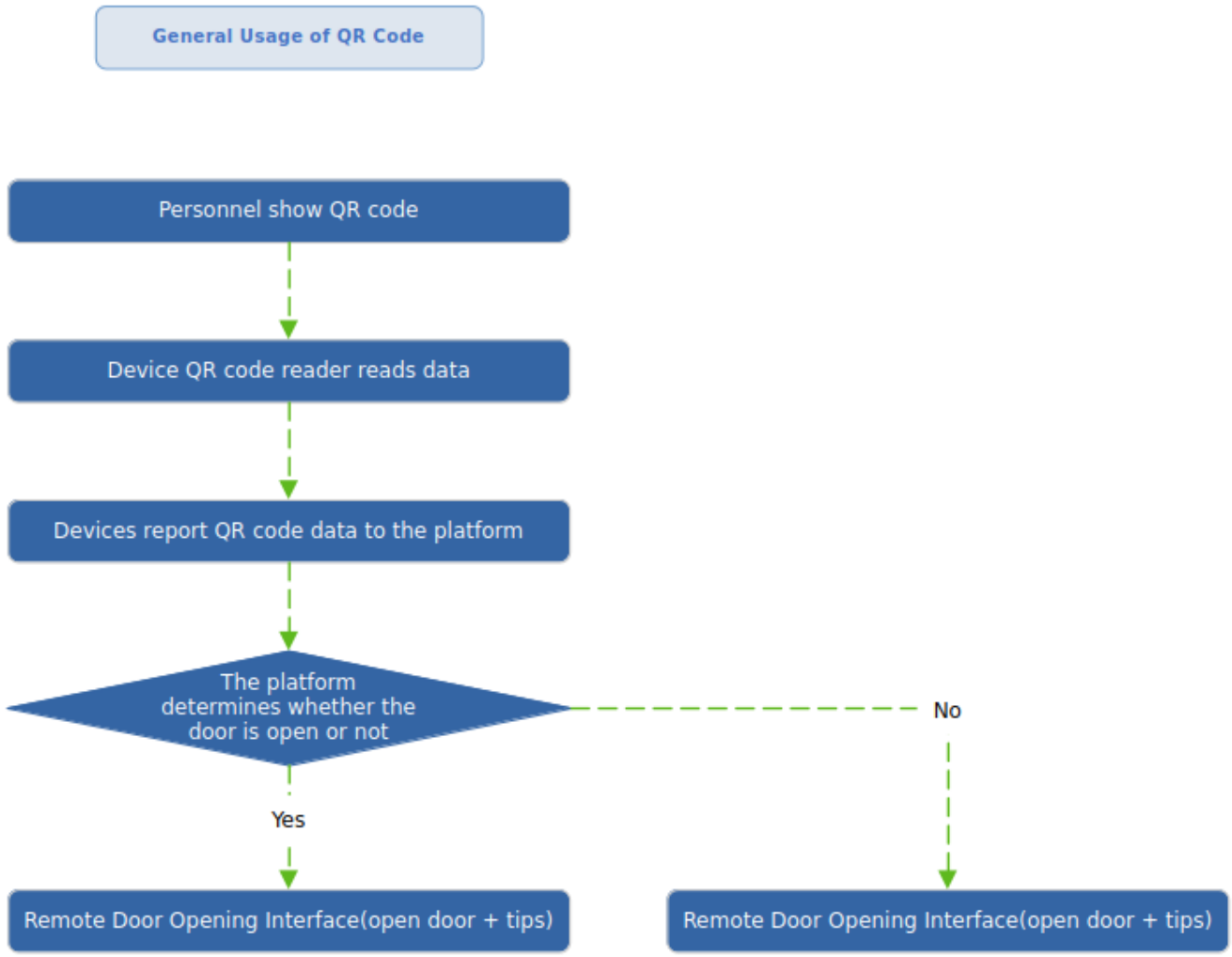
```
{
    "messageId": "ID:localhost-637050888589478689:44009:24:41",
    "operator": "ManualPushSnaps-Ack",
    "code": "200",
    "info": {
        "facesluiceId": "1306612",
        "RecordNum": "1063",
        "result": "ok"
    }
}
```

6.6 QR Code Reporting

1.Description

This interface is mainly for models with external QR code readers or models with built-in QR code readers (except for the health code version) to transmit QR code data to the platform after the customer scans the QR code.

⚠ The usage of this interface is shown below:



2.API Description

Items	Description
Interface Name	QR code reporting
Uplink data topic	mqtt/face/ID/QRCode (Where ID refers to the device ID/SN , e.g,:1306612)
Downlink data topic	mqtt/face/ID (Where ID refers to the device ID/SN , e.g,:1306612)

3.QR Code Reporting Field Descriptions

Parameter information(Note: `Optional` is optional):

Key	Type	Values	Description
operator	String	QRCodePush	QR code reporting
info	String		Concrete content
facesluiceId	String		The clientID for connecting to the server,device ID/SN is used by default to ensure uniqueness
facesluiceName	String		Device name
QRCodeInfo	string		QR code raw data
time	string	YYYY-MM-DD hh:mm:ss	QR code generation time. e.g. 2020-07-29 21:50:05

4.Example of QR Code Reporting

 Note the format of the time, which requires the character T between the date and the hour.

```
{
    "operator": "QRCodePush",
    "info": {
        "facesluiceId": "1326491",
        "facesluiceName": "Face1",
        "QRCodeInfo": "A1B2C3D4E5F6G7",
        "time": "2020-04-24 20:52:51"
    }
}
```

6.7 IC Card/RFID Card Reporting

1.Description

When the device is in door opening mode with card swipe verification, the information after each card swipe is actively reported to the platform.

2.API Description

Items	Description
Interface Name	IC Card/RFID card reporting
Uplink data topic	mqtt/face/ID/Card (Where ID refers to the device ID/SN，e.g,:1306612)
Downlink data topic	mqtt/face/ID (Where ID refers to the device ID/SN，e.g,:1306612)

3.Description of IC Card/RFID Card Reporting fields
Parameter information(Note:`Optional`is optional):

Key	Type	Values	Description
operator	String	CardPush	IC Card/RFID card reporting
info	String		Concrete content
facesluiceId	String		The clientID for connecting to the server,device ID/SN is used by default to ensure uniqueness
facesluiceName	String		Device name
time	string	YYYY-MM-DD hh:mm:ss	Creation time e.g. 2020-07-29 21:50:05
CardInfo	Object		Card Number Information Keywords.
Endian	string	0-1	Card number big/little endian mode 0:Big Endian 1:Little Endian
CardMode	string	0-1	Card Number Method 0:decimal 1:hexadecimal
CardNum	String		Card number

4.Example of IC Card/RFID Card Reporting

```
{
    "operator": "CardPush",
    "info": {
        "facesluiceId": "1306612",
        "facesluiceName": "Face1",
        "time": "2020-11-20 14:28:32",
        "CardInfo": {
            "CardMode": "1",
            "Endian": "1",
            "CardNum": "86664719"
        }
    }
}
```

6.8 Reporting of Identity Card Information

1.Description

When the device selects the door opening mode with ID card verification, the information of each ID card swipe of the personnel will be reported to the platform separately through this interface.

2.API Description

Items	Description
Interface Name	Reporting of Identity Card Information
Uplink data topic	mqtt/face/ID/IDCard (Where ID refers to the device ID/SN , e.g,:1306612)
Downlink data topic	mqtt/face/ID (Where ID refers to the device ID/SN , e.g,:1306612)

3.Description of reported fields
Parameter information(Note:`Optional`is optional):

Key	Type	Values	Description
operator	String	IDCardPush	Reporting of Identity Card Information
info	String		Concrete content
facesluiceId	String		The clientID for connecting to the server,device ID/SN is used by default to ensure uniqueness
facesluiceName	String		Device name
time	string	YYYY-MM-DD hh:mm:ss	Time to swipe the ID card e.g. 2020-07-29 21:50:05
IDCardInfo	Object		Card number information keywords
IDCard_Idno	String		I.D. number
IDCard_Name	String		Name
IDCard_Gender	int		Genders 0: male 1: female
IDCard_Nation	int		Reserve
IDCard_Birthday	String	1992-06-17	Birthday
IDCard_Address	String		Address
IDCard_Idissue	String		Issuing authority
IDCard_Idperiod	String	20090828-20190828	Validity period: Month/Year - Month/Year
IDCard_photo	String		Photo ID, base64 encoding

4.Example of Reporting of Identity Card Information

```
{
    "operator": "IDCardPush",
    "info": {
        "facesluiceId": "1326491",
        "facesluiceName": "Face1",
        "time": "2020-10-13 15:00:38",
        "IDCardInfo": {
            "IDCard_Idno": "421126198702054321",
            "IDCard_Name": "xxx",
            "IDCard_Gender": 0,
            "IDCard_Nation": 1,
            "IDCard_Birthday": "1984-12-11",
            "IDCard_Address": "Shenzhen",
            "IDCard_Idissue": "Shenzhen",
            "IDCard_Idperiod": "20090828-20190828",
            "IDCard_photo": "*Data omitted here*/caowl/9k="
        }
    }
}
```


7.Device Management

7.1 Getting Device Software Version Number

1.Description

Get the software version number of the device. The software version number is used to manage the device and version upgrades.

2.API Description

Items	Description
Interface Name	Get the software version number of the device
Downlink data topic	mqtt/face/ID（Where ID refers to the device ID/SN，e.g,:1306612）
Uplink data topic	mqtt/face/ID/Ack（Where ID refers to the device ID/SN，e.g,:1306612）

3.Getting Device Software Version Number Field Descriptions
Parameter information(Note:`Optional`is optional):

Key	Type	Values	Description
operator	String	Versions	Get the software version number of the device
messageId	String		Message id, used by the platform to distinguish each message.
info	Object		Concrete content

4 Example of Getting Device Software Version Number

```
{
    "operator": "Versions",
    "messageId": "ID:localhost-637050272518414388:79346:87:5",
    "info": {}
}
```

5.Device Response Field Descriptions
Parameter information(Note: `Optional` is optional):

Key	Type	Values	Description
operator		Versions-Ack	Answer to get the software version number of the device.
messageId	String		Message id, used by the platform to distinguish each message.
code	String		Command execution error code 200-successes,see also error cross-reference table
info	Object		Concrete content
facesluiceId	String		The clientID for connecting to the server,device ID/SN is used by default to ensure uniqueness
name	String		The software version number
buildtime	String		Software version build time.
result	String		Operating result
detail	String (optional)		Error message when Result is "Fail"

6.Device Response Example

```
{
    "messageId": "ID:localhost-637050272518414388:79346:87:5",
    "operator": "Versions-Ack",
    "code": "200",
    "info": {
        "facesluiceId": "1306612",
        "name": "v7.14.8.12-L-M-K-2.5W-AS",
        "buildtime": "2021-07-14/23:02:50",
        "result": "ok"
    }
}
```

7.2 Software Version Upgrade

1.Description

Remote upgrade command through the platform to complete the device version upgrade. Before upgrading, it is necessary to note that the upgraded version cannot be upgraded from a new version to an old one.

2.API Description

Items	Description
Interface Name	Remote upgrade software version.
Downlink data topic	mqtt/face/ID（Where ID refers to the device ID/SN，e.g,:1306612）
Uplink data topic	mqtt/face/ID/Ack（Where ID refers to the device ID/SN，e.g,:1306612）

3.Software Version Upgrade Field Descriptions
Parameter information(Note:`Optional`is optional):

Key	Type	Values	Description
operator	String	Upgrade	Remote upgrade software version.
messageId	String		Message id, used by the platform to distinguish each message.
info	Object		Concrete content
facesluiceId	String		The clientID for connecting to the server,device ID/SN is used by default to ensure uniqueness
name	String		File name .
upgradeType	int (optional)	1~4	Upgrade type, default 1; 1: Indicates upgrading software firmware package, *.swx file; 2: Upgrading voice file, *.wav file; 3: Upgrading png image file, *.png file; 4: Upgrading jpg image file, *.jpg file
path	String		The download path of the software version upgrade file (http or https), without upgradeType field or upgradeType=1, it means the download path of the upgrade file, **the suffix must be ".swx"**.

4.Example of Software Version Upgrade

```
{
    "messageId": "ID:localhost-637050900934386959:42763:53:1",
    "operator": "Upgrade",
    "info": {
        "name": "FaceGate_v7.14.8.12-L-M-K-AS-20210712.swx",
        "path": "https://mqttxxxx.oss-cn-
shenzhen.aliyuncs.com/face/FaceGate_v7.14.8.12-L-M-K-AS-20210712.swx"
    }
}
```

5.Device Response Field Descriptions

Parameter information(Note: `Optional` is optional):

Key	Type	Values	Description
operator		Upgrade-Ack	Answer to remote upgrade software version.
messageId	String		Message id, used by the platform to distinguish each message.
code	String		Command execution error code 200-successes,see also 10.28 Software Version Upgrade Error Codes
info	Object		Concrete content
facesluiceId	String		The clientID for connecting to the server,device ID/SN is used by default to ensure uniqueness
name	String		Version name.
result	String		Operating result
detail	String (optional)		Error message when Result is "Fail"

6.Device Response Example

```
{
    "messageId": "ID:localhost-637050900934386959:42763:53:1",
    "operator": "Upgrade-Ack",
    "code": "200",
    "info": {
        "facesluiceId": "1306612",
        "name": "FaceGate_v7.14.8.12-L-M-K-AS-20210712.swx",
        "result": "ok"
    }
}
```

7.3 Device Reboot

1.Description

Device reboot.

2.API Description

Items	Description
Interface Name	Device reboot
Downlink data topic	mqtt/face/ID（Where ID refers to the device ID/SN，e.g,:1306612）
Uplink data topic	mqtt/face/ID/Ack（Where ID refers to the device ID/SN，e.g,:1306612）

3.Device Reboot Field Descriptions
Parameter information(Note:`Optional`is optional):

Key	Type	Values	Description
operator	String	RebootDevice	Device reboot
messageId	String		Message id, used by the platform to distinguish each message.
info	Object		Concrete content

4 Example of Device Reboot

```
{
    "operator": "RebootDevice",
    "messageId": "XXXXXXXXXXXXXXXXXXXXXXXXXXXXXX",
    "info": {}
}
```

5.Device Response Field Descriptions
Parameter information(Note:`Optional`is optional):

Key	Type	Values	Description
operator		RebootDevice-Ack	Answer to device reboot
messageId	String		Message id, used by the platform to distinguish each message.
code	String		Command execution error code 200-successes,see alsoerror cross-reference table
info	Object		Concrete content
facesluiceId	String		The clientID for connecting to the server,device ID/SN is used by default to ensure uniqueness
result	String		Operating result
detail	String (optional)		Error message when Result is "Fail"

6.Device Response Example

```
{
    "messageId": "XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX",
    "operator": "RebootDevice-Ack",
    "code": "200",
    "info": {
        "facesluiceId": "1306612",
        "result": "ok"
    }
}
```

7.4 Restore Factory Settings

1.Description

Restore the factory is mainly used to open the door conditions, network parameters, capture logs, log records, personnel list library, control logs, system parameters and other parameters to restore.Restoring network parameters may cause network connection problems.Restoring the factory personnel list database will delete the personnel list comparison database, and will delete the corresponding certified personnel images of the control logs. **The operation of restoring capture logs, personnel list database and control logs is irreversible, so please be careful.**Parts that do not need to be recovered can be left out and partial recovery is supported.Restore factory settings part of the successful execution of the command device will automatically reboot.

⚠ Deletion of lists, control logs, etc. requires verification of the situation and is not a reversible operation.

