Security PUSH Communication Protocol

PUSH SDK

Date: January 2024

PUSH Protocol Version: 3.1.2

Doc Version: 4.8

English

Thank you for choosing our product. Please read the instructions carefully before operation. Follow these instructions to ensure that the product is functioning properly. The images shown in this manual are for illustrative

For further details, please visit our Company's website www.zkteco.com.

Trademark

ZKTeco is a registered trademark of ZKTeco. Other trademarks involved in this manual are owned by their respective owners.

Disclaimer

This manual contains information on the operation and maintenance of the ZKTeco equipment. The copyright in all the documents, drawings, etc. in relation to the ZKTeco supplied equipment vests in and is the property of ZKTeco. The contents hereof should not be used or shared by the receiver with any third party without express written permission of ZKTeco.

The contents of this manual must be read as a whole before starting the operation and maintenance of the supplied equipment. If any of the content(s) of the manual seems unclear or incomplete, please contact ZKTeco before starting the operation and maintenance of the said equipment.

It is an essential pre-requisite for the satisfactory operation and maintenance that the operating and maintenance personnel are fully familiar with the design and that the said personnel have received thorough training in operating and maintaining the machine/unit/equipment. It is further essential for the safe operation of the machine/unit/equipment that personnel has read, understood and followed the safety instructions contained in the manual.

In case of any conflict between terms and conditions of this manual and the contract specifications, drawings, instruction sheets or any other contract-related documents, the contract conditions/documents shall prevail. The contract specific conditions/documents shall apply in priority.

ZKTeco offers no warranty, guarantee or representation regarding the completeness of any information contained in this manual or any of the amendments made thereto. ZKTeco does not extend the warranty of any kind, including, without limitation, any warranty of design, merchantability, or fitness for a particular purpose.

ZKTeco does not assume responsibility for any errors or omissions in the information or documents which are referenced by or linked to this manual. The entire risk as to the results and performance obtained from using the information is assumed by the user.

ZKTeco in no event shall be liable to the user or any third party for any incidental, consequential, indirect, special, or exemplary damages, including, without limitation, loss of business, loss of profits, business

interruption, loss of business information or any pecuniary loss, arising out of, in connection with, or relating to the use of the information contained in or referenced by this manual, even if ZKTeco has been advised of the possibility of such damages.

This manual and the information contained therein may include technical, other inaccuracies or typographical errors. ZKTeco periodically changes the information herein which will be incorporated into new additions/amendments to the manual. ZKTeco reserves the right to add, delete, amend or modify the information contained in the manual from time to time in the form of circulars, letters, notes, etc. for better operation and safety of the machine/unit/equipment. The said additions or amendments are meant for improvement /better operations of the machine/unit/equipment and such amendments shall not give any right to claim any compensation or damages under any circumstances.

ZKTeco shall in no way be responsible (i) in case the machine/unit/equipment malfunctions due to any non-compliance of the instructions contained in this manual (ii) in case of operation of the machine/unit/equipment beyond the rate limits (iii) in case of operation of the machine and equipment in conditions different from the prescribed conditions of the manual.

The product will be updated from time to time without prior notice. The latest operation procedures and relevant documents are available on http://www.zkteco.com

If there is any issue related to the product, please contact us.

ZKTeco Headquarters

Address: ZKTeco Industrial Park, No. 32, Industrial Road,
Tangxia Town, Dongguan, China.

Phone: +86 769 - 82109991

Fax: +86 755 - 89602394

For business related queries, please write to us at: sales@zkteco.com.

To know more about our global branches, visit www.zkteco.com.

About the Company

ZKTeco is one of the world's largest manufacturer of RFID and Biometric (Fingerprint, Facial, Finger-vein) readers. Product offerings include Access Control readers and panels, Near & Far-range Facial Recognition Cameras, Elevator/floor access controllers, Turnstiles, License Plate Recognition (LPR) gate controllers and Consumer products including battery-operated fingerprint and face-reader Door Locks. Our security solutions are multi-lingual and localized in over 18 different languages. At the ZKTeco state-of-the-art 700,000 square foot ISO9001-certified manufacturing facility, we control manufacturing, product design, component assembly, and logistics/shipping, all under one roof.

The founders of ZKTeco have been determined for independent research and development of biometric verification procedures and the productization of biometric verification SDK, which was initially widely applied in PC security and identity authentication fields. With the continuous enhancement of the development and plenty of market applications, the team has gradually constructed an identity authentication ecosystem and smart security ecosystem, which are based on biometric verification techniques. With years of experience in the industrialization of biometric verifications, ZKTeco was officially established in 2007 and now has been one of the globally leading enterprises in the biometric verification industry owning various patents and being selected as the National High-tech Enterprise for 6 consecutive years. Its products are protected by intellectual property rights.

About the Manual

This manual introduces the **Security PUSH Communication Protocol**.

All figures displayed are for illustration purposes only. Figures in this manual may not be exactly consistent with the actual products.

Document Conventions

Conventions used in this manual are listed below:

GUI Conventions

For Software	
Convention	**Description**
Bold font	Used to identify software interface names e.g., **OK**, **Confirm**, **Cancel**.
>	Multi-level menus are separated by these brackets. For example, File > Create > Folder.
For Device	
Convention	**Description**
< >	Button or key names for devices. For example, press <OK>.
[ ]	Window names, menu items, data table, and field names are inside square brackets. For example, pop up the [New User] window.
/	Multi-level menus are separated by forwarding slashes. For example, [File/Create/Folder].

Symbols

Convention	Description
	This represents a note that needs to pay more attention to.
	The general information which helps in performing the operations faster.
	The information which is significant.
	Care taken to avoid danger or mistakes.
	The statement or event that warns of something or that serves as a cautionary example.