2.API Description

Items	Description
Interface Name	Restore factory settings
Downlink data topic	mqtt/face/ID（Where ID refers to the device ID/SN，e.g,:1306612）
Uplink data topic	mqtt/face/ID/Ack（Where ID refers to the device ID/SN，e.g,:1306612）

3.Restore Factory Settings Field Descriptions
Parameter information(Note:`Optional`is optional):

Key	Type	Values	Description
operator	String	SetFactoryDefault	Restore factory settings
messageId	String		Message id, used by the platform to distinguish each message.
info	Object		Concrete content
DefaltAll	int (optional)	0~1	Restore all factory settings 0: No factory restoration 1: Factory restoration
DefaltDoorSet	int (optional)	0~1	Door open condition whether to restore the factory DefaltAll not zero or not issued is valid. 0: No factory restoration 1: Factory restoration
DefaltSoundSet	int (optional)	0~1	Whether the sound settings are restored to the factory DefaltAll not zero or not issued is valid. 0: No factory restoration 1: Factory restoration
DefaltNetPar	int (optional)	0~1	Whether the network parameters are restored to the factory DefaltAll not zero or not issued is valid. 0: No factory restoration 1: Factory restoratio
DefaltCenterPar	int (optional)	0~1	Whether the center connection parameters are restored to the factory DefaltAll not zero or not issued is valid. 0: No factory restoration 1: Factory restoration
DefaltCapture	int (optional)	0~1	Capture logs + control logs whether to restore the factory DefaltAll not zero or not issued is valid. 0: No factory restoration 1: Factory restoration
DefaltLog	int (optional)	0~1	System Logs whether to restore the factory DefaltAll not zero or not issued is valid. 0: No factory restoration 1: Factory restoration

Key	Type	Values	Description
DefaltPerson	int (optional)	0~1	Personnel list library + control logs whether to restore the factory DefaltAll not zero or not issued is valid. 0: No factory restoration 1: Factory restoration
DefaltRecord	int (optional)	0~1	Control logs whether to restore the factory DefaltAll not zero or not issued is valid. 0: No factory restoration 1: Factory restoration
DefaltMaintainTime	int (optional)	0~1	System maintenance time whether to restore the factory DefaltAll not zero or not issued is valid. 0: No factory restoration 1: Factory restoration
DefaltSystemSettings	int (optional)	0~1	System parameters: ID card reader type; whether to record the capture, ID card, control logs and other parameters whether to restore the factory DefaltAll not zero or not issued is valid. 0: No factory restoration 1: Factory restoration
DefaltMqtt	int (optional)	0~1	MQTT parameter whether to restore the factory DefaltAll not zero or not issued is valid. 0: No factory restoration 1: Factory restoration
DefaltHttp	int (optional)	0~1	HTTP parameter whether to restore the factory DefaltAll not zero or not issued is valid. 0: No factory restoration 1: Factory restoration
Defalt4G	int (optional)	0~1	4G parameter whether to restore the factory DefaltAll not zero or not issued is valid. 0: No factory restoration 1: Factory restoration

Key	Type	Values	Description
DefaltWifi	int (optional)	0~1	Wifi parameter whether to restore the factory DefaltAll not zero or not issued is valid. 0: No factory restoration 1: Factory restoration

4.Example of Restore Factory Settings

```
{
    "operator": "SetFactoryDefault",
    "messageId": "XXXXXXXXXXXXXXXXXXXXXXXXXXXXXX",
    "info": {
        "DefaltDoorSet": 1
    }
}
```

5.Device Response Field Descriptions
Parameter information(Note: `Optional` is optional):

Key	Type	Values	Description
operator		SetFactoryDefault-Ack	Answer to restore factory settings
messageId	String		Message id, used by the platform to distinguish each message.
code	String		Command execution error code 200-successes,see also error cross-reference table.
info	Object		Concrete content
facesluiceId	String		The clientID for connecting to the server,device ID/SN is used by default to ensure uniqueness.
reboot	String	0-1	Does the machine reboot 0: No reboot required 1: Reboot required
result	String		Operating result
detail	String (optional)		Error message when Result is "Fail"

6.Device Response Example

```
{
    "messageId": "XXXXXXXXXXXXXXXXXXXXXXXXXXXXXX",
    "operator": "SetFactoryDefault-Ack",
    "code": "200",
    "info": {
        "facesluiceId": "1306612",
        "reboot": "0",
        "result": "ok"
    }
}
```

7.5 Remote Open Door

1.Description

The remote door opening command sends corresponding door opening information from the server side to the face recognition machine, and the face recognition machine performs the door opening action and UI mention.

2.API Description

Items	Description
Interface Name	Remote open door
Downlink data topic	mqtt/face/ID（Where ID refers to the device ID/SN，e.g,:1306612）
Uplink data topic	mqtt/face/ID/Ack（Where ID refers to the device ID/SN，e.g,:1306612）

3.Remote Open Door Field Descriptions
Parameter information(Note:`Optional`is optional):

Key	Type	Values	Description
operator	String	Unlock	Remote open door
messageId	String		Message id, used by the platform to distinguish each message.
info	Object		Concrete content
facesluiceId	String		The clientID for connecting to the server,device ID/SN is used by default to ensure uniqueness
uid	String (optional)		Reserve
openDoor	int (optional)	0~1	Open door action 0: Do not open door 1: Open door
showInfo	String (optional)	64 bytes (including terminator)	UI tips

4.Example of Remote Open Door

```
{
    "operator": "Unlock",
    "messageId": "XXXXXXXXXXXXXXXXXXXXXXXXXXXXXX",
    "info": {
        "facesluiceId":"123456",
        "openDoor": 1,
        "showInfo": "Welcome"
    }
}
```

5.Device Response Field Descriptions

Parameter information(Note:`Optional`is optional):

Key	Type	Values	Description
operator		Unlock-Ack	Answer to remote open door
messageId	String		Message id, used by the platform to distinguish each message.
code	String		Command execution error code 200-successes,see also 10.29 Remote Open Door Error Codes
info	Object		Concrete content
facesluiceId	String		The clientID for connecting to the server,device ID/SN is used by default to ensure uniqueness
uid	String (optional)		Reserve
openDoor	String (optional)	0~1	Open door action 0: Do not open door 1: Open door
result	String		Operating result
detail	String (optional)		Error message when Result is "Fail"

6.Device Response Example

```
{
    "messageId": "xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx",
    "operator": "Unlock-Ack",
    "code": "200",
    "info": {
        "facesluiceId": "2121805",
        "uid": "00021",
        "openDoor": "1",
        "result": "ok"
    }
}
```

7.6 Device Keep Open Setting

1.Description

Keep the device always open state, at this time, swipe the card, identification will not close the door, can be used for fire escape and other scenarios.

2.API Description

Items	Description
Interface Name	Device keep open setting
Downlink data topic	mqtt/face/ID（Where ID refers to the device ID/SN，e.g,:1306612）
Uplink data topic	mqtt/face/ID/Ack（Where ID refers to the device ID/SN，e.g,:1306612）

3.Device Keep Open Setting Field Descriptions
Parameter information(Note: `Optional` is optional):

Key	Type	Values	Description
operator	String	SetKeepOpen	Device keep open setting
messageId	String		Message id, used by the platform to distinguish each message.
info	Object		Concrete content
KeepOpen	Int	0~2	Whether to set to normally open/closed mode, the interface needs to be called to actively close after setting 0: not enabled 1: door keep open 2: door keep closed

4.Example of Device Keep Open Setting

```
{
    "operator": "SetKeepOpen",
    "messageId": "XXXXXXXXXXXXXXXXXXXXXXXXXXXXXX",
    "info": {
        "KeepOpen": 1
    }
}
```

5.Device Response Field Descriptions
Parameter information(Note: `Optional` is optional):

Key	Type	Values	Description
operator		SetKeepOpen-Ack	Answer to device keep open setting
messageId	String		Message id, used by the platform to distinguish each message.
code	String		Command execution error code 200-successes,see also 10.30 Device Keep Open Setting Error Codes
info	Object		Concrete content
facesluiceId	String		The clientID for connecting to the server,device ID/SN is used by default to ensure uniqueness
result	String		Operating result
detail	String (optional)		Error message when Result is "Fail"

6.Device Response Example

```
{
    "messageId": "XXXXXXXXXXXXXXXXXXXXXXXXXXXXXX",
    "operator": "SetKeepOpen-Ack",
    "code": "200",
    "info": {
        "facesluiceId": "2121805",
        "result": "ok"
    }
}
```

8.Device Parameters

The device parameter setting section deals with parameters such as face machine operation and display.

8.1 System Parameters

The system parameters are mainly related to the system operation parameters of the face machine.

8.1.1 Getting System Parameters

1.Description

The system parameters are mainly related to the system operation parameters of the face machine.

2.API Description

Items	Description
Interface Name	Get the system parameters.
Data flow (active command side)	Platform->Device
Downlink data topic	mqtt/face/ID（Where ID refers to the device ID/SN，e.g,:1306612）
Uplink data topic	mqtt/face/ID/Ack（Where ID refers to the device ID/SN，e.g,:1306612）

3.Getting System Parameters Field Descriptions
Parameter information(Note:`Optional`is optional):

Key	Type	Values	Description
operator	String	Getconfig	Get the system parameters.
messageId	String		Message id, used by the platform to distinguish each message.
info	Object		Concrete content

4.Example of Getting System Parameters

```
{
    "messageId": "xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx",
    "operator": "Getconfig",
    "info": {}
}
```

5.Device Response Field Descriptions

Key	Type	Values	Description
operator		Getconfig-Ack	Answer to get the system parameters
messageId	String		Message id, used by the platform to distinguish each message.
code	String		Command execution error code 200-successes,see alsoerror cross-reference table
info	Object		Concrete content
facesluiceId	String		The clientID for connecting to the server,device ID/SN is used by default to ensure uniqueness
Language	String	0~16	Device Language:Universal versions only support 0~2 by default 0:English 1:Chinese Simplified 2:Chinese Traditional 3:Portuguese 4:Korean 5:Russian (supported by special versions) 6:Italian 7:French 8:Spanish 9:Japanese 10:Hebrew 11:Thai 12:Slovak 13:Polish 14:Serbian 15:Czech 16:Vietnamese Changing the device language will reboot the machine.
DataBaseEnable	String	0~1	Whether to save record capture+record ID+record authentication 0:No 1:Yes Default:1, if not turned on, the capture, ID and authentication image data will not be saved.
IDcardType	String	Reserve	Reserve

Key	Type	Values	Description
FaceDisplay	String	0~1,default 1	Whether to display face frame 0:No 1:Yes
LiveDetectType	String	0~2,default 1	Live detection mode 0: live detection always on 1: live detection off 2: live detection on by time period
LiveDetectTimeBeg	String	e.g. 07:00:00	Live detect daily start point Required when LiveDetectType=2
LiveDetectTimeEnd	String	e.g. 19:00:00	Live detect daily end point Required when LiveDetectType=2
LiveThreshold	String	e.g. 90.000000	Live detection threshold
LiveFrameNum	String		The number of consecutive frames for live body detection, the larger the number of frames, the higher the recognition occupation time.
LedLightType	String	0~5	Enable White Light Type 0:Never 1:Time Control 2:Light Sensitive Control 3:Face Sense 4:Face or Time 5:Face or Light Sense
LedBrightness	String	1~100	White light brightness
LedTimeBeg	String	e.g. 19:00:00	White light activation start time per day (white light activation time) required when LedLightType=1,4
LedTimeEnd	String	e.g. 07:00:00	White light activation end time per day (white light off time) required when LedLightType=1,4
LedDisableAfterSec	String	(1~600) Default:30	How many seconds after no one is there to turn off the white light (face sensing) required when LedLightType=3,4,5 (Unit: second)

Key	Type	Values	Description
LcdBLDisable	String		Whether to turn off the screen when no one is present 0: Never 1: Turn off the screen display when no one is around
LcdBLDisableAfterSec	String	(10s~600) Default:30	Disable screen when no pass after seconds (Unit: second) required when LcdBLDisable=1 (Unit: second)
ScreenBrightness	String	Default:50	Screen brightness (Reserve)
WebTimeOut	String	2~10 Default:5	Web page login timeout (Unit: minute)
SceneSnap	String	0~1	Whether to enable scene image capture
NightMode	int	0-2	Night mode on type 0: Off 1: On by Time 2: All Day On
NightTimeBeg	String	e.g. 19:00:00	Night start-up time point required when NightMode=1
NightTimeEnd	String	e.g. 19:00:00	End-of-night time point required when NightMode=1
result	String	"ok"/"fail"	Operating result
detail	String (optional)		Error message when Result is "Fail"

6.Device Response Example


```
{
    "messageId": "xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx",
    "operator": "Getconfig-Ack",
    "code": "200",
    "info": {
        "facesluiceId": "1305433",
        "Language": "1",
        "DataBaseEnable": "1",
        "IDcardType": "1",
        "CardReaderType": "1",
        "CardOpenDoorWay": "0",
        "FaceDisplay": "1",
        "LiveDetectType": "0",
```

```
        "LiveDetectTimeBeg": "07:00:00",
        "LiveDetectTimeEnd": "19:00:00",
        "LiveThreshold": "90.340000",
        "LiveFrameNum": "2",
        "LedLightType": "0",
        "LedTimeBeg": "07:00:00",
        "LedTimeEnd": "19:00:00",
        "LedBrightness": "50",
        "LedDisableAfterSec": "30",
        "LcdBLDisable": "0",
        "LcdBLDisableAfterSec": "30",
        "ScreenBrightness": "50",
        "WebTimeOut" : "5",
        "SceneSnap" : "0",
        "NightMode" : 1,
        "NightTimeBeg": "19:00:00",
        "NightTimeEnd": "23:00:00",
        "result": "ok"
    }
}
```

8.1.2 Setting System Parameters

1.Description

Set the system parameters. Changing the face machine language option will reboot the device, the default universal version does not support all languages.

2.API Description

Items	Description
Interface Name	Set the system parameters.
Data flow (active command side)	Platform->Device
Downlink data topic	mqtt/face/ID（Where ID refers to the device ID/SN，e.g,:1306612）
Uplink data topic	mqtt/face/ID/Ack（Where ID refers to the device ID/SN，e.g,:1306612）

3.Setting System Parameters Field Descriptions
Parameter information(Note:`Optional`is optional):

Key	Type	Values	Description
operator		Upconfig	Set the system parameters.
messageId	String		Message id, used by the platform to distinguish each message.
info	Object		Concrete content
Language	int (optional)	0~16	Device Language:Universal versions only support 0~2 by default 0:English 1:Chinese Simplified 2:Chinese Traditional 3:Portuguese 4:Korean 5:Russian (supported by special versions) 6:Italian 7:French 8:Spanish 9:Japanese 10:Hebrew 11:Thai 12:Slovak 13:Polish 14:Serbian 15:Czech 16:Vietnamese Changing the device language will reboot the machine.
DataBaseEnable	int (optional)	0~1	Whether to save record capture+record ID+record authentication 0:No 1:Yes Default:1, if not turned on, the capture, ID and authentication image data will not be saved.
IDcardType	int (optional)	Reserve	Reserve
FaceDisplay	int (optional)	0~1 , Default 1	Whether to display face frame 0:No 1:Yes

Key	Type	Values	Description
LiveDetectType	int (optional)	0~2 , Default 1	Live detection mode 0: live detection always on 1: live detection off 2: live detection on by time period
LiveDetectTimeBeg	String (optional)	e.g. 07:00:00	Live detect daily start point Required when LiveDetectType=2
LiveDetectTimeEnd	String (optional)	e.g. 19:00:00	Live detect daily end point Required when LiveDetectType=2
LedLightType	int (optional)	0~5	Enable White Light Type 0:Never 1:Time Control 2:Light Sensitive Control 3:Face Sense 4:Face or Time 5:Face or Light Sense
LedBrightness	int (optional)	1~100	White light brightness
LedTimeBeg	String (optional)	e.g. 19:00:00	White light activation start time per day (white light activation time) required when LedLightType=1,4
LedTimeEnd	String (optional)	e.g. 07:00:00	White light activation end time per day (white light off time) required when LedLightType=1,4
NightMode	int (optional)	0~2	Night mode on type 0: Off 1: On by Time 2: All Day On
NightTimeBeg	String (optional)	e.g. 19:00:00	Night start-up time point required when NightMode=1
NightTimeEnd	String (optional)	e.g. 19:00:00	End-of-night time point required when NightMode=1
LedDisableAfterSec	int (optional)	(1~600) Default :30	How many seconds after no one is there to turn off the white light (face sensing) required when LedLightType=3,4,5 (Unit: second)

Key	Type	Values	Description
LcdBLDisable	int (optional)		Whether to turn off the screen when no one is present 0: Never 1: Turn off the screen display when no one is around
LcdBLDisableAfterSec	int (optional)	(10s~600) Default :30	Disable screen when no pass after seconds （Unit: second） required when LcdBLDisable=1（Unit: second）
BodySensorEnable	int (optional)	0~1	Whether to turn on the body sensor 0:Turn on 1:Turn off
ScreenBrightness	int (optional)	Default :50	Screen brightness(Reserve)
WebTimeOut	int (optional)	2~10 Default :5	Web page login timeout（Unit: minute）
SceneSnap	int	0~1	Whether to enable scene image capture.

4.Example of Setting System Parameters

```
{
    "messageId": "xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx",
    "operator": "Upconfig",
    "info": {
        "LedLightType": 1,
        "LedTimeBeg": "20:00:00",
        "LedTimeEnd": "08:00:00",
        "LedBrightness": 60
    }
}
```

5.Device Response Field Descriptions

Key	Type	Values	Description
operator		Upconfig-Ack	Answer to set the system parameters.
messageId	String		Message id, used by the platform to distinguish each message.
code	String		Command execution error code 200-successes,see also 10.31 Setting System Parameters Error Codes
info	Object		Concrete content
facesluiceId	String		The clientID for connecting to the server,device ID/SN is used by default to ensure uniqueness
reboot	int		Whether the device reboots automatically 0: Does not reboot 1: Automatic reboot
result	String		Operating result
detail	String (optional)		Error message when Result is "Fail"

6.Device Response Example

```
{
    "messageId": "XXXXXXXXXXXXXXXXXXXXXXXXXXXXXX",
    "operator": "Upconfig-Ack",
    "code": "200",
    "info": {
        "facesluiceId": "1305433",
        "reboot": "0",
        "result": "ok"
    }
}
```

8.2 Prompt Sound and Interface Display Parameters

This interface is used for the sound playback of the face machine, display of UI elements and other major parameter settings.

8.2.1 Getting Parameters of the Prompt Sound and the Interface Display

1.Description

Get the main parameters for controlling the sound playback control of the face machine, display of UI elements, and so on.

2.API Description

Items	Description
Interface Name	Get the sound and display parameters.
Data flow (active command side)	Platform->Device
Downlink data topic	mqtt/face/ID（Where ID refers to the device ID/SN，e.g,:1306612）
Uplink data topic	mqtt/face/ID/Ack（Where ID refers to the device ID/SN，e.g,:1306612）

3.Getting the Parameters of the Prompt Sound and the Interface Display Field Descriptions
Parameter information(Note:`Optional`is optional):

Key	Type	Values	Description
operator	String	GetSoundconfig	Get the “sound and display” parameter.
messageId	String		Message id, used by the platform to distinguish each message.
info	Object		Concrete content

4.Example of Getting the Parameters of the Prompt Sound and the Interface Display

```
{
    "messageId": "xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx",
    "operator": "GetSoundconfig",
    "info": {}
}
```

5.Device Response Field Descriptions

Key	Type	Values	Description
operator		GetSoundconfig-Ack	Answer to Get the sound and display parameters.
messageId	String		Message id, used by the platform to distinguish each message.
code	String		Command execution error code 200-successes,see alsoerror cross-reference table
info	Object		Concrete content
facesluiceId	String		The clientID for connecting to the server,device ID/SN is used by default to ensure uniqueness
VerifySuccAudio	String	0~1	Whether or not to broadcast sound after successful authentication 0: No broadcast 1: Broadcast
VerifyFailAudio	String	0~1	Whether or not to broadcast sound when authentication fails 0: No broadcast 1: Broadcast
Volume	String	0~100	Volume value
VerifySuccGuiTip	String	0~1	Whether or not the authentication success screen prompts 0: no prompt 1: prompt
VerifyFailGuiTip	String	0~1	Whether or not the authentication failure screen prompts 0: No prompt 1: Prompt
UnregisteredGuiTip	String	0~1	Is the list not registered screen prompted 0: Not prompted 1: Prompted
RemoteCtrlAudio	String	0~1	Remote control of voice announcements 0: No broadcast 1: Broadcast

Key	Type	Values	Description
IPHide	String	0~1	Whether the interface IP is hidden or not 0: Not hidden 1: Hidden
IsShowName	String	0~2	Successful match, name prompts whether to display 0: not displayed 1: displayed 2: partially displayed (hide the latter part)
DisplayPicture	String	0~1	Images prompted by successful comparison 0: Captured image 1: Registered image
IsShowDeviceID	String	0~1	Whether to display the local ID 0: Do not show 1: Show
IsShowPersonNum	String	0~1	Whether to display the number of registered lists 0: Not displayed 1: Displayed
VerifyTipContent	String	User-defined	Successful authentication UI tips (length of 32 bytes or less, not too long, or lead to incomplete UI display) Default: “Access granted!”
UnregisterTipContent	String	User-defined	List Unregistered UI tips (32 bytes or less in length, otherwise it causes the UI to display incomplete) Default: “Stranger!”
BlacklistTipContent	String	User-defined	Authentication Failure UI Text Alert (within 32 bytes in length, otherwise it will cause the UI display to be incomplete) Default: “Access denied!”
CompanyName	String	User-defined	Company name (length of 64 bytes or less)
ICCardNumHide	String	0~1	Whether to hide the card number in the interface**(Built-in card models)** 0: Not hidden 1: Hidden

Key	Type	Values	Description
IDCardNumHide	String (optional)	0~1	Whether to hide the IC card number in the interface**(Supported except for built-in card models)** 0: Not hidden 1: Hidden
bTempLowAudio	String (optional)	0~1	Whether to turn on low-temperature broadcasting 0: No broadcast 1: Broadcast
bTempDistance	String (optional)	0~1	Whether to turn on the temperature measurement distance prompt
DoorContact	String (optional)	0~1	Whether to turn on the door magnet （requires hardware support）
bIsShowTamperAlarm	int (optional)	0~1	Whether tamper-evident is configured (requires hardware support)
CompanyLogoShow	int (optional)	0~1	Whether to display the company logo
CompanyLogo_X	int (optional)		When CompanyLogoShow=1, the horizontal coordinate of the logo.
CompanyLogo_Y	int (optional)		When CompanyLogoShow=1, the vertical coordinate of the logo.
CompanyLogo_H	int (optional)		When CompanyLogoShow=1, the height of the logo.
CompanyLogo_W	int (optional)		When CompanyLogoShow=1, the width of the logo.
result	String	"ok"/"fail"	Operating result
detail	String (optional)		Error message when Result is "Fail"

6.Device Response Example


```
{
    "messageId": "XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX",
    "operator": "GetSoundconfig-Ack",
    "code": "200",
    "info": {
        "facesluiceId": "1305433",
```

```
        "VerifySuccAudio": "1",
        "VerifyFailAudio": "1",
        "Volume": "90",
        "VerifySuccGuiTip": "1",
        "VerifyFailGuiTip": "1",
        "UnregisteredGuiTip": "1",
        "DisplayPicture": "1",
        "RemoteCtrlAudio": "0",
        "IPHide": "0",
        "IsShowName": "1",
        "IsShowTitle": "0",
        "IsShowVersion": "0",
        "IsShowDate": "1",
        "IsShowTime": "1",
        "ICCardNumHide": "0",
        "VerifyTipContent": "Access granted!",
        "UnregisterTipContent": "Stranger!",
        "BlacklistTipContent": "Access denied!",
        "CompanyName": "XXX Technology Co.",
        "CompanyLogoShow": 0,
        "result": "ok"
    }
}
```

8.2.2 Setting the Parameters of the Sound of the Prompt and the Interface Display

1.Description

This interface is used for sound playback control of the face machine, display of UI elements and other parameter settings.

2.API Description

Items	Description
Interface Name	Set the Prompt sound and interface display parameters
Data flow (active command side)	Platform->Device
Downlink data topic	mqtt/face/ID（Where ID refers to the device ID/SN，e.g,:1306612）
Uplink data topic	mqtt/face/ID/Ack（Where ID refers to the device ID/SN，e.g,:1306612）

3.Setting the Parameters of the Prompt Sound and the Interface DisplayField Descriptions

Parameter information(Note:`Optional`is optional):

Key	Type	Values	Description
operator		UpSoundconfig	Setting the prompt sound and interface display parameters
messageId	String		Message id, used by the platform to distinguish each message.
info	Object		Concrete content
facesluiceId	String		The clientID for connecting to the server,device ID/SN is used by default to ensure uniqueness
VerifySuccAudio	int (optional)	0~1	Whether or not to broadcast sound after successful authentication 0: No broadcast 1: Broadcast
VerifyFailAudio	int (optional)	0~1	Whether to broadcast sound when authentication fails 0: No broadcast 1: Broadcast
Volume	int (optional)	0~100	Volume value
VerifySuccGuiTip	int (optional)	0~1	Whether or not the authentication success screen prompts 0: no prompt 1: prompt
VerifyFailGuiTip	int (optional)	0~1	Whether or not the authentication failure screen prompts 0: No prompt 1: Prompt
UnregisteredGuiTip	int (optional)	0~1	Is the list not registered screen prompted 0: Not prompted 1: Prompted
DisplayPicture	int (optional)	0~1	Whether the interface IP is hidden or not 0: Not hidden 1: Hidden
RemoteCtrlAudio	int (optional)	0~1	Remote control of voice announcements 0: No broadcast 1: Broadcast

Key	Type	Values	Description
IPHide	int (optional)	0~1	Whether the interface IP is hidden or not 0: Not hidden 1: Hidden
IsShowName	int (optional)	0~2	Successful match, name prompts whether to display 0: not displayed 1: displayed 2: partially displayed (hide the latter part)
IsShowDeviceID	int (optional)	0~1	Whether to display the local ID 0: Do not show 1: Show
IsShowPersonNum	int (optional)	0~1	Whether to display the number of registered lists 0: Not displayed 1: Displayed
bIsShowTamperAlarm	int (optional)	0~1	Whether tamper-evident is configured (requires hardware support)
IDCardNumHide	int (optional)	0~1	Whether to hide the card number in the interface**(Built-in card models)** 0: Not hidden 1: Hidden
ICCardNumHide	int (optional)	0~1	Whether to hide the IC card number in the interface**(Supported except for built-in card models)** 0: Not hidden 1: Hidden
DoorContact	int (optional)	0~1	Whether to turn on the door magnet (requires hardware support)
VerifyTipContent	String (optional)	User-defined	Successful authentication UI tips (length of 32 bytes or less, not too long, or lead to incomplete UI display) Default: “Access granted!”
UnregisterTipContent	String (optional)	User-defined	List Unregistered UI tips (32 bytes or less in length, otherwise it causes the UI to display incomplete) Default: “Stranger!”

Key	Type	Values	Description
BlacklistTipContent	String (optional)	User-defined	Authentication Failure UI Text Alert (within 32 bytes in length, otherwise it will cause the UI display to be incomplete) Default: “Access denied!”
CompanyName	String (optional)	User-defined	Company name (length of 64 bytes or less)
bTempLowAudio	int (optional)	0~1	Whether to turn on low-temperature broadcasting 0: No broadcast 1: Broadcast
bTempDistance	int (optional)	0~1	Whether to turn on the temperature measurement distance prompt （Available for 1st and 3rd generation thermometry）
CompanyLogoShow	int (optional)	0~1	Whether to display the company logo 0:Do not show 1:Show
CompanyLogo_X	int (optional)	User-defined	When CompanyLogoShow=1, the horizontal coordinate of the logo.
CompanyLogo_Y	int (optional)	User-defined	When CompanyLogoShow=1, the vertical coordinate of the logo.
CompanyLogo_H	int (optional)	User-defined	When CompanyLogoShow=1, the height of the logo.
CompanyLogo_W	int (optional)	User-defined	When CompanyLogoShow=1, the width of the logo.

4.Example of Setting the Parameters of the Prompt Sound and the Interface Display

```
{
    "messageId": "XXXXXXXXXXXXXXXXXXXXXXXXXXXXXX",
    "operator": "UpSoundconfig",
    "info": {
        "Volume": 95,
        "UnregisteredGuiTip": 1
    }
}
```

5.Device Response Field Descriptions

Key	Type	Values	Description
operator		UpSoundconfig-Ack	Answer to Setting the prompt sound and interface display parameters
messageId	String		Message id, used by the platform to distinguish each message.
code	String		Command execution error code 200-successes,see also 10.32 Setting the Parameters of the Sound of the Prompt and the Interface Display Error Codes
info	Object		Concrete content
facesluiceId	String		The clientID for connecting to the server,device ID/SN is used by default to ensure uniqueness
result	String		Operating result.
detail	String (optional)		Error message when Result is "Fail"

6.Device Response Example

```
{
    "messageId": "XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX",
    "operator": "UpSoundconfig-Ack",
    "code": "200",
    "info": {
        "facesluiceId": "1362003",
        "result": "ok"
    }
}
```

8.3 Door Opening Conditions and Output Control Parameters

Door opening condition and output control parameter part mainly involves controlling the main parameter settings such as door opening condition and output control of the face machine, which can be adjusted according to the actual situation to realize different door opening condition to open the door.

8.3.1 Obtaining Door Opening Conditions and Outputting Control Parameters

1.Description

The main purpose of this interface is to obtain the main parameters involving the opening conditions and output control of the face machine.

2.API Description

Items	Description
Interface Name	Obtaining door opening conditions and outputting control parameters.
Data flow (active command side)	Platform->Device
Downlink data topic	mqtt/face/ID（Where ID refers to the device ID/SN，e.g,:1306612）
Uplink data topic	mqtt/face/ID/Ack（Where ID refers to the device ID/SN，e.g,:1306612）

3.Obtaining Door Opening Conditions and Outputting Control Parameters Field Descriptions
Parameter information(Note:`Optional`is optional):

Key	Type	Values	Description
operator	String	GetDoorconfig	Obtaining door opening conditions and outputting control parameters.
messageId	String		Message id, used by the platform to distinguish each message.
info	Object		Concrete content

4.Example of Obtaining Door Opening Conditions and Outputting Control Parameters

```
{
    "messageId": "XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX",
    "operator": "GetDoorconfig",
    "info": {}
}
```

5.Device Response Field Descriptions

Key	Type	Values	Description
operator		GetDoorconfig-Ack	Answer to obtaining door opening conditions and outputting control parameters.
messageId	String		Message id, used by the platform to distinguish each message.
code	String		Command execution error code 200-successes,see alsoerror cross-reference table
info	Object		Concrete content
facesluiceId	String		The clientID for connecting to the server,device ID/SN is used by default to ensure uniqueness
OpendoorWay	String	0~3	0:Face door opening method; 1:Remote door opening method; 2:Remote door opening or face method; 3:Face verification + remote door opening method
FaceThreshold	String	50~100	Black and White List comparison threshold
IDCardThreshold	String	50~100	ID card comparison threshold
VerifyMode	String	1:Universal model 1~4 or 21~26； 2.Masks or thermometry model: Whether mask testing or temperature testing is required Support for mask detection or (|) operation 0x100 indicates a pass with mask detection; Supports body temperature detection or (|) operation 0x200 indicates a pass with temperature detection	Authentication Type 1: Whitelist Verification 2: ID Verification 3: Whitelist + ID Verification 4: Whitelist or ID Verification 5:Separate mask or separate body temperature or separate mask + body temperature (to be used in conjunction with whether or not to include mask detection or body temperature detection, mask or temperature measurement models) 21:RF Card Verification(**Built-in card swiping model**) 22:RF Card Verification + Whitelist Verification(**Built-in card swiping model**)

Key	Type	Values	Description
		Support mask + body temperature detection or (|) operation 0x300 means with mask + temperature detection pass Example: 1:Universal model 1)Whitelist validation:1 2)Wergand card + whitelist validation:25 2:Mask or temperature measuring machine 1)Separate mask validation pass (0x105=0x100&5=261) 2)Mask + whitelist validation (0x101=0x100&1=257) 3)Separate body temperature pass (0x205=0x200&5=517) 4)Body temperature + whitelist verification (0x201=0x200&1=513) 5)Mouthpiece + body temperature alone (0x305=0x300&5=773) 6)Mouthpiece + body temperature + Wigan swipe + whitelist verification (0x319=0x300&25=793)	23:RF card authentication or white list authentication (built-in card model) 24:Wiegand Card Verification 25:Wiegand Card + Whitelist Verification 26:Wiegand Card or Whitelist Authentication The following door opening methods are only supported by the health code version: 47:QR code or whitelist or ID card 48:whitelist + QR code or whitelist + ID card 55:QR code or ID card or whitelist (to check the health code) 56:QR code or ID card to skip the swipe 57:whitelist + health code information
IsMaskOK	String	0~1 default 0	Permission to pass without a mask, used in conjunction with mask testing (supported by special versions) 0: Not allowed 1: Allowed