Revision History

Date	Version	Description	Revised By	Remarks
2024/01/12	V4.8	1. Add event code 4009~4013, 6008~6010 in Appendix 19. 2. Add BIOPHOTO optional field in 9.6.2 ENROLL_BIO. 3. Add Wiegand2.0 communication protocol (chapter 4.4/7.2/9.1.1) 4. Add video intercom contact list protocol (chapter 4.4/9.5.1/9.1.1/9.1.2)	Cao Yanming	
2023/07/14	V4.6	1. Modify the remote registration protocol in 9.6 Remote Registration. 2. Add explanation in 9.1.1 Update-Unified Template. 3. Add event code 6006~6007 in Appendix 19. 4. Add position value 62 of AccSupportFunList in 4.4 Registration. 5. Add the distribution of user validity range (StartTime and EndTime) in 9.1.1 Update-User's Access Control Privilege.	Cao Yanming	
2023/01/07	V4.2	1. Add event code 4008 and 6005 in Appendix 19. 2. Add DevNSLevel parameter and position value 60-61 of AccSupportFunList in 4.4 Registration. 3. Add SvrNSLevel parameter in 4.5 Download Configuration Parameters. 4. Add StartTime and EndTime fields in 9.1.1 Update-Time Rule. 5. Add protocol 9.1.1 Update-NTP Server Information. 6. Add NtpFunOn parameter in Appendix 12. 7. Add new verification modes 25-29 in Appendix 3. 8. Modify the position of TIPS field in 10 Remote Identification.	Cao Yanming	

		9. Add the visible light palm bit in the MultiBioDataSupport and MultiBioPhotoSupport examples in 3.1 Specification of Hybrid Identification Protocol. 10. Add description of type=10 visible light palm in 7.11 Upload Integrated Template, 9.1.1 Update-Unified Template, 9.1.2 Delete-Unified Template, 9.1.3 Count-Unified Template Count and 9.1.4 Query- Unified Template. 11. Add biometric type10 visible light palm in Appendix 18.		
2022/05/20	V3.8	1. Modify Majorver as MajorVer, and modify Minorver as MinorVer in 9.1.1 Update-Unified Template. 2. Add event code 4002~4007, 5017~5022, 6001~6004 in Appendix 19. 3. Add TIPS field in Server Response in 10 Remote Identification. 4. Add error code -30 in Appendix 1. 5. Add VF_STYLE_ HEALTH_QR_CODE = 12 in NewVFStyles, and add 58-59 newly supported by AccSupportFunList parameter in 4.4 Registration. 6. Add VF_TYPE_ HEALTH_QR_CODE = 12 in Access control new verification parameters in 9.5.1 Set Options. 7. Modify as "Unified use of UTF-8 encoding" in 1.2 Encoding. 8. Modify the description of the alarm field in 7.3 Upload Real-time Status.	Cao Yanming	
2021/07/28	V2.9	1. Add event code 43, 80, 132-134, 150-155, 249-252 in Appendix 2. 2. Modify the description when AA=05, add the descriptions when AA=14/15/16/17/18 in 9.4 Control Device. 3. Add protocols of Elevator control privilege group/Expansion board property/Expansion board configuration/Time rule and passing mode in 9.1.1	Cao Yanming	

		Update. 4. Add 50-57 newly supported by AccSupportFunList parameter, modify the value range of IsSupportQRcode parameter, add the parameter descriptions of NewNormalEventTypes/NewErrorEventTypes/NewWarningEventTypes in 4.4 Registration. 5. Add upload and setting of elevator control parameters. 6. Add special instructions for hybrid identification protocol in 3.1 Specification of Hybrid Identification Protocol. 7. Modify the parameter description of event in 7.2 Upload Real-time Events. 8. Add Appendix 19-Description of Extended Real-time Events. 9. Add protocols of channel controller (chapter7.13/7.14/9.4.3). 10. Add Appendix 20-Description of Channel Controller Upload Parameters. 11. Modify the $(SP) between command conditions as $(HT) in 9.1.2 Delete-Unified Template. 12. Add the field description of CardNo in 9.1.1 Update-User Information.		
2021/01/06	V2.3	1. Add event code 70-79, 111-121, 219, 224, 243-248 in Appendix 2. 2. Add VMSUserName, VMSPasswd parameters to upload in 7.4 Registration. 3. Add new access control verification method rules. 4. Add temperature measurement protocol. 5. Add QR code encryption protocol. 6. Add remote registration protocol.	Cao Yanming	
2020/07/30	V1.8	1. Add subcontracting upgrade protocol switch parameter: SubcontractingUpgradeFunOn in 4.4 Registration.	Cao Yanming	

		2. Add subcontracting upgrade protocol in 9.4.1 Upgrade. 3. Add description of sitecode and linkid fields in 10.2 Upload Real-time Events.		
2020/04/14	V1.7	1. Add event code 69, 233, 234, 237, 238, 239, 240, 241, 242 in Appendix 2 Description of Real-time Events.	Cao Yanming	
2020/03/31	V1.6	1. Add position value 47, 48, 49 in 7.4 Registration. 2. Add event code 63, 64, 65, 66, 67, 68, 109, 110, 235, 236 in Appendix 2 Description of Real-time Events. 3. Add the 6th~8th digits of the alarm field in 7.3 Upload Real-time Status.	Cao Yanming	
2020/03/20	V1.5	1. Add hybrid identification protocol. 2. Add deliver unified template protocol. 3. Add delete unified template protocol. 4. Modify the get comparison photo count protocol. 5. Add get unified template count protocol. 6. Modify the query comparison photo protocol. 7. Add query unified template protocol. 8. Modify the deliver comparison photo protocol. 9. Add URL mode to deliver user photo protocol.	Cao Yanming	
2019/08/02	V1.4	1. Added an error logging protocol.	Li Xianping	
2019/05/10	V1.3	2. Added a protocol for pushing device parameters.	Yan Guangtian	
2019/02/19	V1.2	1. Added a protocol for acquiring and querying comparison photos. 2. Modified the document format and some errors.	Yan Guangtian	
2018/10/10	V1.1	1. Added two communication encryption protocols: a) exchange public key protocol. b) exchange factor protocol. 2. Described the version of the supported	Yan Guangtian	

		communication encryption protocol: a) Access control PUSH: 3.1.1 or later. 3. Described the supported communication encryption protocol in detail (see Appendix 16).		
2018/8/29	V1.0	1. Added protocols for visible light face recognition.	Yan Guangtian	
2018/4/16	First edition		Li Xianping	

Table of Contents

1. Overview

The PUSH protocol is a data protocol defined based on the Hyper Text Transmission Protocol (HTTP) established on TCP/IP connection. The Push protocol applied to the data interchange between a server and a ZKTeco attendance device or a ZKTeco access control device defines the transmission formats of data (including user information, biological recognition templates, and attendance records) and the command format for control devices. At present, the protocol supports ZKTeco's WDMS, ZKECO, ZKNET, ZKBioSecurity3.0, and other servers as well as third-party servers such as ESSL from India.

1.1 Features

- Active uploading of new data.
- Resuming transmission from breakpoint.
- The client initiates all functions such as uploading data or performing commands issued by the server.

1.2 Encoding

Most data transmitted via the protocol consists of ASCII characters, but fields like the user name need to be encoded. The following are the rules defined for data of this type:

- Unified use of UTF-8 encoding

Currently, the following data involves this encoding type

- User names in a user information table

1.3 Introduction to HTTP Protocol

The PUSH protocol is a data protocol defined based on the HTTP protocol, and the following explains a brief introduction to the HTTP protocol. Skip this part if you are already familiar with this concept.

The HTTP is a request/response protocol. The format of request sent by a client to a server is a request method. A URI, a protocol version number, and then a MIME-like message containing modifiers, client information and a possible message body. The format of a response sent by the server to the client is a status line followed by a MIME-like message containing server information, entity meta-information and possible entity-body content. The status line the protocol version number of the message and a success code or error code. The following is an example.

Client Request

GET	http://113.108.97.187:8081/iclock/accounts/login/?next=/iclock/data/iclock/HTTP/1.1
User-Agent	Fiddler
Host	113.108.97.187:8081

Server Response

```
HTTP/1.1 200 OK
Server: Apache-Coyote/1.1
Date: Fri, 10 Jul 2015 03: 53: 16 GMT
Content-Type: text/html; charset=utf-8
Transfer-Encoding: chunked
Connection: close
Content-Language: en
Expires: Fri, 10 Jul 2015 03: 53: 16 GMT
Vary: Cookie, Accept-Language
Last-Modified: Fri, 10 Jul 2015 03: 53: 16 GMT
ETag: "c487be9e924810a8c2e293dd7f5b0ab4"
Pragma: no-cache
Cache-Control: no-store
Set-Cookie: csrftoken=60fb55cedf203c197765688ca2d7bf9e; Max-Age=31449600;
Path=/
Set-Cookie: sessionid=06d37fdc8f36490c701af2253af79f4a; Path=/
0
```

Remarks

- HTTP communication usually occurs under a TCP/IP connection.
- The default port is TCP 80, and other ports also be used.
- However, the HTTP protocol also is implemented via other protocols.
- Only reliable transmission is likely to be expected from the HTTP.
  (**Note:** HTTP usually is established on a transport layer protocol).
- Therefore, the user can use any protocols providing the same guarantee.

2. Definition

In this documentation, the format of definition reference is **${ServerIP}**.

Parameters	Description
ServerIP	The IP address of the server
ServerPort	A port of the server
XXX	An unknown value
Value1\Value2\Value3……\Valuen	Value 1\Value 2\Value 3\……\Value n
Required	Mandatory
Optional	Selectable
SerialNumber	Serial number. It can have characters, numbers, or combination of characters and numbers
NUL	Null (\0)
SP	Space
LF	Line feeder or a line break (\n)
CR	Carriage return (\r)
HT	Horizontal tab (\t)
DataRecord	A data record
CmdRecord	A command record
CmdID	The ID of a command
CmdDesc	Command description
Reserved	A reserved field
BinaryData	A binary data flow
BinaryData	A Base64-based data flow
TableName	The name of the data table
Key	A key
Value	A value
FilePath	A file path
URL	A resource location
Cond	A condition

3. Functions

The following describes functions supported by the PUSH protocol for the client.

- Initialize Information Interaction
- Upload Commands
- Download Commands
- Server Commands
- Appendixes

3.1 Specification of Hybrid Identification Protocol

With more and more types of biometrics, the instructions issued by different types of biometrics are also different, making software docking protocols exceedingly difficult.

In order to simplify the development process, the specifications for biological template/ photo issue/ upload/ query/ delete are unified.

Hybrid identification protocol docking process:

1. The device pushes the following 5 parameters to the server through registration interface:
   - MultiBioDataSupport
   - MultiBioPhotoSupport
   - MultiBioVersion
   - MaxMultiBioDataCount
   - MaxMultiBioPhotoCount.

   See [Registration] interface description for details.
2. The server issues the following two parameters to the device through the [Download Configuration Parameters] interface:
   - MultiBioDataSupport
   - MultiBioPhotoSupport.
3. Both the device and the server will determine the finally supported hybrid identification template/ photo type based on the MultiBioDataSupport and MultiBioPhotoSupport parameters pushed by each other.

   For example:

   Device Side: MultiBioDataSupport = 0:1:0:0:0:0:0:0:0:1:0, MultiBioPhotoSupport = 0:0:0:0:0:0:0:0:0:1:0

Server Side: MultiBioDataSupport = 0:0:0:0:0:0:0:0:0:1:0, MultiBioPhotoSupport = 0:0:0:0:0:0:0:0:0:1:0

The device supports fingerprint templates, visible light face templates, and visible light face photos. The software supports face templates and visible light face photos. Because the software does not support fingerprint templates, finally after the device docking with the software, it only supports visible light face templates and visible light face photos.

Hybrid identification protocol unified upload/ issue bio-templates format:

After successfully connecting to the hybrid identification protocol, a unified template format can be used for the types supported by the device and the server.

Types supported	Unified Template
The server issues the templates to the device.	Issue Unified Templates
The server issues the photos to the device.	Issue Comparison Photos
The server queries the template data.	Query Unified Templates
The sever queries the quantity of templates.	Query the Quantity of Unified Templates
The device uploads the templates to the server.	Upload Unified Templates
The device uploads the comparison photos to the server.	Upload Comparison Photos

Hybrid identification protocol unified upload templates/ photos quantity interface:

1. For devices that support hybrid identification protocol, the maximum number of templates/ photos supported by the current device will be pushed to the server at the registration interface: **MaxMultiBioDataCount**, **MaxMultiBioPhotoCount**.
2. The server can get all the photos/templates saved by the current device by [Get comparison photo count] and [Get unified template count].

Hybrid identification protocol specification real-time upload of unified templates and photos:

1. The bio-templates/ comparison photos registered by the device will be uploaded to the server in real time.

   Note: Upload interface refer to [upload unified templates] and [upload comparison photos].

2. You can use **PostBackTmpFlag** to specify whether you want the device to return the unified templates when the software issues the comparison photos.

   Note: For specific interface, please refer to [Issue Comparison Photos].