Key	Type	Values	Description
IsCardOK	String	0~1 default 0	Unregistered card numbers allowed to pass, used in conjunction with swipe cards(supported by special versions) 0: Not allowed 1: Allowed
VerifyResetTime	String	1~10	Verification reset time (sec) UI displays a prompt message as well as the time that the same person is not recognized again during the verification reset time.
SnapResetTime	String	0-15 (default 0)	Recognition interval (seconds):Waiting time to recognize the next person after the comparison is passed.Default 0, does not affect access efficiency .
Wiegand	String	0~1 or 4~7	Wiegand protocol type 0: 26 bits 1: 34 bits 4: 26 bits(8+16): facility code+userid 5: 34 bits(8+24); 6: 26 bits(8+16 fill in separately); 7: 34 bits(8+24 fill in separately);
PublicMjCardNo	String		Public Access Card Number
AutoMjCardBgnNo	String		Starting card number for automatic generation of access card number
AutoMjCardEndNo	String		Automatic generation of end card numbers for access control card number
ControlType	String	0~2	Controls the way the door opener interfaces. 0:Wiegand interface 1:switching mode 2:Wiegand interface+switching mode
IOType	String	0~1	Door opening action when controlling the door opening interface method 0:close 1:turn off
IOStayTime	String	0~50000	Open Door Hold Time (ms)

Key	Type	Values	Description
DoorContactTime	String		Door magnetic trigger hold time (s) When the door magnet is triggered, there is no alarm within the time period, and the alarm will be triggered after the time period.
Endian	String	0~1	Card number big/little endian mode 0:Big Endian 1:Little Endian
CardMode	String	0~1	Card Number Method 0:decimal 1:hexadecimal
IsOutFF	String	0~1	Stranger Wiegand output FFFFFFFF (Wiegand opens door) 0:non-output 1:output
KeepCardZero	String	0~1	Whether or not to report and display the 0 in front of the reserved card number 0: not retained 1: retained
PasswordEnable	int (optional)	0~1	Whether to enable password detection (device must have password function)
PasswordType	int (optional)	0~1	Password Detection Method: 0:common password 1:personal password (The device needs to have a password function)
keyPassword	String (option)		When PasswordType=0, the common password is a 6-digit number (the device must be equipped with a password function)
result	String	"ok"/"fail"	Operating result
detail	String (optional)		Error message when Result is "Fail"

6.Device Response Example

```
{
    "messageId": "xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx",
```

```
    "operator": "GetDoorconfig-Ack",
    "code": "200",
    "info": {
        "facesluiceId": "1305433",
        "OpendoorWay": "0",
        "FaceThreshold": "85",
        "IDCardThreshold": "50",
        "VerifyMode": "1",
        "IsMaskOK": "0",
        "IsCardOK": "0",
        "VerifyResetTime": "2",
        "SnapResetTime": "0",
        "Wiegand": "1",
        "PublicMjCardNo": "1",
        "AutoMjCardBgnNo": "1000",
        "AutoMjCardEndNo": "9999",
        "ControlType": "2",
        "IOType": "1",
        "IOStayTime": "200",
        "Endian": "0",
        "CardMode": "0",
        "IsOutFF": "0",
        "result": "ok"
    }
}
```

8.3.2 Setting Door Opening Conditions and Outputting Control Parameters

1.Description

This interface is used to set the main parameters involving the door opening conditions and output control of the face machine.

2.API Description

Items	Description
Interface Name	Set the parameters of door opening conditions and output control.
Data flow (active command side)	Platform->Device
Downlink data topic	mqtt/face/ID（Where ID refers to the device ID/SN，e.g,:1306612）
Uplink data topic	mqtt/face/ID/Ack（Where ID refers to the device ID/SN，e.g,:1306612）

3.Setting Door Opening Conditions and Outputting Control Parameters Field Descriptions

Parameter information(Note:`Optional`is optional):

Key	Type	Values	Description
operator		UpDoorconfig	Set the parameters of door opening conditions and output control.
messageId	String		Message id, used by the platform to distinguish each message.
info	Object		Concrete content
OpendoorWay	int (optional)	0~3	0:Face door opening method; 1:Remote door opening method; 2:Remote door opening or face method; 3:Face verification + remote door opening method
FaceThreshold	int (optional)	50~100	Black and White List comparison threshold
IDCardThreshold	int (optional)	50~100	ID card comparison threshold
IsMaskOK	int (optional)	0~1，Default 0	Permission to pass without a mask, used in conjunction with mask testing (supported by special versions) 0: Not allowed 1: Allowed
IsCardOK	int (optional)	0~1，Default 0	Unregistered card numbers allowed to pass, used in conjunction with swipe cards(supported by special versions) 0: Not allowed 1: Allowed
VerifyMode	int (optional)	1:Universal model 1~4 or 21~26； 2.Masks or thermometry model: Whether mask testing or temperature testing is required Support for mask detection or (|) operation 0x100 indicates a pass with mask detection; Supports body	Authentication Type 1: Whitelist Verification 2: ID Verification 3: Whitelist + ID Verification 4: Whitelist or ID Verification 5:Separate mask or separate body temperature or separate mask + body temperature (to be used in conjunction with whether or not to include mask detection or body temperature detection, mask or temperature measurement models) 21:RF Card Verification(**Built-in card**

Key	Type	Values	Description
		temperature detection or (|) operation 0x200 indicates a pass with temperature detection Support mask + body temperature detection or (|) operation 0x300 means with mask + temperature detection pass Example: 1:Universal model 1)Whitelist validation:1 2)Wergand card + whitelist validation:25 2:Mask or temperature measuring machine 1)Separate mask validation pass (0x105=0x100&5=261) 2)Mask + whitelist validation (0x101=0x100&1=257) 3)Separate body temperature pass (0x205=0x200&5=517) 4)Body temperature + whitelist verification (0x201=0x200&1=513) 5)Mouthpiece + body temperature alone (0x305=0x300&5=773) 6)Mouthpiece + body temperature + Wigan swipe + whitelist verification (0x319=0x300&25=793)	**swiping model**) 22:RF Card Verification + Whitelist Verification(**Built-in card swiping model**) 23:RF card authentication or white list authentication (built-in card model) 24:Wiegand Card Verification 25:Wiegand Card + Whitelist Verification 26:Wiegand Card or Whitelist Authentication
VerifyResetTime	int (optional)	1~10	Verification reset time (sec) UI displays a prompt message as well as the time that the same person is not recognized again during the verification reset time.

Key	Type	Values	Description
SnapResetTime	int (optional)	0-15 default 0	Recognition interval (seconds):Waiting time to recognize the next person after the comparison is passed.Default 0, does not affect access efficiency .
PasswordEnable	int (optional)	0~1	Whether to enable password detection (device must have password function)
PasswordType	int (optional)	0~1	Password Detection Method: 0:common password 1:personal password （The device needs to have a password function）
keyPassword	String (optional)		PasswordType=0, the common password is a 6-digit number (the device must be equipped with a password function)
Wiegand	int (optional)	0~1 or 4~7	Wiegand protocol type 0: 26 bits 1: 34 bits 4: 26 bits(8+16): facility code+userid 5: 34 bits(8+24); 6: 26 bits(8+16 fill in separately); 7: 34 bits(8+24 fill in separately);
PublicMjCardNo	int (optional)		Public Access Card Number
AutoMjCardBgnNo	int (optional)		Starting card number for automatic generation of access card number
AutoMjCardEndNo	int (optional)		Automatic generation of end card numbers for access control card number
ControlType	int (optional)	0~2	Controls the way the door opener interfaces. 0:Wiegand interface 1:switching mode 2:Wiegand interface+switching mode
IOType	int (optional)	0~1	Door opening action when controlling the door opening interface method 0:close 1:turn off

Key	Type	Values	Description
IOStayTime	int (optional)	0~50000 Default 200(ms)	Open Door Hold Time (ms)
DoorContactTime	int (optional)	0~30	Door magnetic trigger hold time (s) When the door magnet is triggered, there is no alarm within the time period, and the alarm will be triggered after the time period.
Endian	int (optional)	0~1	Card number big/little endian mode 0:Big Endian 1:Little Endian
CardMode	int (optional)	0~1	Card Number Method 0:decimal 1:hexadecimal
IsOutFF	int (optional)	0~1	Stranger Wiegand output FFFFFFFF (Wiegand opens door) 0:non-output 1:output
KeepCardZero	int (optional)	0~1	Whether or not to report and display the 0 in front of the reserved card number 0: not retained 1: retained

4.Example of Setting Door Opening Conditions and Outputting Control Parameters

```
{
    "messageId": "XXXXXXXXXXXXXXXXXXXXXXXXXXXXXX",
    "operator": "UpDoorconfig",
    "info": {
        "FaceThreshold": 90,
        "VerifyMode": 4
    }
}
```

5.Device Response Field Descriptions

Key	Type	Values	Description
operator		UpDoorconfig-Ack	Answer to set the parameters of door opening conditions and output control.
messageId	String		Message id, used by the platform to distinguish each message.
code	String		Command execution error code 200-successes,see also 10.33 Setting Door Opening Conditions and Outputting Control Parameters Error Codes
info	Object		Concrete content
facesluiceId	String		The clientID for connecting to the server,device ID/SN is used by default to ensure uniqueness
result	String		Operating result.
detail	String (optional)		Error message when Result is "Fail"

6.Device Response Example

```
{
    "messageId": "XXXXXXXXXXXXXXXXXXXXXXXXXXXXXX",
    "operator": "UpDoorconfig-Ack",
    "code": "200",
    "info": {
        "facesluiceId": "1362003",
        "result": "ok"
    }
}
```

8.3.3 Introduction to Door Opening Methods

At present, the face recognition machine supports the following door opening methods: face door opening method, remote door opening method, remote door opening or face method, face verification + remote door opening method.

0:Face opening mode:Face recognition all-in-one machine will push the stranger capture information and authentication comparison information, after passing the comparison, it will directly execute the door opening and UI prompting information, in this mode, calling the remote door opening instruction will be invalid.

1:Remote door opening mode: face recognition all-in-one machine only push the stranger capture information, do not push the authentication comparison information, **does not produce a control log**, the comparison through the door will not be directly executed by opening the door and the UI information prompts, this mode, you can call the remote door opening instructions, this mode is mainly used for the front-end capture, the

back-end comparison, and the back-end implementation of the remote door opening.

2:Remote door opening or face mode:face recognition all-in-one machine will push the stranger capture information and authentication comparison information, and directly execute the door opening and UI prompt information after the comparison passes, in this mode, you can call the remote door opening instruction.

3:Face verification + remote door opening mode: face recognition all-in-one machine will push the stranger capture information and authentication comparison information, the comparison through the door will not be directly implemented to open the door and UI information prompts, open the door information and UI prompts need to be in the reply to the authentication push packet received in response to open the door information and UI prompts. This mode requires HTTP settings to be used **in the** `Continue Transmitting After Disconnection` **mode** and **the authentication subscription is turned on**. At this time, the face recognition all-in-one machine in the case of matching passes, the authentication matching information will be pushed to the HTTP subscription address (platform), the platform needs to receive the authentication subscription of the push information, and then reply to the `Continue Transmitting After Disconnection` json data to confirm the packet directly with the open door and prompt information, the all-in-one machine to directly implement the open door and the UI information prompts, do not need to call the remote open the door command. In this mode, you need to pay attention to exclude whether the push authentication data is `Continue Transmitting After Disconnection` authentication data, if it is `Continue Transmitting After Disconnection` data and remote door opening authentication data, then the reply packet does not need to perform the door opening action again.This mode may have some latency due to network problems and the time taken by the interaction data, resulting in the door not being opened in time.

Example of an information confirmation packet received from an authentication subscription using Face Verification + Remote Door Opening when `Continue Transmitting After Disconnection` is enabled:

Parameter information(Note:`Optional`is optional): :

Key	Type	Values	Description
PushAckType	int	2	2:Responding to Received Authentication Record Information
SnapOrRecordID	int		Fill in the received control log library ID when PushAckType=2
openDoor	int (optional)		Open door action 0: Do not open door 1: Open door
showInfo	String (optional)	64bytes (including terminator)	UI tips

e.g.1,Response needs to open the door+UI tips

```
{
    "messageId": "XXXXXXXXXXXXXXXXXXXXXXXXXXXXXX",
    "operator": "PushAck",
    "info": {
        "PushAckType": 2,
        "SnapOrRecordID": 381,
        "openDoor": 1,
        "showInfo": "XXX, please"
    }
}
```

e.g.2,ReplyNo need to open the door+UI tips

```
{
    "messageId": "XXXXXXXXXXXXXXXXXXXXXXXXXXXXXX",
    "operator": "PushAck",
    "info": {
        "PushAckType": 2,
        "SnapOrRecordID": 382,
        "openDoor": 0,
        "showInfo": "XX is not authorized to pass"
    }
}
```

8.4 HTTP Subscription Parameters

The HTTP Subscription Parameters section is mainly concerned with obtaining and setting HTTP subscription parameters via the MQTT protocol (WAN). This section mainly considers that if the device is in an intranet environment, it is not possible to modify and set HTTP subscription server parameters and other information via the HTTP protocol.

8.4.1 Getting HTTP Subscription Parameters

1.Description

Get the HTTP subscription parameters.

2.API Description

Items	Description
Interface Name	Get the HTTP subscription parameters.
Data flow (active command side)	Platform->Device
Downlink data topic	mqtt/face/ID（Where ID refers to the device ID/SN，e.g,:1306612）
Uplink data topic	mqtt/face/ID/Ack（Where ID refers to the device ID/SN，e.g,:1306612）

3.Getting HTTP Subscription Parameters Field Descriptions
Parameter information(Note:`Optional`is optional):

Key	Type	Values	Description
operator		GetHTTPconfig	Get the HTTP subscription parameters.
messageId	String		Message id, used by the platform to distinguish each message.
info	Object		Concrete content

4.Example of Getting HTTP Subscription Parameters

```
{
    "messageId": "xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx",
    "operator": "GetHTTPconfig",
    "info": {}
}
```

5.Device Response Field Descriptions

Key	Type	Values	Description
operator		GetHTTPconfig-Ack	Answer to get the HTTP subscription parameters.
messageId	String		Message id, used by the platform to distinguish each message.
code	String		Command execution error code 200-successes,see alsoerror cross-reference table
info	Object		Concrete content
facesluiceId	String		The clientID for connecting to the server,device ID/SN is used by default to ensure uniqueness
ProtocolType	String	0~1	HTTP protocol type 0:LAN 1:WAN
ServerAddr	String		Server address (up to 64 bytes in length) e.g. 192.168.2.11 or www.xxx.com
ServerPort	String		Server port，e.g. 80
Verify	String	0~4	Authentication results subscription 0: Non-subscription 1: Subscription authentication result upload (capture image) 2: Subscription authentication result upload (registration image) 3: Subscription authentication result upload (capture + registration image) 4: Subscription authentication result upload (no image)
VerifyURL	String		Authentication result subscription URL (length 128 bytes or less) Default :/Subscribe/Verify
Snap	String	0~1	Stranger capture subscription 0:Non-subscription 1:Subscribe to stranger capture reporting
SnapURL	String		Stranger capture subscription URL(length 128 bytes or less) Default:/Subscribe/Snap

Key	Type	Values	Description
QRCode	String	0~1	QR code subscription 0:Non-subscription 1:Subscribe to QR code reporting
QRCodeURL	String		QR code subscription URL(length 128 bytes or less) Default :/Subscribe/QRCode
IDCard	String	0~1	ID information subscription 0:Non-subscription 1:Subscription to ID card information reporting
IDCardURL	String		ID information subscriptionURL(length 128 bytes or less) Default:/Subscribe/IDCard
Alarm	String	0~1	Alert subscription 0:Non-subscription 1:Subscribe to alarm reporting
AlarmURL	String		Alert subscription URL(length 128 bytes or less) Default:/Subscribe/Alarm
WGCard	String	0~1	Wiegand card information subscription 0:Non-subscription 1:Subscribe to the wiegand information reporting
WGCardURL	String		Wiegand card information subscription URL(length 128 bytes or less) Default:/Subscribe/WGorRFCard
BeatInterval	String	30	heartbeat interval
BeatURL	String		Heartbeat URL(length 128 bytes or less) Default:/Subscribe/heartbeat
TimePush	String	0~1	Timed Push 0:Not enabled 1:Enabled
PushInterval	String		Timed push interval

Key	Type	Values	Description
Auth	String	0~1	Post Whether to authenticate when submitting reported information None : Authentication processing is not required Basic : post Basic Authentication
UserName	String (optional)		User name, maximum length 64 character length (including terminator)
PassWord	String (optional)		Password, maximum length 64 character length (including terminator)

ResumefromBreakpoint	String	0~1	Whether to enable `Continue Transmitting After Disconnection` (with Stranger Capture subscription or Authentication subscription mode enabled)
			0:Not enabled
			1:Enabled.
			Don't enable `Continue Transmitting After Disconnection` function, only ensure the push data, no need the server return.
			If the feature is enabled, the following json packets are required to be returned by the server for pushing the stranger capture and authentication information.
			{
			"code": 200,
			"desc": "OK",
			}
			If the All-in-One machine does not receive the correct packet from the server after 10 seconds, it continues to push this message.

Key	Type	Values	Description
RFBPTimeBegin	String (optional)	YYYY-MM-DD hh:mm:ss	The default start time of the `Continue Transmitting After Disconnection`, you can not fill in. Default is the time to set up the stranger capture subscription or authentication subscription and enbale the `Continue Transmitting After Disconnection`. If the stranger capture and authentication capture information before the breakpoint is pushed, this time will be changed to the time of completing the push of the `Continue Transmitting After Disconnection` e.g. 2018-03-12 09:10:00
result	String	"ok"/"fail"	Operating result
detail	String (optional)		Error message when Result is "Fail"

6.Device Response Example

```
{
    "messageId": "XXXXXXXXXXXXXXXXXXXXXXXXXXXXXX",
    "operator": "GetHTTPconfig-Ack",
    "code": "200",
    "info": {
        "facesluiceId": "1362003",
        "ProtocolType": "0",
        "ServerAddr": "172.168.2.90",
        "ServerPort": "0",
        "Verify": "1",
        "VerifyURL": "/Subscribe/Verify",
        "Snap": "1",
        "SnapURL": "/Subscribe/Snap",
        "QRCode": "0",
        "QRCodeURL": "/Subscribe/QRCode",
        "IDCard": "0",
        "IDCardURL": "/Subscribe/IDCard",
        "Alarm" : "0",
        "AlarmURL" : "/Subscribe/Alarm",
        "WGCard": "0",
        "WGCardURL": "/Subscribe/WGorRFCard",
        "BeatInterval": "30",
        "BeatURL": "/Subscribe/heartbeat",
        "TimedPush": "0",
        "PushInterval": "0",
```

```
        "Auth": "0",
        "UserName": "admin",
        "PassWord": "admin",
        "ResumefromBreakpoint": "0",
        "RFBPTimeBegin": "2021-7-29 14:54:35",
        "result": "ok"
    }
}
```

8.4.2 Setting HTTP Subscription Parameters

1.Description

Set the HTTP subscription parameters.

2.API Description

Items	Description
Interface Name	Set the HTTP subscription parameters.
Data flow (active command side)	Platform->Device
Downlink data topic	mqtt/face/ID（Where ID refers to the device ID/SN，e.g,:1306612）
Uplink data topic	mqtt/face/ID/Ack（Where ID refers to the device ID/SN，e.g,:1306612）

3.Setting HTTP Subscription Parameters Field Descriptions
Parameter information(Note:`Optional`is optional):

Key	Type	Values	Description
operator		UpHTTPconfig	Set the HTTP subscription parameters.
messageId	String		Message id, used by the platform to distinguish each message.
info	Object		Concrete content
ServerAddr	String (optional)		Server address (up to 64 bytes in length) e.g. 192.168.2.11 or www.xxx.com
ServerPort	int (optional)		Server port , e.g. 80
Verify	int (optional)	0~4	Authentication results subscription 0: Non-subscription 1: Subscription authentication result upload (capture image) 2: Subscription authentication result upload (registration image) 3: Subscription authentication result upload (capture + registration image) 4: Subscription authentication result upload (no image)
VerifyURL	String (optional)		Authentication result subscription URL (length 128 bytes or less) Default :/Subscribe/Verify
Snap	int (optional)	0~1	Stranger capture subscription 0:Non-subscription 1:Subscribe to stranger capture reportin
SnapURL	String (optional)		Stranger capture subscription URL(length 128 bytes or less) Default:/Subscribe/Snap
QRCode	int (optional)	0~1	QR code subscription 0:Non-subscription 1:Subscribe to QR code reporting
QRCodeURL	String (optional)		QR code subscription URL(length 128 bytes or less) Default :/Subscribe/QRCode

Key	Type	Values	Description
IDCard	int (optional)	0~1	ID information subscription 0:Non-subscription 1:Subscription to ID card information reporting
IDCardURL	String (optional)		ID information subscriptionURL(length 128 bytes or less) Default:/Subscribe/IDCard
Alarm	int (optional)	0~1	Alert subscription 0:Non-subscription 1:Subscribe to alarm reporting
AlarmURL	String (optional)		Alert subscription URL(length 128 bytes or less) Default:/Subscribe/Alarm
WGCard	int (optional)	0~1	Wiegand card information subscription 0:Non-subscription 1:Subscribe to the wiegand information reporting
WGCardURL	String (optional)		Wiegand card information subscription URL(length 128 bytes or less) Default:/Subscribe/WGorRFCard
BeatInterval	int (optional)	30	heartbeat interval
BeatURL	String (optional)		Heartbeat URL(length 128 bytes or less) Default:/Subscribe/heartbeat
TimedPush	int (optional)	0~1	Timed Push 0:Not enabled 1:Enabled
PushInterval	int (optional)		Timed push interval
Auth	int (optional)	0~1	Post Whether to authenticate when submitting reported information None : Authentication processing is not required Basic : post Basic Authentication

Key	Type	Values	Description
UserName	String (optional)		User name, maximum length 64 character length (including terminator) Required for Auth=1
PassWord	String (optional)		Password, maximum length 64 character length (including terminator) Required for Auth=1

ResumefromBreakpoint	int (optional)	0~1	Whether to enable `Continue Transmitting After Disconnection` (with Stranger Capture subscription or Authentication subscription mode enabled)
			0:Not enabled
			1:Enabled.
			Don't enable `Continue Transmitting After Disconnection` function, only ensure the push data, no need the server return.
			If the feature is enabled, the following json packets are required to be returned by the server for pushing the stranger capture and authentication information.
			{
			"code": 200,
			"desc": "OK",
			}
			If the All-in-One machine does not receive the correct packet from the server after 10 seconds, it continues to push this message.

RFBPTimeBegin	String (optional)	YYYY-MM-DD hh:mm:ss	The default start time of the `Continue Transmitting After Disconnection`, you can not fill in. Default is the time to set up the stranger capture subscription or authentication subscription and enbale the `Continue Transmitting After Disconnection`. If the stranger capture and authentication capture information before the breakpoint is pushed, this time will be changed to the time of completing the push of the `Continue Transmitting After Disconnection` e.g. 2018-03-12 09:10:00

4.Example of Getting HTTP Subscription Parameters

```
{
    "messageId": "XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX",
    "operator": "UpHTTPconfig",
    "info": {
        "ServerAddr": "172.168.2.90",
        "VerifyURL": "/Subscribe/Snap/Test"
    }
}
```

5.Device Response Field Descriptions

Key	Type	Values	Description
operator		UpHTTPconfig -Ack	Answer to set the HTTP subscription parameters.
messageId	String		Message id, used by the platform to distinguish each message.
code	String		Command execution error code 200-successes,see also 10.34 Setting HTTP Subscription Parameters Error Codes
info	Object		Concrete content
facesluiceId	String		The clientID for connecting to the server,device ID/SN is used by default to ensure uniqueness
result	String		Operating result.
detail	String (optional)		Error message when Result is "Fail"

6.Device Response Example

```
{
    "messageId": "XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX",
    "operator": "UpHTTPconfig-Ack",
    "code": "200",
    "info": {
        "facesluiceId": "1362003",
        "result": "ok"
    }
}
```

8.5 Temperature Parameters

The temperature parameters is mainly related to the temperature operation parameter of the face recognition all-in-one machine,**only the models that support temperature detection support these parameter.**

8.5.1 Getting Temperature Parameters

1.Description

Get the temperature parameters.

2.API Description

Items	Description
Interface Name	Get the temperature parameters.
Data flow (active command side)	Platform->Device
Downlink data topic	mqtt/face/ID（Where ID refers to the device ID/SN，e.g,:1306612）
Uplink data topic	mqtt/face/ID/Ack（Where ID refers to the device ID/SN，e.g,:1306612）

3.Getting Temperature Parameters Field Descriptions
Parameter information(Note:`Optional`is optional):

Key	Type	Values	Description
operator		GetTPTconfig	Get the temperature parameters.
messageId	String		Message id, used by the platform to distinguish each message.
info	Object		Concrete content

4.Example of Getting Temperature Parameters

```
{
    "messageId": "XXXXXXXXXXXXXXXXXXXXXXXXXXXXXX",
    "operator": "GetTPTconfig",
    "info": {}
}
```

5.Device Response Field Descriptions

Key	Type	Values	Description
operator		GetTPTconfig-Ack	Answer to get the temperature parameters.
messageId	String		Message id, used by the platform to distinguish each message.
code	String		Command execution error code 200-successes,see alsoerror cross-reference table
info	Object		Concrete content
facesluiceId	String		The clientID for connecting to the server,device ID/SN is used by default to ensure uniqueness
TemperatureMode	String	0~1	Temperature Display Mode 0:Celsius 1:Fahrenheit
ShowAbnormalTemp	String	0~1	Display abnormal temperature 0:Not display 1:Display
TemperatureCheck	double	默认：0.00	Temperature calibration
TempAutoLow	String	0~1(默认值１)	Automatically adjusts the low temperature threshold (Default on will reduce parameter setting problems due to ambient temperature changes) 0:No 1:Yes
TemperatureLow	String	34.00(Default for C models); 28.00(Default except for C models)	Low Temperature Threshold (Detected temperatures below the Low Temperature Threshold will be filtered, when TempAutoLow=1, the Low Temperature Threshold will be adaptive according to the outside ambient temperature)
TemperatureHigh	String	Default:37.30	High Temperature Threshold (detected temperatures above the high temperature threshold will be considered abnormal for body temperature)

Key	Type	Values	Description
TempFaceMinW	String	100~500	Minimum face recognition pixel width for temperature measurement The smaller the value, the farther the recognition distance for temperature measurement (default 350).
TempFaceMinH	String	50~1000	Minimum face recognition pixel height, limited to forehead. The smaller the value the smaller the recognition area, the larger the value the larger the recognition area (default 560).
bTempTimeHigh	String	0~1	Whether to limit the time period for high temperature detection 0: No time period limitation 1: Time period limitation
bTempokSwitch	String	0~1	Whether to turn on high-temperature alert 0: No 1: On
BDHC	double	40.00	BDHC：(Supported by special model)
BDLC	double	28.00	BDLC：(Supported by special model)
nTempWay	String	0~1	Temperature measurement mode:(Supported by special model) 0:Precision mode 1:Fast mode
nTempTimeBeg	String		High Temperature Detection Start Point Seconds counted from 0:00 of the day
nTempTimeEnd	String		High temperature detection end point Seconds counted from 0:00 of the day
result	String	"ok"/"fail"	Operating result
detail	String		Error message when Result is "Fail"

6.Device Response Example


```
{
    "messageId": "xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx",
    "operator": "GetTPTconfig-Ack",
    "code": "200",
    "info": {
```

```
        "facesluiceId": "2864575",
        "TemperatureMode": "0",
        "ShowAbnormalTemp": "0",
        "TemperatureCheck": "0.00",
        "TemperatureLow": "33.70",
        "TemperatureHigh": "37.30",
        "TempFaceMinW": "100",
        "TempFaceMinH": "700",
        "result": "ok"
    }
}
```

8.5.2 Setting Temperature Parameters

1.Description

Set the temperature parameters.

2.API Description

Items	Description
Interface Name	Set the temperature parameters.
Data flow (active command side)	Platform->Device
Downlink data topic	mqtt/face/ID（Where ID refers to the device ID/SN，e.g,:1306612）
Uplink data topic	mqtt/face/ID/Ack（Where ID refers to the device ID/SN，e.g,:1306612）

3.Setting Temperature Parameters Field Descriptions
Parameter information(Note:`Optional`is optional):

Key	Type	Values	Description
operator		UpTPTconfig	Set the temperature parameters.
messageId	String		Message id, used by the platform to distinguish each message.
info	Object		Concrete content
TemperatureMode	int (optional)	0~1	Temperature display mode 0:Celsius 1:Fahrenheit
ShowAbnormalTemp	int (optional)	0~1	Display abnormal temperature 0:Not display 1:Display
TemperatureCheck	double (optional)	Default：0.00	Temperature calibration.
TemperatureLow	double (optional)	34.00(Default for C models); 28.00(Default except for C models)	Low Temperature Threshold(Detected temperatures below the low temperature threshold will be filtered,when TempAutoLow=1 the low temperature threshold will be adaptive according to the external ambient temperature).
TemperatureHigh	double	默认:37.30	High Temperature Threshold(Detected temperatures above the high temperature threshold will be considered as body temperature anomalies).
TempAutoLow	int (optional)	0~1(default 1)	hether to limit the time period for high temperature detection 0: No time period limitation 1: Time period limitation
nTempWay	int (optional)	0~1	Temperature measurement mode:(Supported by special model) 0:Precision mode 1:Fast mode
BDHC	double (optional)	40.00	BDHC：(Supported by special model)
BDLC	double (optional)	28.00	BDLC：(Supported by special model)

Key	Type	Values	Description
TempFaceMinW	int (optional)	100~500	Minimum face recognition pixel width for temperature measurement The smaller the value, the farther the recognition distance for temperature measurement (default 350).
TempFaceMinH	int (optional)	50~1000	Minimum face recognition pixel height, limited to forehead. The smaller the value the smaller the recognition area, the larger the value the larger the recognition area (default 560).
bTempTimeHigh	int (optional)	0~1	Whether to limit the time period for high temperature detection 0: No time period limitation 1: Time period limitation
bTempokSwitch	int (optional)	0~1	Whether to turn on high-temperature alert 0: No 1: On
nTempTimeBeg	int (optional)		High Temperature Detection Start Point Seconds counted from 0:00 of the day
nTempTimeEnd	int (optional)		High temperature detection end point Seconds counted from 0:00 of the day

4.Example of Setting Temperature Parameters

```
{
    "messageId": "XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX",
    "operator": "UpTPTconfig",
    "info": {
        "TempAutoLow": 1
    }
}
```

5.Device Response Field Descriptions

Key	Type	Values	Description
operator		UpTPTconfig-Ack	Answer to Set the temperature parameters.
messageId	String		Message id, used by the platform to distinguish each message.
code	String		Command execution error code 200-successes,see also 10.35 Setting Temperature Parameters Error Codes
info	Object		Concrete content
facesluiceId	String		The clientID for connecting to the server,device ID/SN is used by default to ensure uniqueness.
result	String		Operating result.
detail	String (optional)		Error message when Result is "Fail"

6.Device Response Example

```
{
    "messageId": "XXXXXXXXXXXXXXXXXXXXXXXXXXXXXX",
    "operator": "UpTPTconfig-Ack",
    "code": "200",
    "info": {
        "facesluiceId": "2864575",
        "result": "ok"
    }
}
```

8.6 MQTT Parameters

MQTT parameters are mainly concerned with settings related to the connection between the panelizer and the MQTT server, etc.

8.6.1 Getting MQTT Parameters

1.Description

Get the MQTT parameters.