Hybrid identification protocol provides optimization strategies:

For devices that support both templates and photos issuing, the server can determine the device template version number based on the **MultiBioVersion** parameter uploaded by the device. If the server has saved the template of the current version number, the template can be issued first instead of comparison photos.

Note: To issue the comparison photos, the device needs to extract photos into templates, which is less efficient than directly issuing templates.

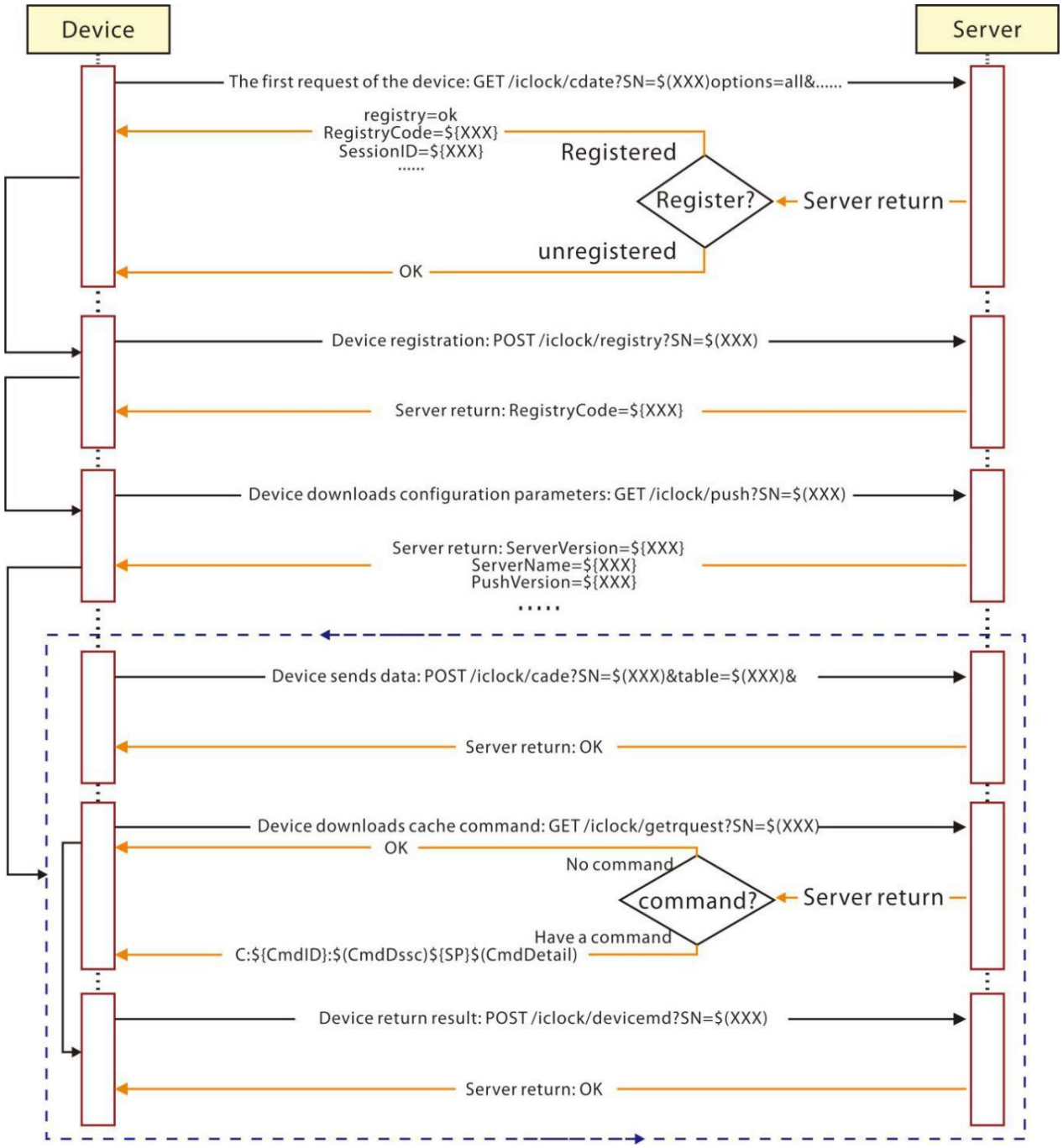
Special instructions for hybrid identification protocol:

If a template with the same algorithm version number is issued for the same type of biometric identification of the same person, both the issued template and the issued photo can make the device recognize normally. Please do not send out the template and photograph at the same time. This will only add to the device's workload, and it makes no sense.

4. Process

In PUSH mode, the client must first initiate a request to "Initialize Information Interaction" and can use other functions only after the request is successful, such as uploading data, obtaining server commands, uploading update information, and replying to the server commands.

These functions can be used in any sequence, specifically depending on the development of the client applications, as shown in the following figure:



4.1 Initialize Information Interaction

After the server receives the request sent by the client, two kinds of procedures are available depending on whether the device has been registered. If the device has been registered, the server returns the registration code and its configuration parameters and completes the connection. If the device is not registered, the client needs to initiate a registration request and send device parameters to the server. After the device is registered, the server returns the registration code. Then, the client sends a request to download the server configuration parameters.

The interaction is successful after the client obtains the configurations. The details are as follows:

Connection request of the client:

Application Scenario

If the device has not connected to the software, it needs to send a connection request to create a connection.

Client Request

HTTP	GET/iclock/cdata?SN=${SerialNumber}&pushver=${XXX} &options=all&${XXX}=${XXX} HTTP/1.1
Host	${ServerIP}:${ServerPort}

Annotation

HTTP Request Method	GET method
URI	/iclock/cdata
HTTP Protocol Version	1.1

Client Configuration Information

Parameter	Required/ Optional	Description
SN	${Required}	Client's serial number.
pushver	${Optional}	Latest Push protocol version of the device. For details, see Appendix 13.
options	${Required}	Obtaining server configuration parameters, and only the value "all" is available currently.
Host header	${Required}	-
Other headers	${Optional}	-
${XXX}		It is pushed only when this parameter has been set for the device, for example, DeviceType.

Note: If the device has been registered, the next step is to send a request to download the configuration parameters; otherwise, the next step is to send a registration request and then send a request to download the configuration parameters after the device is registered. Currently, the client needs to be registered for each connection.

Server Response

If the device is not registered, the format of the normal server response is as follows:

```
HTTP/1.1 200 OK
Server: Apache-Coyote/1.1
Content-Length: ${XXX}
Date: ${XXX}
OK
```

If the device has been registered, the format of the normal server response is as follows:

```
HTTP/1.1 200 OK
Server: ${XXX}
Content-Length: ${XXX}
Date: ${XXX}
…
registry=ok
```

```
RegistryCode=${XXX}
ServerVersion=${XXX}
ServerName=${XXX}
PushProtVer=${XXX}
ErrorDelay=${XXX}
RequestDelay=${XXX}
TransTimes=${XXX}
TransInterval=${XXX}
TransTables=${XXX}
Realtime=${XXX}
SessionID=${XXX}
TimeoutSec=${XXX}
```

If the device has been registered, the software will and must return the registry and RegistryCode; otherwise, the software will not return the registry and RegistryCode.

Server Configuration Information

Parameter	Description
registry	OK means that the device has been registered.
RegistryCode	A random number generated by the server, up to 32 bytes.
ServerVersion	The version of the server.
ServerName	The name of the server.
PushProtVer	The protocol version based on which the server is developed. For details, see Appendix 13.
ErrorDelay	Interval time for the client to reconnect to the server after networking connection failure. The recommended value is 30 to 300s.
RequestDelay	The interval (in seconds) at which the client sends the command acquiring request. If no value is configured at the client, the default value 30s are used.
TransTimes	Time at which the client checks for and transmits new data regularly (in a 24-hour format: hour: minute) and multiple times are separated by semicolons. Up to 10 times are supported. For example, TransTimes=00: 00;14: 00
TransInterval	The interval (in minutes) at which the system checks whether any

	new data needs to be transmitted. If no value is configured at the client, the default value 2 min is used.
TransTables	The new data that needs to be checked and uploaded. The default value is User Transaction, which means the user and access control records need to be automatically uploaded.
SessionID	The ID of the PUSH communication session. This field is used to calculate the new Token for all subsequent requests from the device.
TimeoutSec	The network timeout time. If no value is configured at the client, the default value 10s is used.
HTTP Status Line	Defined according to the standard HTTP protocol.
HTTP Response Header Field	-
Date header field	${Required} This header field is used for server time synchronization in GMT time format. For example, Date: Fri, 03 Jul 2015 06:53:01 GMT.
Content-Length header field	Based on the HTTP 1.1 protocol, this header field is usually used to specify the data length of a response entity. If the entity size is uncertain, header fields Transfer-Encoding: chunked, Content-Length and Transfer-Encoding are supported, all of which are standard definitions of the HTTP protocol.

Example:

Client Request

HTTP	GET/iclock/cdata?SN=3383154200002&pushver=3.0.1&options=all HTTP/1.1
Host	192.168.213.17:8088
User-Agent	iClock Proxy/1.09
Connection	starting
Accept	application/push
Accept-Charset	UTF-8
Accept-Language	VoicePrint-CN

Server Response

If the device is not registered, the format of the normal server response is as follows:

```
HTTP/1.1 200 OK
Server: Apache-Coyote/1.1
Content-Length:2
Date: Mon, 09 Jan 2017 02:20:19 GMT
OK
```

4.2 Exchange Public Keys (when communication encryption is supported)

Application scenario

The device pushes its public key to the server and receives the server's public key from the server.

Client Request

POST	POST /iclock/exchange?SN=$(SerialNumber)&type=publickey
Host	${ServerIP}:${ServerPort}
Content-Length	${XXX} …
Publickey	${XXX}

Annotation

HTTP Request Method	POST method
URI	/iclock/ exchange
HTTP Protocol Version	1.1

Client Configuration Information

Parameter	Required/ Optional	Description
Host header	${Required}	-
Other headers	${Optional}	-
Publickey		Call the device public key that is returned by the encryption library.

Server Response

```
HTTP/1.1 200 OK
Server: ${XXX}
Set-Cookie: ${XXX}; Path=/; HttpOnly
Content-Type: application/push;charset=UTF-8
Content-Length: ${XXX}
Date: ${XXX}
PublicKey= ${XXX}
```

Server Configuration Information

Parameter	Description
PublicKey	The server's public key returned by the server.

Example:

Client Request

HTTP	POST/iclock/exchange?SN=ODG7030067031100001&type=publickey HTTP/1.1
Cookie	token=e6bc9a9c2f9ce675e3548d5aeda2777e
Host	192.168.52.44:8088

User-Agent	iClock Proxy/1.09
Connection	starting
Accept	application/push
Accept-Charset	UTF-8
Accept-Languag e	zh-CN
Content-Type	application/push;charset=UTF-8
Content-Langua ge	zh-CN
Content-Length	3580

PublicKey=DMCtR13RwiGI4M9TRn/3xEmkddz2lqoZR7zUrOMhOc3FLvhLtpIWs3
REOSaKT4A9WO0ONt3V+mVb0W3Ka3NeCTWLjf9LplV1EyJIoZwXspGroPMTEWitLE
+LLsrO1r47OQRr62j5YSViUDKgzLVCvEek2iJ+3D181Z3qxV7a7WIoQ9DUGiPaU8
gml4cmiyqQimxIQ1wwMcMpcIFOIsSx7UjCG+D41dM/vh5UZxrQwn7IiMOmNdFXlB
+TOjaJ+4K/n3TDjubrbebqx6H2+nErH1mBuCCSNIKfwc5earkNfXPuqgBNGqCFJo
jgcQiOySquaq2DFXdUwYBLIURDnBLf+TtoSh4=