2.API Description

Items	Description
Interface Name	Get the MQTT parameters.
Data flow (active command side)	Platform->Device
Downlink data topic	mqtt/face/ID（Where ID refers to the device ID/SN，e.g,:1306612）
Uplink data topic	mqtt/face/ID/Ack（Where ID refers to the device ID/SN，e.g,:1306612）

3.Getting MQTT Parameters Field Descriptions

Parameter information(Note:`Optional`is optional):

Key	Type	Values	Description
operator	String	GetMQTTconfig	Get the MQTT parameters.
messageId	String		Message id, used by the platform to distinguish each message.
info	Object		Concrete content

4.Example of Getting MQTT Parameters

```
{
    "messageId": "xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx",
    "operator": "GetMQTTconfig",
    "info": {}
}
```

5.Device Response Field Descriptions

Key	Type	Values	Description
operator		GetMQTTconfig-Ack	Answer to get the MQTT parameters.
messageId	String		Message id, used by the platform to distinguish each message.
code	String		Command execution error code 200-successes,see alsoerror cross-reference table
info	Object		Concrete content
facesluiceId	String		The clientID for connecting to the server,device ID/SN is used by default to ensure uniqueness
StrangerUploadType	String	0~1	Whether to upload stranger capture information (default:0) **0:upload** **1:no upload**
RecordUploadType	String	0~2	Authentication Recognition Information Upload or Not (Default:1) 0:No upload 1:Recognition Record Upload with Snapshot 2:Recognition Record Upload without Snapshot Image
Direction	String	0~2	Entrance/Exit Direction 0:Unidirectional 1:Entrance 2:Exit
QRCode	String	0~1	Whether to upload QR code scanning result (default:0) 0:Not upload 1:Upload
IDCard	int	0~1	Whether to upload ID information (default:1) 0:Not uploaded 1:Uploaded

Key	Type	Values	Description
Card	int	0~1	Whether IC or RF card number is uploaded (default:1) 0:Not uploaded 1:Uploaded
Alarm	int	0~1	Whether alarms are uploaded (default:1) 0:not uploaded 1:uploaded
KeepAliveInterval	String	20~300	Application layer heartbeat interval
OnlineTopic	String		Topics for online and offline notification
HeartbeatTopic	String		Heartbeat topic
ResumefromBreakponit	String	0~1	MQTT stranger capture information and identification records whether to enable `Continue Transmitting After Disconnection` (under enable stranger capture upload or identification records upload mode) 0:Not enable 1:Enable Without `Continue Transmitting After Disconnection` function, it only guarantees to push the data, and it does not need to be returned by the platform (cloud server). Turn on the function, then push the stranger capture information and authentication information need to be returned by the server (see MQTT protocol 6.1 `Continue Transmitting After Disconnection` instructions) 10 seconds after the all-in-one machine does not receive the correct data packet from the server, then continue to push this information.

Key	Type	Values	Description
BeginTime	String (optional)	YYYY-MM-DDThh:mm:ss	The default start time of MQTT `Continue Transmitting After Disconnection`, you can leave it blank. Default is the time to set up the stranger capture upload or authentication record upload and enable the `Continue Transmitting After Disconnection` function. If the stranger capture and authentication record information before the breakpoint is pushed, this time will be changed to the time to finish the `Continue Transmitting After Disconnection` .e.g., 2020-05-25T09:10:00
result	String	"ok"/"fail"	Operating result
detail	String (optional)		Error message when Result is "Fail"

6.Device Response Example

```
{
    "messageId": "XXXXXXXXXXXXXXXXXXXXXXXXXXXXXX",
    "operator": "GetMQTTconfig-Ack",
    "code": "200",
    "info": {
        "facesluiceId": "2864575",
        "StrangerUploadType": "0",
        "RecordUploadType": "1",
        "Direction": "0",
        "QRCode": "0",
        "IDCard": 1,
        "Card": 1,
        "Alarm": 1,
        "KeepAliveInterval": "60",
        "OnlineTopic": "mqtt/face/basic",
        "HeartbeatTopic": "mqtt/face/heartbeat",
        "ResumefromBreakpoint": "0",
        "BeginTime": "2021-09-03T17:18:15",
        "result": "ok"
    }
}

```

8.6.2 Setting MQTT Parameters

1.Description

Set the MQTT parameters.

2.API Description

Items	Description
Interface Name	Set the MQTT parameters.
Data flow (active command side)	Platform->Device
Downlink data topic	mqtt/face/ID（Where ID refers to the device ID/SN，e.g,:1306612）
Uplink data topic	mqtt/face/ID/Ack（Where ID refers to the device ID/SN，e.g,:1306612）

3.Setting MQTT Parameters Field Descriptions
Parameter information(Note:`Optional`is optional):

Key	Type	Values	Description
operator		UpMQTTconfig	Set the MQTT parameters.
messageId	String		Message id, used by the platform to distinguish each message.
info	Object		Concrete content
StrangerUploadType	int (optional)	0~1	Whether to upload stranger capture information (default:0) **0:upload** **1:no upload**
RecordUploadType	int (optional)	0~2	Authentication Recognition Information Upload or Not (Default:1) 0:No upload 1:Recognition Record Upload with Snapshot 2:Recognition Record Upload without Snapshot Image
Direction	int (optional)	0~2	Entrance/Exit Direction 0:Unidirectional 1:Entrance 2:Exit
QRCode	int (optional)	0~1	Whether to upload QR code scanning result (default:0) 0:Not upload 1:Upload
IDCard	int (optional)	0~1	Whether to upload ID information (default:1) 0:Not uploaded 1:Uploaded
Card	int (optional)	0~1	Whether IC or RF card number is uploaded (default:1) 0:Not uploaded 1:Uploaded
Alarm	int (optional)	0~1	Whether alarms are uploaded (default:1) 0:not uploaded 1:uploaded
KeepAliveInterval	int (optional)	20~300	Application layer heartbeat interval

Key	Type	Values	Description
OnlineTopic	String (optional)		Topics for online and offline notification
HeartbeatTopic	String (optional)		Heartbeat topic
ResumefromBreakponit	int (optional)	0~1	MQTT stranger capture information and identification records whether to enable `Continue Transmitting After Disconnection` (under enable stranger capture upload or identification records upload mode) 0:Not enable 1:Enable Without `Continue Transmitting After Disconnection` function, it only guarantees to push the data, and it does not need to be returned by the platform (cloud server). Turn on the function, then push the stranger capture information and authentication information need to be returned by the server (see MQTT protocol 6.1 `Continue Transmitting After Disconnection` instructions) 10 seconds after the all-in-one machine does not receive the correct data packet from the server, then continue to push this information.
BeginTime	String (optional)	YYYY-MM-DDThh:mm:ss	The default start time of MQTT `Continue Transmitting After Disconnection`, you can leave it blank. Default is the time to set up the stranger capture upload or authentication record upload and enable the `Continue Transmitting After Disconnection` function. If the stranger capture and authentication record information before the breakpoint is pushed, this time will be changed to the time to finish the `Continue Transmitting After Disconnection` e.g. 2020-04-01T09:10:00

4.Example of Setting MQTT Parameters

```
{
    "messageId": "XXXXXXXXXXXXXXXXXXXXXXXXXXXXXX",
    "operator": "UpMQTTconfig",
    "info": {
        "KeepAliveInterval": 120
    }
}
```

5.Device Response Field Descriptions

Key	Type	Values	Description
operator		UpMQTTconfig-Ack	Answer to Set the MQTT parameters.
messageId	String		Message id, used by the platform to distinguish each message.
code	String		Command execution error code 200-successes,see also 10.36 Setting MQTT Parameters Error Codes
info	Object		Concrete content
facesluiceId	String		The clientID for connecting to the server,device ID/SN is used by default to ensure uniqueness
ReConnet	String (optional)	0~1	Does the device require reconnect service 0: No 1: Yes
Reboot	String (optional)	0~1	Does the device need to be rebooted 0: No 1: Yes
result	String		Operating result.
detail	String (optional)		Error message when result is "Fail"

6.Device Response Example

```
{
    "messageId": "XXXXXXXXXXXXXXXXXXXXXXXXXXXXXX",
    "operator": "UpMQTTconfig-Ack",
    "code": "200",
    "info": {
        "facesluiceId": "2864575",
        "ReConnect": "1",
        "Reboot": "0",
        "result": "ok"
    }
}
```

8.7 Time Parameters

Get or set the device's time parameters.

8.7.1 Getting the Time Parameters

1.Description

Get the current time of the device.

2.API Description

Items	Description
Interface Name	Get the current time of the device.
Data flow (active command side)	Platform->Device
Downlink data topic	mqtt/face/ID（Where ID refers to the device ID/SN，e.g,:1306612）
Uplink data topic	mqtt/face/ID/Ack（Where ID refers to the device ID/SN，e.g,:1306612）

3.Getting the Time Parameters Field Descriptions
Parameter information(Note:`Optional`is optional):

Key	Type	Values	Description
operator	String	GetSysTime	Get the current time of the device.
messageId	String		Message id, used by the platform to distinguish each message.
info	Object		Concrete content

4.Example of Getting the Time Parameters

```
{
    "messageId": "XXXXXXXXXXXXXXXXXXXXXXXXXXXXXX",
    "operator": "GetSysTime",
    "info": {}
}
```

5.Device Response Field Descriptions

Key	Type	Values	Description
operator		GetSysTime-Ack	Answer to get the current time of the device.
messageId	String		Message id, used by the platform to distinguish each message.
code	String		Command execution error code 200-successes,see alsoerror cross-reference table
info	Object		Concrete content
facesluiceId	String		The clientID for connecting to the server,device ID/SN is used by default to ensure uniqueness
SysTime	String	YYYY-MM-DDThh:mm:ss	Current time of the device e.g. 2020-04-01T09:10:00
result	String	"ok"/"fail"	Operating result
detail	String (optional)		Error message when result is "Fail"

6.Device Response Example

```
{
    "messageId": "XXXXXXXXXXXXXXXXXXXXXXXXXXXXXX",
    "operator": "GetSysTime-Ack",
    "code": "200",
    "info": {
        "facesluiceId": "2864575",
        "SysTime": "2021-10-10T22:29:28",
        "result": "ok"
    }
}
```

8.7.2 Setting the Time Parameters

1.Description

Set the current time of the device.

2.API Description

Items	Description
Interface Name	Set the current time of the device.
Data flow (active command side)	Platform->Device
Downlink data topic	mqtt/face/ID（Where ID refers to the device ID/SN，e.g,:1306612）
Uplink data topic	mqtt/face/ID/Ack（Where ID refers to the device ID/SN，e.g,:1306612）

3.Setting the Time Parameters Field Descriptions
Parameter information(Note:`Optional`is optional):

Key	Type	Values	Description
operator		SetSysTime	Set the current time of the device.
messageId	String		Message id, used by the platform to distinguish each message.
info	Object		Concrete content
facesluiceId	String		The clientID for connecting to the server,device ID/SN is used by default to ensure uniqueness
SysTime	String	YYYY-MM-DDThh-mm-ss	Set the current time of the device e.g. 2020-04-01T09:10:00

4.Example of Setting the Time Parameters

```
{
    "messageId": "xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx",
    "operator": "SetSysTime",
    "info": {
        "facesluiceId":"123456",
        "SysTime": "2021-10-11T22:29:28"
    }
}
```

5.Device Response Field Descriptions

Key	Type	Values	Description
operator		SetSysTime-Ack	Answer to set the current time of the device.
messageId	String		Message id, used by the platform to distinguish each message.
code	String		Command execution error code 200-successes,see also 10.37 Setting the Time Parameters Error Codes
info	Object		Concrete content
facesluiceId	String		The clientID for connecting to the server,device ID/SN is used by default to ensure uniqueness
result	String	"ok"/"fail"	Operating result.
detail	String (optional)		Error message when Result is "Fail"

6.Device Response Example

```
{
    "messageId": "XXXXXXXXXXXXXXXXXXXXXXXXXXXXXX",
    "operator": "SetSysTime-Ack",
    "code": "200",
    "info": {
        "facesluiceId": "2864575",
        "result": "ok"
    }
}
```

8.8 Face Recognition Parameters

The face recognition parameters are mainly related to the recognition parameters of the Face Recognition All-in-One Machine. The recognition distance (corresponding to the minimum face pixel) can be modified through the interface. The X and Y values of the face detection area represent the starting position of the detection area of the video screen, and the width and height of the face detection area represent the width and height of the video screen detected based on X and Y. The width and height of the face detection area should be matched with the width and height of the face detection area. The face detection width and height should be used in conjunction with the X and Y values of the face detection area. When X>0, the face detection width should be less than (**maximum** face detection width - X). **Generally only the recognition distance (corresponding to the smallest pixel of the face) needs to be changed, and other parameters do not need to be changed again.**

8.8.1 Getting Face Recognition Parameters

1.Description

Get the face recognition parameters.

2.API Description

Items	Description
Interface Name	Get the face recognition parameters.
Data flow (active command side)	Platform->Device
Downlink data topic	mqtt/face/ID（Where ID refers to the device ID/SN，e.g,:1306612）
Uplink data topic	mqtt/face/ID/Ack（Where ID refers to the device ID/SN，e.g,:1306612）

3.Getting Face Recognition Parameters Field Descriptions
Parameter information(Note:`Optional`is optional):

Key	Type	Values	Description
operator	String	GetFaceconfig	Get the face recognition parameters.
messageId	String		Message id, used by the platform to distinguish each message.
info	Object		Concrete content

4.Example of Getting Face Recognition Parameters

```
{
    "messageId": "XXXXXXXXXXXXXXXXXXXXXXXXXXXXXX",
    "operator": "GetFaceconfig",
    "info": {}
}
```

5.Device Response Field Descriptions

Key	Type	Values	Description
operator		GetFaceconfig-Ack	Answer to get the face recognition parameters.
messageId	String		Message id, used by the platform to distinguish each message.
code	String		Command execution error code 200-successes,see alsoerror cross-reference table
info	Object		Concrete content
facesluiceId	String		The clientID for connecting to the server,device ID/SN is used by default to ensure uniqueness
recDistance	String	30~200(unit:cm)	recognition distance(cm)
faceMinPixel	String	32~672	Minimum pixel of the face (deprecated, use recognition distance instead)
detectArea_x	String	0~960(default 0)	Face detection area X start point (even)
detectArea_y	String	0~1280(default 0)	Face detection area Y start point (even)
detectArea_w	String	0~960(default 960)	Face detection area width (even)
detectArea_h	String	0~1280(default 1280)	Face detection area height (even)
result	String	"ok"/"fail"	Operating result
detail	String (optional)		Error message when Result is "Fail"

6.Device Response Example

```
{
    "messageId": "XXXXXXXXXXXXXXXXXXXXXXXXXXXXXX",
    "operator": "GetFaceconfig-Ack",
    "code": "200",
    "info": {
        "facesluiceId": "1463596",
        "recDistance": "200",
        "faceMinPixel": "124",
        "detectArea_x": "0",
        "detectArea_y": "0",
        "detectArea_w": "960",
```

```
        "detectArea_h": "1280",
        "result": "ok"
    }
}
```

8.8.2 Setting Face Recognition Parameters

1.Description

Set the face recognition parameters.

2.API Description

Items	Description
Interface Name	Set the face recognition parameters.
Data flow (active command side)	Platform->Device
Downlink data topic	mqtt/face/ID（Where ID refers to the device ID/SN，e.g,:1306612）
Uplink data topic	mqtt/face/ID/Ack（Where ID refers to the device ID/SN，e.g,:1306612）

3.Setting Face Recognition Parameters Field Descriptions
Parameter information(Note:`Optional`is optional):

Key	Type	Values	Description
operator		UpFaceconfig	Set the face recognition parameters.
messageId	String		Message id, used by the platform to distinguish each message.
info	Object		Concrete content
facesluiceId	String		The clientID for connecting to the server,device ID/SN is used by default to ensure uniqueness
recDistance	int (optional)	30~200(unit:cm)	recognition distance(cm)
faceMinPixel	int (optional)	32~672	Minimum pixel of the face (deprecated, use recognition distance instead)
detectArea_x	int (optional)	0~960(default 0)	Face detection area X start point (even)
detectArea_y	int (optional)	0~1280(default 0)	Face detection area Y start point (even)
detectArea_w	int (optional)	0~960(default 960)	Face detection area width (even)
detectArea_h	int (optional)	0~1280(default 1280)	Face detection area height (even)

4.Example of Setting Face Recognition Parameters

```
{
    "messageId": "XXXXXXXXXXXXXXXXXXXXXXXXXXXXXX",
    "operator": "UpFaceconfig",
    "info": {
        "recDistance": 150
    }
}
```

5.Device Response Field Descriptions

Key	Type	Values	Description
operator		UpFaceconfig-A ck	Answer to set the face recognition parameters.
messageId	String		Message id, used by the platform to distinguish each message.
code	String		Command execution error code 200-successes,see also 10.38 Setting Face Recognition Parameters Error Codes
info	Object		Concrete content
facesluiceId	String		The clientID for connecting to the server,device ID/SN is used by default to ensure uniqueness
result	String	"ok"/"fail"	Operating result
detail	String (optional)		Error message when Result is "Fail"

6.Device Response Example

```
{
    "messageId": "XXXXXXXXXXXXXXXXXXXXXXXXXXXXXX",
    "operator": "UpFaceconfig-Ack",
    "code": "200",
    "info": {
        "facesluiceId": "2864575",
        "result": "ok"
    }
}
```

8.9 4G Card Information

This interface is used for 4G card information reporting, only supports models with 4G communication module.

8.9.1 Getting 4G Card Information

1.Description

Get the 4G card information of the device.

2.API Description

Items	Description
Interface Name	Get the 4G card information of the device.
Data flow (active command side)	Platform->Device
Downlink data topic	mqtt/face/ID（Where ID refers to the device ID/SN，e.g,:1306612）
Uplink data topic	mqtt/face/ID/Ack（Where ID refers to the device ID/SN，e.g,:1306612）

3.Getting 4G Card Information Field Descriptions
Parameter information(Note:`Optional`is optional):

Key	Type	Values	Description
operator	String	QuerySIMCardInfo	Get the 4G card information of the device.
messageId	String		Message id, used by the platform to distinguish each message.
info	Object		Concrete content

4.Example of Getting 4G Card Information

```
{
    "messageId": "xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx",
    "operator": "QuerySIMCardInfo",
    "info": {}
}
```

5.Device Response Field Descriptions

Key	Type	Values	Description
operator		QuerySIMCardInfo-Ack	Answer to get the 4G card information of the device.
messageId	String		Message id, used by the platform to distinguish each message.
code	String		Command execution error code 200-successes,see alsoerror cross-reference table
info	Object		Concrete content
facesluiceId	String		The clientID for connecting to the server,device ID/SN is used by default to ensure uniqueness
imei	String		International Mobile Equipment Identity .
iccid	String		SIM card number.
imsi	String		International Mobile Subscriber Identifier.
result	String	"ok"/"fail"	Operating result
detail	String (optional)		Error message when Result is "Fail"

6.Device Response Example

```
{
    "messageId": "xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx",
    "operator": " QuerySIMCardInfo-Ack",
    "code": "200",
    "info": {
        "facesluiceId": "1379743",
        "imei": "***********",
        "iccid": "********************",
        "imsi": "*************",
        "result": "ok"
    }
}
```

8.10 RTSP Parameters

RTSP parameters are mainly the RTSP operation parameters of the device, set the RTSP parameters successfully **the device will reboot**, currently this interface is only supported in some models.

8.10.1 Getting RTSP Parameters

1.Description

Get the RTSP parameters of the device.

2.API Description

Items	Description
Interface Name	Get the RTSP parameters of the device.
Data flow (active command side)	Platform->Device
Downlink data topic	mqtt/face/ID（Where ID refers to the device ID/SN，e.g,:1306612）
Uplink data topic	mqtt/face/ID/Ack（Where ID refers to the device ID/SN，e.g,:1306612）

3.Getting RTSP Parameters Field Descriptions
Parameter information(Note:`Optional`is optional):

Key	Type	Values	Description
operator	String	GetRTSPCfg	Get the RTSP parameters of the device.
messageId	String		Message id, used by the platform to distinguish each message.
info	Object		Concrete content

4.Example of Getting RTSP Parameters

```
{
    "messageId": "XXXXXXXXXXXXXXXXXXXXXXXXXXXXXX",
    "operator": "GetRTSPCfg",
    "info": {}
}
```

5.Device Response Field Descriptions

Key	Type	Values	Description
operator		GetRTSPCfg-Ack	Answer to get the RTSP parameters of the device.
messageId	String		Message id, used by the platform to distinguish each message.
code	String		Command execution error code 200-successes,see alsoerror cross-reference table
info	Object		Concrete content
facesluiceId	String		The clientID for connecting to the server,device ID/SN is used by default to ensure uniqueness
OpenVerify	String	0~1	Whether to enable RTSP service 0: Not enabled 1: Enabled
PackSize	String	1~1500 （default:1500）	Package size (unit:byte)
RTSPPort	String		RTSP port
result	String	"ok"/"fail"	Operating result
detail	String (optional)		Error message when Result is "Fail"

6.Device Response Example

```
{
    "messageId": "XXXXXXXXXXXXXXXXXXXXXXXXXXXXXX",
    "operator": "GetRTSPCfg-Ack",
    "code": "200",
    "info": {
        "facesluiceId": "1463596",
        "OpenVerify": "0",
        "PackSize": "1800",
        "RTSPPort": "0",
        "result": "ok"
    }
}
```

8.10.2 Setting RTSP Parameters

1.Description

Set the RTSP parameters of the device.

2.API Description

Items	Description
Interface Name	Set the RTSP parameters of the device.
Data flow (active command side)	Platform->Device
Downlink data topic	mqtt/face/ID（Where ID refers to the device ID/SN，e.g,:1306612）
Uplink data topic	mqtt/face/ID/Ack（Where ID refers to the device ID/SN，e.g,:1306612）

3.Setting RTSP Parameters Field Descriptions
Parameter information(Note:`Optional`is optional):

Key	Type	Values	Description
operator		SetRTSPCfg	Set the RTSP parameters of the device.
messageId	String		Message id, used by the platform to distinguish each message.
info	Object		Concrete content
facesluiceId	String		The clientID for connecting to the server,device ID/SN is used by default to ensure uniqueness
OpenVerify	int (optional)	0~1	Whether to enable RTSP service 0: Not enabled 1: Enabled
PackSize	int (optional)	1~1500 （default:1500）	Package size (unit:byte)
RTSPPort	int (optional)		RTSP port

4.Example of Setting RTSP Parameters

```
{
    "messageId": "XXXXXXXXXXXXXXXXXXXXXXXXXXXXXX",
    "operator": "SetRTSPCfg",
    "info": {
        "OpenVerify": 1
    }
}
```

5.Device Response Field Descriptions

Key	Type	Values	Description
operator		SetRTSPCfg-Ack	Answer to set the RTSP parameters of the device.
messageId	String		Message id, used by the platform to distinguish each message.
code	String		Command execution error code 200-successes,see also 10.39 Setting RTSP Parameters Error Codes
info	Object		Concrete content
facesluiceId	String		The clientID for connecting to the server,device ID/SN is used by default to ensure uniqueness
result	String		Operating result.
detail	String (optional)		Error message when Result is "Fail"

6.Device Response Example

```
{
    "messageId": "XXXXXXXXXXXXXXXXXXXXXXXXXXXXXX",
    "operator": "SetRTSPCfg-Ack",
    "code": "200",
    "info": {
        "facesluiceId": "2864575",
        "result": "ok"
    }
}
```

8.11 GPS Location Information

Only models with GPS modules support interfaces related to GPS location information.

8.11.1 Getting GPS Location Information

1.Description

Get the GPS parameters of the device.

2.API Description

Items	Description
Interface Name	Get the GPS parameters of the device.
Data flow (active command side)	Platform->Device
Downlink data topic	mqtt/face/ID（Where ID refers to the device ID/SN，e.g,:1306612）
Uplink data topic	mqtt/face/ID/Ack（Where ID refers to the device ID/SN，e.g,:1306612）

3.Getting GPS Location Information Field Descriptions
Parameter information(Note:`Optional`is optional):

Key	Type	Values	Description
operator	String	GetGpsInfo	Get the GPS parameters of the device.
messageId	String		Message id, used by the platform to distinguish each message.
info	Object		Concrete content

4.Example of Getting GPS Location Information

```
{
    "messageId": "XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX",
    "operator": "GetGpsInfo",
    "info": {}
}
```

5.Device Response Field Descriptions

Key	Type	Values	Description
operator		GetGpsInfo-Ack	Answer to set the GPS parameters of the device.
messageId	String		Message id, used by the platform to distinguish each message.
code	String		Command execution error code 200-successes,see also 10.40 Getting GPS Location Information Error Codes
info	Object		Concrete content
facesluiceId	String		The clientID for connecting to the server,device ID/SN is used by default to ensure uniqueness
GpsState	String		GPS module status.: "NotConfigGPSType": The current version of the device does not have a GPS module set "Initiating": GPS module is initializing. "OK": GPS module working properly "NotConncet": GPS module not connected "BadSignal": No signal from GPS module
GpsNum	String (optional)	0-N	Number of satellites captured
CoordinateSystemType	String (optional)	0/1	Coordinate system 0: WGS84 1: GCJ02
Longitude	String (optional)		Longitude
Latitude	String (optional)		Latitude
EorW	String (optional)		East/West
NorS	String (optional)		North/South
Altitude	String (optional)		Height above sea level
UtcDate	String (optional)	YYYY-MM-DD	UTC date via satellite

Key	Type	Values	Description
UtcTime	String (optional)	hh:mm:ss	UTC time via satellite
result	String	"ok"/"fail"	Operating result
detail	String (optional)		Error message when Result is "Fail"

6.Device Response Example

```
{
"messageId":"XXXXXXXXXXXXXXXXXXXXXXXXXXXXX",
"operator": "GetGpsInfo-Ack",
"code":"200",
"info": {
"facesluiceId":"1804269",
"GpsState":"OK",
"GpsNum":"22",
"CoordinateSystemType":"0",
"Longitude":"113.83173",
"Latitude":"22.60939",
"EorW":"E",
"NorS":"N",
"Altitude":"25.6",
"UtcDate":"2023-01-17",
"UtcTime":"03:03:16",
"result":"ok"}
}
```

8.11.2 Setting GPS Parameters

1.Description

Set the GPS parameters of the device.

2.API Description

Items	Description
Interface Name	Set the GPS parameters of the device.
Data flow (active command side)	Platform->Device
Downlink data topic	mqtt/face/ID（Where ID refers to the device ID/SN，e.g,:1306612）
Uplink data topic	mqtt/face/ID/Ack（Where ID refers to the device ID/SN，e.g,:1306612）

3.Setting GPS Location Information Field Descriptions
Parameter information(Note:`Optional`is optional):

Key	Type	Values	Description
operator		SetGpsInfo	Set the GPS parameters of the device.
messageId	String		Message id, used by the platform to distinguish each message.
info	Object		Concrete content
CoordinateSystemType	int (optional)		Type of coordinate system 0: WGS84 1: GCJ02
result	String	"ok"/"fail"	Operating result
detail	String (optional)		Error message when Result is "Fail"

4.Example of Getting GPS Location Information

```
{
    "messageId": "xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx",
    "operator": "SetGpsInfo",
    "info": {
        "CoordinateSystemType": 0
    }
}
```

5.Device Response Field Descriptions

Key	Type	Values	Description
operator		SetGpsInfo-Ac>k	Answer to set the GPS parameters of the device.
messageId	String		Message id, used by the platform to distinguish each message.
code	String		Command execution error code 200-successes,see also 10.41 Setting GPS Location Information Error Codes
info	Object		Concrete content
facesluiceId	String		The clientID for connecting to the server,device ID/SN is used by default to ensure uniqueness
result	String		Operating result.
detail	String (optional)		Error message when Result is "Fail"

6.Device Response Example

```
{
    "messageId": "XXXXXXXXXXXXXXXXXXXXXXXXXXXXXX",
    "operator": "SetGpsInfo-Ack",
    "code": "200",
    "info": {
        "facesluiceId": "2864575",
        "result": "ok"
    }
}
```

9.FAQs

9.1 MQTT connection issues

9.2 Issues of issuance of personnel

9.2.1 Problems with consecutive calls to single personnel list distribution interface

Some platforms have interfaces for Single `Personnel List Addition or Modification`, if you need to call it several times, you need to add a time interval (500ms~1000ms is recommended) in the call, or wait for the device to return the Ack and then call the interface for sending a single list of people. The device is a single-threaded message processing task, a large number of calls may lead to some information loss.

9.3 Issue of Continue Transmitting After Disconnection

9.4 Access Strategy Issues

9.5 Image Related Questions

9.5.1 Questions with Base64 data format

Do not add escape characters to the code, the correct data is shown below:

```
"*Data omitted here, not to exceed 1M.*/"
```

9.5.2 List Image Does not Meet the Requirements Resulting in Failure to Issue

Image requirements:
size in 200KB or so, the pixel is not larger than 1080 * 1920
face area pixels not less than 128 × 128
face size accounted for more than 1 / 3 of the whole image
Other precautions to see the "registration photo specification" document, you can contact sales or technical support to provide.

10.Error Cross-Reference Table

10.1 Single Personnel List Addition or Modification Error Codes

code	Type	String	Description
200	int		The operation was successful.
460	int	"Data out of range"	Single packet of data over 1M
461	int	"can't find customId"	Failed to get keyword “customId”.
462	int	"Parameter error"	Failed to get keyword “info”.
463	int	"pic base64 data decode err"	“pic” image base64 decoding failed
464	int	"Get pic Person Feature err, please change a pic"	Failed to extract facial features from images
465	int	"other"	Database-related operations fail
466	int		Reserve
467	int	"Person num is full"	List database is full
468	int	"get URI pic data len too short"	Get the URI image data less than 1000 bytes
469	int	"get URI pic data len too long"	Get the URI image data more than 1M(image pixel limit within 1080P)
470	int	"get URI server ip error"	Failed to parse the server address of the image
471	int	"get pic and get URI error"	The keyword “pic” was not received, nor was the keyword “picURI”.
472	int	"get pic and connect URI IP error"	Get the URI image timeout or download image failure (Note that the device's DNS is correct)
473	int	"RFIDCard already exist"	RFIDCard number already exists(For built-in card models)
474	int	"Get WGFacilityCode error"	Failed to get keyword "WGFacilityCode"
475	int	"Get cardNum2 error"	Failed to get keyword "cardNum2"
476	int	"cardNum2 alread exist"	cardNum2 Wiegand card number already exists.
477	int	"Face alread exist"	The face image already exists.

10.2 Batch List Additions or Modifications Error Codes

code is the success or failure of the execution of the entire command, and errcode is the error code of the bulk operation interface for a single operation.

code	Type	Detail	Description
410	int	"EditPersonsNew is busy"	The Bulk Add or Modify interface is busy and the last Bulk Add or Modify command has not yet been completed.
411	int	"EditPersonsNew is not ready"	The Bulk Add or Modify interface is busy and the last Bulk Add or Modify command has not yet been completed.
412	int	"can not find DataBegin"	Failed to get keyword "DataBegin"
413	int	"can not find DataEnd"	Failed to get keyword "DataEnd"
414	int	"Parameter error"	Failed to get keyword “info”.
415	int	"can not find PersonNum"	Failed to get keyword "PersonNum"
416	int	"PersonNum out of range"	The total number of lists exceeds the maximum value of 1000
417	int	"json of person's data is not equal PersonNum"	The number of json array names obtained is different from the value of PersonNum.
418	int	"can not find info"	Failed to get keyword “info”.
419	int	"MQTT change list is not authorized"	No permission to add or modify lists via MQTT
460	int	"Data out of range"	Single packet of data over 1M

errcode	Type	Description
461	int	customId already exists
462	int	Reserve
463	int	Reserve
464	int	Failed to parse the server address of the image
465	int	Get the URI image timeout or download image failure (Note that the device's DNS is correct)
466	int	Failed to get URI image data content
467	int	Image data is too large
468	int	Failed to extract facial features from images.
469	int	Failed to write image data to database
470	int	Database write to list data fails
471	int	Failed to search database writable locations
472	int	Failed to overwrite disk
473	int	Failed to modify database list
474	int	List database is full
475	int	Wiegand card number already exists.
476	int	Failed to get keyword "WGFacilityCode"
477	int	Failed to get keyword "cardNum2"
478	int	Excessive similarity between people's pictures and other people's pictures in the base library.
479	int	Failed to get keyword "WiegandType"
480	int	RFIDCard number already exists. (For built-in card models)
481	int	Failed to get keyword "picURI"
482	int	Failed to read pictures of existing people.

10.3 Batch Add or Modify Progress Query Error Codes

code	Type	Detail	Description
461	int	"Can't find info"	Failed to get keyword “info”.