Server Response

HTTP/1.1 200 OK
Server: Apache-Coyote/1.1
Content-Length: 590
Date: Mon, 10 Sep 2018 10:10:55 GMT
PublicKey=DMCvFlzRwiGI4M9TRn/3xEmkddz2lqoZR7zUrOMhOc3FLvhLtpIYu3ZMNS
eKVLcZUv4iHNnxzl9B8SfuVSxXAwyijYAj6Wg4YyxTj4stv4K7q54sUCikb1CbQ/H0m9
QZGyhM1WjrHhppXT/CsOAquEy/2gxfrSt3nai38Hb/8QoTHvnXJR2EVpcY6u47jBeGiX
M3ZUQgCtcdB7JBXsOr71XWEsLX1fIC3GofGCy0g0bUkumWJfNKwBWfWzb95o6klDi8uP
/wU+DS1uLs1VcCN0WNtX+DCajyzcYvecR8cgbs0F1QfMmiRr/dYAOkwF/bSMyuLkd+o6
FmLBAh9keFtgkFa+PC5RlFGrmxpJx4lMoLfaNqUNwyAuRdKezvYBDUrRhGwgtwo/BRGU
oWCeOB4YP/gHHGro0M8f3/HlSqliuT55Xks/Btp1tpfO/OeJjELUA9Yu0o4TQlnM19Pu
OGsYhipM9NeGGexKjtotHotLT4Ccso04nAf7TltDavoPvVGJGiDbnN7l8wsUCsqcCRsi
KhpmON2waLjdFa8PNJ62N6Dl6QRPKn9XLnIDFdtKSq5Vgn

4.3 Exchange Factors (when communication encryption is supported)

Application scenario

The device pushes its factors to the server and receives the server's factors from the server.

Client Request

POST	POST/iclock/exchange?SN=$(SerialNumber)&type=factors
Host	${ServerIP}:${ServerPort}
Content-Length	${XXX} …
Factors	${XXX}

Annotation

HTTP Request Method	POST method
URI	/iclock/ exchange
HTTP Protocol Version	1.1

Client Configuration Information

Parameter	Required/ Optional	Description
Host header	${Required}	-
Other headers	${Optional}	-
Factors		Call the device factor that is returned by the encryption library.

Server Response

```
HTTP/1.1 200 OK
Server: ${XXX}
Set-Cookie: ${XXX}; Path=/; HttpOnly
Content-Type: application/push;charset=UTF-8
Content-Length: ${XXX}
Date: ${XXX}
Factors=${XXX}
```

Server Configuration Information

Parameter	Description
Factors	The server's factor key returned by the server.

Example:

Client Request

HTTP	POST/iclock/exchange?SN=ODG7030067031100001&type=factors HTTP/1.1
Cookie	token= e6bc9a9c2f9ce675e3548d5aeda2777e
Host	192.168.52.44:8088
User-Agent	iClock Proxy/1.09
Connection	starting
Accept	application/push
Accept-Charset	UTF-8
Accept-Language	zh-CN
Content-Type	application/push;charset=UTF-8
Content-Language	zh-CN

Content-Length	352
Factors=D/VtnItwfRAbVfyAJku6jKdobqCUIz2eTJ0yXIwI8ZMi5wSylQePPhVDGQvrppcssZ/zgX6qAWSKUXVlGRXNQ6kBk7+Kc6SaI090cvvMeLHCc1h69TIsbfkMtLGd1npscnDmmOoC19qqIbtTha9zNHmf38dHDkGZf3vLv/I8iwqV3RAX7MalBW4M+kYvbdYNgzR5kqcG+wTZOet5QAf/8YoRg7qZmnkAG6sAYALQotTfwQcjGUCtVoJONw8WshzkpdzgNsoo7UtYEmmhbEPHtqgaDN5OLexoE9u6Ip3gMd+V2hf70rs+mVUYR6frkEKe/6rA+oIdguhb2lp6HnwfNQ==	

Server Response

```
HTTP/1.1 200 OK
Server: Apache-Coyote/1.1
Content-Length:180
Date: Mon, 10 Sep 2018 10:10:55 GMT
Factors= XQ10e0WiFtslJW5ob221T/WCK42GXGP6mmiBB9yB93rD3CxlKZo3mavyqfTKFxtCn8AtkxL7MH4UeRvRnFTrv3Q4kKaYndiiphuvxOGQxrzcGGjH0sRzgPcTtAQu0U7A8vg2sMzZOxokqLuDVE5nlsx1/1V46wTK+oNU9q8fgKM=
```

4.4 Registration

Application scenario

After the client sends a connection request, if no registration code is returned (which means the device is not registered), the client needs to send a registration request to register the device.

Client Request

POST	POST /iclock/registry?SN=$(SerialNumber) HTTP/1.1
Host	${ServerIP}:${ServerPort}
Content-Length	${XXX} …
DeviceType=acc,~DeviceName=${XXX},FirmVer=${XXX},PushVersion=${XXX},MAC=$(XXX),CommType=$(XXX),MaxPackageSize=${XXX},LockCount=${XXX},ReaderCount=${XXX},AuxInCount=${XXX},AuxOutCount=${XXX},MachineType=${XXX},~IsOnlyRFMachine=${XXX},~MaxUserCount=${XXX},~M	

```
axAttLogCount=${XXX},~MaxUserFingerCount=${XXX},MThreshold=${XXX
},IPAddress=${XXX},NetMask=${XXX},GATEIPAddress=${XXX},~ZKFPVers
ion=${XXX},~REXInputFunOn=${XXX},~CardFormatFunOn=${XXX},~SupAut
hrizeFunOn=${XXX},~ReaderCFGFunOn=${XXX},~ReaderLinkageFunOn=${X
XX},~RelayStateFunOn=${XXX},~Ext485ReaderFunOn=${XXX},~TimeAPBFu
nOn=${XXX},~CtlAllRelayFunOn=${XXX},~LossCardFunOn=${XXX},Simple
EventType=${XXX},VerifyStyles=${XXX},EventTypes=${XXX},DisableUs
erFunOn=${XXX},DeleteAndFunOn=${XXX},LogIDFunOn=${XXX},DateFmtFu
nOn=${XXX},DelAllLossCardFunOn=${XXX},AutoClearDay=${XXX},FirstD
elayDay=${XXX},DelayDay=${XXX},StopAllVerify=${XXX},FvFunOn=${XX
X},FaceFunOn=${XXX},FingerFunOn=${XXX},CameraOpen=${XXX},AccSupp
ortFunList=${XXX},AutoServerFunOn=${XXX},DelayOpenDoorFunOn=${XX
X},UserOpenDoorDelayFunOn=${XXX},MultiCardInterTimeFunOn=${XXX},
OutRelaySetFunOn=${XXX},MachineTZFunOn=${XXX},DSTFunOn=${XXX},Ca
rdSiteCodeFunOn=${XXX},MulCardUserFunOn=${XXX},UserNameFunOn=${X
XX},StringPinFunOn=${XXX},MaxLockCount=${XXX},MaxZigbeeCount=${X
XX},MaxMCUCardBits=${XXX},SupportReaderType=${XXX},CmdFormat=${X
XX},authKey=${XXX},MultiStageControlFunOn=${XXX},MasterControlOn
=${XXX},SubControlOn=${XXX},BioPhotoFun=${XXX}BioDataFun=${XXX},
UserPicURLFunOn=${XXX},MultiBioDataSupport=${XXX},MultiBioPhotoS
upport=${XXX},MultiBioVersion=${XXX},MaxMultiBioDataCount=${XXX}
,MaxMultiBioPhotoCount=${XXX},SubcontractingUpgradeFunOn=${XXX},
IRTempDetectionFunOn=${XXX},MaskDetectionFunOn=${XXX},VMSUserNam
e=${XXX},VMSPasswd=${XXX},NewVFStyles=${XXX},IsSupportQRcode=${X
XX},QRCodeEnable=${XXX},QRCodeDecryptFunList=${XXX},DevNSLevel=$
{XXX}
```