10.4 Deletion of Single Personnel List Error Codes

code	Type	Detail	Description
461	int	"can't find customId"	Failed to get keyword “customId”.
462	int	"Parameter error"	Failed to get keyword “info”.
464	int	"can't find customId's person"	No matches were queried

10.5 Batch Deletion of Personnel Lists Error Codes

code is the success or failure of the execution of the entire command, and errcode is the error code of the bulk operation interface for a single operation.

code	Type	Detail	Description
412	int	"can not find DataBegin"	Failed to get keyword "DataBegin"
413	int	"can not find DataEnd"	Failed to get keyword "DataEnd"
414	int	"Parameter error"	Failed to get keyword “info”.
415	int	"can not find PersonNum"	Failed to get keyword "PersonNum"
416	int	"PersonNum out of range"	The total number of lists exceeds the maximum value of 200
417	int	"json of person's data is not equal PersonNum"	The number of json array names obtained is not equal to the value of "PersonNum".
418	int	"can not find customId"	Failed to get keyword “customId”.
419	int	"MQTT change list is not authorized"	No permission to delete list via MQTT

10.6 All Lists Deleted Error Codes

code	Type	Detail	Description
419	int	"MQTT change list is not authorized"	No permission to delete all list via MQTT
462	int	"Parameter error"	Failed to get keyword “info”.
463	int	"Parameter deleteall error"	Failed to get keyword "deleteall"

10.7 Query Single Personnel List Information Error Codes

code	Type	Detail	Description
462	int	"Parameter error"	Failed to get keyword “info”.
463	int	"Parameter customId error"	Failed to get keyword “customId”.
464	int	"can't find customId's person"	No matches were queried

10.8 Query Multiple Personnel List Information Error Codes

code	Type	Detail	Description
462	int	"Parameter error"	Failed to get keyword “info”.
465	int	"Unknow facesluiceId"	The "facesluiceId" does not match the device ID
466	int	"can not find person"	No matches were queried

10.9 Add/Modify Access Strategies Error Codes

code	Type	Detail	Description
411	int	"Unknow facesluiceId"	The "facesluiceId" does not match the device ID
412	int	"Parameter error"	Failed to get keyword “info”.
413	int	"can not find strategyID"	No matching access strategy IDs were queried
414	int	"strategyID can not be null or zero"	Access strategy ID is empty or 0
415	int	"can not find startDate"	Failed to get keyword "startDate"
416	int	"can not find endDate"	Failed to get keyword "endDate"
417	int	"can not find allowCnt"	Failed to get keyword "allowCnt"
418	int	"db update error"	Database update failure
419	int	"Time period setting greater than 6or3"	Setting the number of time slots greater than 6

10.10 Delete Access Strategies Error Codes

code	Type	Detail	Description
411	int	"Unknow facesluiceId"	The "facesluiceId" does not match the device ID
412	int	"Parameter error"	Failed to get keyword “info”.
413	int	"can not find strategyID"	No matching access strategy IDs were queried
414	int	"strategyID can not be null or zero"	Access strategy ID is empty or 0

10.11 Query All Access Strategies Error Codes

code	Type	Detail	Description
411	int	"Unknow facesluiceId"	The "facesluiceId" does not match the device ID
412	int	"Parameter error"	Failed to get keyword “info”.

10.12 Query All Associated Users by Access Strategy ID Error Codes

code	Type	Detail	Description
411	int	"Unknow facesluiceId"	The "facesluiceId" does not match the device ID
412	int	"Parameter error"	Failed to get keyword “info”.
413	int	"can not find strategyID"	No matching access strategy IDs were queried
414	int	"strategyID can not be null or zero"	Access strategy ID is empty or 0

10.13 Query Access Strategy Details Error Codes

code	Type	Detail	Description
461	int	"Unknow facesluiceId"	The "facesluiceId" does not match the device ID
462	int	"Parameter error"	Failed to get keyword “info”.
463	int	"can not find strategyID"	No matching access strategy IDs were queried

10.14 Personnel Binding Access Strategy Error Codes

code	Type	Detail	Description
411	int	"Unknow facesluiceId"	The "facesluiceId" does not match the device ID
412	int	"Parameter error"	Failed to get keyword “info”.
413	int	"can not find personsInfo"	Failed to get keyword "personsInfo"
414	int	"personsInfo Json num error"	The number in personsInfo is 0.
461	int	"can not find person szCustomID	The person corresponding to the customid was not found
462	int	"can not find person strategyID"	The strategyID array was not found in the parameter
463	int	"Json of strategyID num error"	Failed to parse strategyID array
464	int	"Json of strategyID no find"	The bound strategyID does not exist in the access strategy database
465	int	"Json of strategyID num more mix"	Bound access strategy groups exceed the limit of 64

10.15 Personnel Unbinding Access Strategy Error Codes

code	Type	Detail	Description
411	int	"Unknow facesluiceId"	The "facesluiceId" does not match the device ID
412	int	"Parameter error"	Failed to get keyword “info”.
413	int	"can not find personsInfo"	Failed to get keyword "personsInfo"
414	int	"personsInfo Json num error"	The number in personsInfo is 0.
461	int	"can not find person szCustomID	The person corresponding to the customid was not found
462	int	"can not find person strategyID"	The strategyID array was not found in the parameter
463	int	"Json of strategyID num error"	Failed to parse strategyID array
464	int	"Json of strategyID no find"	The unbound strategyID does not exist in the pass strategy database
465	int	"Json of strategyID num more mix"	Unbound access strategy groups exceed the limit of 64

10.16 Upload Ads Error Codes

code	Type	Detail	Description
461	int	"Parameter error"	Failed to get keyword “info”.
462	int	"Can't find adslot"	Failed to get keyword "adslot"
463	int	"adslot out of range"	The value of "adslot" is out of range.
464	int	"Key path value error"	Image download path error
465	int	"AD file download fail"	Failed to download ad content
466	int	"Unknow picture type"	Unknown image type
467	int	"unknow facesluiceId"	The "facesluiceId" does not match the device ID

10.17 Drop Ads Error Codes

code	Type	Detail	Description
461	int	"Parameter error"	Failed to get keyword “info”.
462	int	"Can't find adslot"	Failed to get keyword "adslot"
463	int	"adslot out of range"	The value of "adslot" is out of range.
467	int	"unknow facesluiceId"	The "facesluiceId" does not match the device ID

10.18 QR Code Display Error Codes

code	Type	Detail	Description
462	int	"Parameter error"	Failed to get keyword “info”.
463	int	"can't find QRCodeData Param or Out of limit size"	Failed to get keyword "QRCodeData "
464	int	"Unknow facesluiceId"	The "facesluiceId" does not match the device ID
465	int	"This Version can not support this interface"	This interface is not supported in the current version.

10.19 Manual Snapshot Error Codes

code	Type	Detail	Description
462	int	"Parameter error"	Failed to get keyword “info”.
463	int	"GetSceneSnap error"	Failed to get panorama snap
464	int	"Version not support GetSceneSnap"	This interface is not supported in the current version.
465	int	"Unknow facesluiceId"	The "facesluiceId" does not match the device ID

10.20 Image Similarity Comparison Error Codes

code	Type	Detail	Description
462	int	"Parameter error"	Failed to get keyword “info”.
463	int	"Unknow facesluiceId"	The "facesluiceId" does not match the device ID
464	int	"Unknow pic1info"	Failed to get keyword "pic1info"
465	int	"Unknow pic2info"	Failed to get keyword "pic2info"
466	int	"base64_decode pic1info error"	Failed to decode image data base64 in "pic1info".
467	int	"GetPersonFeature pic1info error"	Failed to extract facial features from "pic1info". (Please change the image, or check if the image has the correct base64 data)
468	int	"base64_decode pic2info error"	Failed to decode image data base64 in “pic2info”.
469	int	"GetPersonFeature pic2info error"	Failed to extract facial features from "pic2info". (Please change the image, or check if the image has the correct base64 data)

10.21 Search Local Face Database by Image Error Codes

code	Type	Detail	Description
460	int	"Data out of range"	Single packet of data over1M
461	int	"Unkonw facesluiceId"	The "facesluiceId" does not match the device ID
462	int	"Parameter error"	Failed to get keyword “info”.
463	int	"Unkonw MaxSimilarity"	Failed to get keyword "MaxSimilarity"
464	int	"Unkonw MaxNum"	Failed to get keyword "MaxNum"
465	int	"Unkonw picinfo"	Failed to get keyword "picinfo"
466	int	"No Meet Conditions Person"	No matching personnel
467	int	"GetPersonFeature pic1info error"	Failed to extract facial features from "picinfo". (Please change the image, or check if the image has the correct base64 data)
468	int	"base64 decode picinfo error"	Failed to decode image data base64 in “picinfo”.

10.22 Detecting Faces in Image Error Codes

code	Type	Detail	Description
462	int	"Parameter error"	Failed to get keyword “info”.
463	int	"Unknow facesluiceId"	The "facesluiceId" does not match the device ID
464	int	"Unkonw picinfo"	Failed to get keyword "picinfo"
465	int	"base64_decode picinfo error"	Failed to decode image data base64 in “picinfo”.

10.23 Getting Audio Files Error Codes

code	Type	Detail	Description
462	int	"Parameter error"	Failed to get keyword “info”.
463	int	"Can't find Audio"	Can't find the audio file to play

10.24 Playing Audio Files Error Codes

code	Type	Detail	Description
462	int	"Parameter error"	Failed to get keyword “info”.
463	int	"can't find voicetype param"	Failed to get keyword "voicetype "
464	int	"can't find VoiceFileName"	Audio file does not exist
465	int	"Remote audio control is not enabled"	Remote control playback sound not enabled

10.25 Continue Transmitting After Disconnection Error Codes

code	Type	Detail	Description
462	int	"Can't find info"	Failed to get keyword “info”.
463	int	"can't find PushAckType param"	Failed to get keyword "PushAckType"
464	int	"PushAckType Error"	PushAckType has the wrong value.
465	int	"SnapOrRecordID Param Error"	Failed to get keyword "SnapOrRecordID"
466	int	"ResumefromBreakpoint !=1 or Snap or Record Push mode error"	Continue Transmitting After Disconnection not enabled or "Identification Log" or "Stranger" reporting
468	int	"SetItemPushed error"	Marking the log as pushed fails
469	int	"can not find SnapOrRecordID"	No record of the relevant ID was found

10.26 Manual Push Control Logging Error Codes

code	Type	Detail	Description
462	int	"Parameter error"	Failed to get keyword “info”.
463	int	"Unknow facesluiceId"	The "facesluiceId" does not match the device ID
464	int	"Unknow TimeS"	Failed to get keyword "TimeS"
465	int	"Unknow TimeE"	Failed to get keyword "TimeE"
466	int	"Can't Find any record issue, please check TimeS and TimeE"	No matching control logs
467	int	"Unknow RecordUploadType"	This interface requires a subscription to “Identification Log”.

10.27 Manual Push Stranger Logging Error Codes

code	Type	Detail	Description
462	int	"Parameter error"	Failed to get keyword “info”.
463	int	"Unknow facesluiceId"	The "facesluiceId" does not match the device ID
464	int	"Unknow TimeS"	Failed to get keyword "TimeS"
465	int	"Unknow TimeE"	Failed to get keyword "TimeE"
466	int	"Can't Find any snap issue, please check TimeS and TimeE"	No matching stranger capture logs
467	int	"Unknow StrangerUploadType"	This interface requires a subscription to "Stranger".

10.28 Software Version Upgrade Error Codes

code	Type	Detail	Description
461	int	"Parameter error"	Failed to get keyword “info”.
462	int	"Can't find name"	Failed to get keyword "name"
463	int	"Upgrade file download fail"	Failed to download upgrade file
464	int	"Can't find path"	Failed to get keyword "path"
465	int	"opendir fail"	Failed to open directory
466	int	"move file error"	Copy file exception
467	int		Reserve
468	int	"unknow facesluiceId"	Failed to get keyword "facesluiceId"
469	int	"check upgrade fileInfo error"	Failed to get upgrade package information
470	int	"check Platform or Lcdtype error"	Failed to verify upgrade file information
471	int	"get FSVersion error"	Failed to get upgrade package information
472	int	"check ALG Version error"	Error in the version number of the calibration algorithm
473	int	"extract upgrade files error"	Failed to unzip upgrade zip file
474	int	"executable program not find"	Executable program not found in upgrade file
475	int	"Upgrade is not supported for this type's file"	Upgrade type not supported at this time
476	int	"Upgrade wav file failed,please check the fileName"	Failed to upgrade voice file, please check the file name.
477	int	"UpgradeType is inconsistent with the download file"	Inconsistency between upgrade type and file type.
478	int	"The number of customized voice files exceeds 41"	The number of customized voice files exceeds 41.
479	int	"The number of ShowInfo Pic files exceeds 10"	The number of remote door opening UI image files exceeds 10.
480	int	"Please chaeck this pic!"	Image file format error,please check the image file.
481	int	"CheckLicens fail"	Authorization Detection Failure
482	int	"This version CustomID does not match"	Client code mismatch.

code	Type	Detail	Description
483	int	"This version is old"	Algorithm version too low.

10.29 Remote Open Door Error Codes

code	Type	Detail	Description
461	int	"Can't get json key 'xxx'"	Parameter error, field not found'xxx'
462	int	"Key 'xxx' value[yyy] is wrong"	Parameter error, field value error

10.30 Device Keep Open Setting Error Codes

code	Type	Detail	Description
462	int	"Parameter Info error"	Parameter error, field not found 'info'
463	int	"Key['xxx'] value error"	Parameter error, field value error

10.31 Setting System Parameters Error Codes

code	Type	Detail	Description
461	int	"Set param fail"	Failed to set parameters.
462	int	"Parameter error"	Failed to get keyword “info”.
463	int	"Language is not supported in this version"	This language is not supported in this version.
464	int	"MQTT change LiveDetectType is not authorized"	No permission to modify the "LiveDetectType" via MQTT

10.32 Setting the Parameters of the Sound of the Prompt and the Interface Display Error Codes

code	Type	Detail	Description
462	int	"Parameter error"	Failed to get keyword “info”.

10.33 Setting Door Opening Conditions and Outputting Control Parameters Error Codes

code	Type	Detail	Description
462	int	"Parameter error"	Failed to get keyword “info”.
463	int	"VerifyMode error"	Wrong way to open the door
464	int	"keyPassword error"	Incorrectly set door opening password

10.34 Setting HTTP Subscription Parameters Error Codes

code	Type	Detail	Description
462	int	"Parameter error"	Failed to get keyword “info”.
463	int	"ServerAddr error"	HTTP server address parameter error
464	int	"ServerPort error"	HTTP server port parameter error

10.35 Setting Temperature Parameters Error Codes

code	Type	Detail	Description
462	int	"Can't find info"	Failed to get keyword “info”.

10.36 Setting MQTT Parameters Error Codes

code	Type	Detail	Description
462	int	"Parameter error"	Failed to get keyword “info”.
463	int	"ResumefromBreakpoint need Topic Snap or Verify"	To enable Continue Transmitting After Disconnection mode, you need to subscribe to at least one of “Stranger” and “Identification Log”.

10.37 Setting the Time Parameters Error Codes

code	Type	Detail	Description
462	int	"Parameter error"	Failed to get keyword “info”.
463	int	"Can't find SysTime"	Failed to get keyword "SysTime"
464	int	"Can't get json key"	Parameter error, required field not found
465	int	"Value is wrong"	Parameter error, field value error

10.38 Setting Face Recognition Parameters Error Codes

code	Type	Detail	Description
462	int	"Parameter error"	Failed to get keyword “info”.
463	int	"unknow facesluiceId"	The "facesluiceId" does not match the device ID

10.39 Setting RTSP Parameters Error Codes

code	Type	Detail	Description
462	int	"Parameter error"	Failed to get keyword “info”.
463	int	"unknow facesluiceId"	The "facesluiceId" does not match the device ID
464	int	"RTSPPort error"	RTSP port setting error
464	int	"Device's model can not support this interface"	This interface is not supported in the current version.

10.40 Getting GPS Location Information Error Codes

code	Type	Detail	Description
461	int	"Device not support this interface"	Device not support this interface

10.41 Setting GPS Location Information Error Codes

code	Type	Detail	Description
461	int	"Device not support this interface"	This interface is not supported in the current version.
462	int	"Can't get json key 'xxx'"	Parameter error, field not found'xxx'
463	int	"Key 'xxx' value[yyy] is wrong"	Parameter error, field value error
464	int	"Save 'xxx' Error"	Failed to save configuration xxx