Annotation

HTTP Request Method	POST method
URI	/iclock/ registry
HTTP Protocol Version	1.1

Client Configuration Information

Parameter	Description
DeviceType	**Attendance module:** att **Security module:** acc
DeviceName	Device name
FirmVer	Firmware version
PushVersion	Push SDK version
MAC	MAC address of device
CommType	Communication type **ethernet:** Cable network communication **usb-4G-modem:** 4G network communication **serial-wireless:** Wi-Fi (serial port and Sdio) communication
MaxPackageSize	Maximum size of communication package
LockCount	Number of doors
ReaderCount	Number of readers
AuxInCount	Number of auxiliary inputs
AuxOutCount	Number of auxiliary outputs
MachineType	Machine type Integrated machine 101
~IsOnlyRFMachine	Whether only RF card switching parameters are supported
~MaxUserCount	Maximum storage of user information (maximum storage of card information)
~MaxAttLogCount	Maximum storage of attendance record information
MaxUserFingerCount	Maximum storage of single user fingerprint
MThreshold	Fingerprint 1:N matching threshold
IPAddress	IP Address of cable network
NetMask	Subnet mask of cable network
GATEIPAddress	Gateway address of cable network
~ZKFPVersion	Version of Fingerprint algorithm

~REXInputFunOn	Switch parameter of exit button lock function, unused in the integrate machine.
~CardFormatFunOn	Whether Wiegand Reader can support the parameters of custom card format function.
~SupAuthrizeFunOn	Parameters of Super user
~Ext485ReaderFunOn	External RS485 Reader
~TimeAPBFunOn	Parameters of APB switch
~CtlAllRelayFunOn	Parameters of open all relay function
~LossCardFunOn	Parameters of losing card function
SimpleEventType	Event merging
VerifyStyles	Supported verification mode has 32 bytes, the first 16 bytes correspond to different combinations of authentication modes, details in Appendix 3 – Description of Verification Mode Code. The corresponding byte 1 means this verification method is supported, 0 means not supported, value 200 means access control event.
EventTypes	Supported event type has 32 bytes (0 to 31). Every byte has 8 bits (represent with 2 hexadecimal digits) total 256 bytes (0 to 255). For the corresponding function of every byte, please refer to Appendix 2 - Description of Real-time Event. The corresponding byte 1 means supported, and 0 means not supported. For example, consider BF0FE03D300001000000000070000000000000000000000000077002001000000. The 0 byte means BF, and the binary is 11111101 (lower byte is ahead), the 6th byte is 0, the other is 1, that means in the function that controlled by the first 8 bytes, only the linkage event of 6th byte cannot be supported, and the others can be supported.
NewNormalEventTypes	The format is a hexadecimal string with a maximum length of 256 bytes that specifies the offset value of the supported normal event type, with 4000 as the starting value.
NewErrorEventTypes	The format is a hexadecimal string with a maximum length of 256 bytes, indicating the offset value of the supported

	abnormal event type, and the starting value of the event type is 5000.
NewWarningEventTypes	The format is a hexadecimal string with a maximum length of 256 bytes, indicating the offset value of the supported warning event type, and the starting value of the event type is 6000. If the supported event type is 4000,4001,4003,4005,4009,4012, the corresponding is 0,1,3,5,9,12, and the corresponding binary bits are counted form right to left by byte: The offset binary corresponding bits of the event type: 7 6 5 4 3 2 1 0 15 14 13 12 11 10 9 8 The corresponding bit 1 means supported, 0 means not supported: 0 0 1 0 1 0 1 1 0 0 0 1 0 0 1 0 The corresponding string generated is 2B12.
DisableUserFunOn	Parameters of blocklist function.
DeleteAndFunOn	Parameters of delete and function.
LogIDFunOn	Parameters of id record function
DateFmtFunOn	Parameters of date format function (control the validity time of user table)
DelAllLossCardFunOn	Parameters of delete all blocklist function
AutoClearDay	The interval time of clear event log automatically.
FirstDelayDay	The first-time authorized door open time after unauthenticated user registration.
DelayDay	The first-time authorized door open delay time after unauthenticated user registration.
StopAllVerify	Stop all verification functions and forbid opening door with card function.
FvFunOn	Parameters of finger vein verification function
FaceFunOn	Parameters of face verification function
FingerFunOn	Parameters of fingerprint verification function
CameraOpen	Parameters of camera function
AutoServerFunOn	Parameters of background verification function

DelayOpenDoorFunOn	Door open delay function, not supported
UserOpenDoorDelayFunOn	Extend user door open time function, not supported
MultiCardInterTimeFunOn	Set the interval time of multiple card swiping function, not supported
OutRelaySetFunOn	Parameters of output configuration function, not supported
MachineTZFunOn:	Settings of time zone of the machine, not supported
DSTFunOn	Parameters of daylight-saving time function, not supported
CardSiteCodeFunOn	Sitecode function of card
MulCardUserFunOn	Multiple card user function (logic judgement usage of device business)
UserNameFunOn	User name function
StringPinFunOn	Supports employee number in string mode
MaxLockCount	Maximum number of doors
MaxZigbeeCount	Maximum number of Zigbee
MaxMCUCardBits	Maximum number of card binary byte
SupportReaderType	Supported reader type
CmdFormat	Command format, detail in Appendix 14.
MultiStageControlFunOn	Multiple stage control parameters, detail in Appendix 15.
MasterControlOn	Multiple stage control parameters, detail in Appendix 15.
SubControlOn	Multiple stage control parameters, detail in Appendix 15.
BioPhotoFun	Photos comprise parameter is supported
BioDataFun	Visible light face template parameter is supported.
UserPicURLFunOn	Supports issuing user photos by URL.
MultiBioDataSupport	Supports multi-modal bio-template parameters. The type is defined bit by bit. Different types are separated by colons, 0 means not supported, 1 means supported. The supported version number, such as: 0: 1: 1: 0: 0: 0: 0: 0: 0: 0, indicating support for fingerprint template and near-infrared face template.

MultiBioPhotoSupport	Supports multi-modal biometric photo parameters. The type is defined bit by bit. Different types are separated by colons, 0 means not supported, 1 means supported. The supported version number, such as: 0: 1: 1: 0: 0: 0: 0: 0: 0: 0, indicating support for fingerprint photo and near-infrared face photo.
MultiBioVersion	The multi-modal biometric data version. Different types are separated by colons, 0 means not supported, 1 means supported. The supported version number, such as: 0: 10: 0: 7: 0: 0: 0: 0: 0: 0, indicating support for fingerprint algorithm10.0 and near-infrared face algorithm7.0.
MaxMultiBioDataCount	Supports maximum number of multi-modal bio-templates. The type is defined bit by bit. Different types are separated by colons, 0 means not supported, 1 means supported. The supported maximum number of templates, such as: 0: 10000: 2000: 0: 0: 0: 0: 0: 0: 0, indicating support for the maximum number of fingerprint templates is 10000 and the maximum number of near-infrared face templates is 2000.
MaxMultiBioPhotoCount	Supports maximum number of multi-modal biometric photos. The type is defined bit by bit. Different types are separated by colons, 0 means not supported, 1 means supported. The supported maximum number of photos, such as: 0: 10000: 2000: 0: 0: 0: 0: 0: 0: 0, indicating support for the maximum number of fingerprint photos is 10000 and the maximum number of near-infrared face photos is 2000.
authKey	Communication key
SubcontractingUpgradeFunOn	Subcontract upgrade protocol function switch parameter
IRTempDetectionFunOn	The infrared temperature detection function is turned on
MaskDetectionFunOn	Mask detection function is on
VMSUserName	vms login user account
VMSPasswd	vms login password

NewVFStyles	The device supports new verification methods.
	A total of 16 digits, currently only supports up to 12 digits, issued as a string.
	`typedef enum _VERIFY_STYLE_E`
	`{`
	`VF_STYLE_NONE = 0, //0: Invalid bit`
	`VF_STYLE_FACE = 1, //1: Face`
	`VF_STYLE_PALM_PRINT = 2, //2: Palmprint`
	`VF_STYLE_PALM_VEIN = 3, //3: Palm vein`
	`VF_STYLE_FP = 4, //4: Fingerprint`
	`VF_STYLE_VEIN = 5, //5: Finger vein`
	`VF_STYLE_VP = 6, //6: Vocal print`
	`VF_STYLE_IRIS = 7, //7: Iris`
	`VF_STYLE_RETINA = 8, //8: Retina`
	`VF_STYLE_PW = 9, //9: Password`
	`VF_STYLE_PIN = 10, //10: User ID`
	`VF_STYLE_RF = 11, //11: Card`
	`VF_STYLE_HEALTH_QR_CODE = 12, //12: QR code/Health code`
	`}VERIFY_STYLE_E;`
	Such as:
	The device supports face, palm vein, fingerprint, password, UserID, card, then upload 0000111000011010.
	The device supports face, palm vein, password, card, then upload 0000101000001010.

IsSupportQRcode	Whether the device supports the QR code function, the value is as follows: **0:** QR code is not supported **1:** QR code is supported
QRCodeEnable	Whether to enable the QR code function (0: Off, 1: On)
QRCodeDecryptFunList	QR Code Decryption Function Parameter. This parameter is used to determine which decryption methods the device

<table>
<tr>
<td></td>
<td>specifically supports, and the function support parameters are obtained according to the bits (the parameter cannot be modified in software). If it is not transmitted, it is not supported by default.
The position value refers to the string position of the QRCodeDecryptFunList parameter value, starting from 0 and from left to right, as follows:

<table>
<tr><th>Position value</th><th>Meaning</th><th>Remarks</th></tr>
<tr><td>0</td><td>Device supports scheme 1 decryption</td><td>0 not supported;
1 supported</td></tr>
<tr><td>1</td><td>Device supports scheme 2 decryption</td><td>0 not supported;
1 supported</td></tr>
<tr><td>2</td><td>Device supports scheme 3 decryption</td><td>0 not supported;
1 supported</td></tr>
</table>
Remarks:
<table>
<tr><td>Scheme 1</td><td>Use the date of the day as the key and use the AES256 algorithm for encryption (the key is fixed).</td></tr>
<tr><td>Scheme 2</td><td>The system randomly generates a key and encrypts it with the AES256 algorithm (the key is not fixed).</td></tr>
<tr><td>Scheme 3</td><td>The system randomly generates a key and encrypts it with the RSA1024 algorithm (public and private keys are not fixed).</td></tr>
</table>
For example:
QRCodeDecryptFunList=101, which means that the device supports scheme one and three decryption methods.</td>
</tr>
<tr>
<td>DevNSLevel</td>
<td>National standard grade of device 0 to 4.
Note: The device and software can only be connected if they are of the same grade.</td>
</tr>
</table>

<table>
<tr><td>WGRuleVer</td><td>The Wiegand rule version currently has 1.0 and 2.0. When this value exists and is 2.0, it is Wiegand V2.0. Otherwise, it is considered as Wiegand V1.0.</td></tr>
<tr><td>AccSupportFunList</td><td>Obtains function support parameter according to the number of byte (this parameter cannot be modified), if no parameter is uploaded, default is 0.
Note: Position value means the value of the bit in the string of AccSupportFunList parameter value, start from 0 and from left to right.
<table>
<tr><th>Position Value</th><th>Description</th></tr>
<tr><td>0</td><td>Forbid reader function.
0 is not supported; 1 is supported.</td></tr>
<tr><td>1</td><td>RS485 reader encryption function.
0 is not supported, 1 is supported.</td></tr>
<tr><td>2</td><td>Lock door function.
0 is not supported, 1 is supported.</td></tr>
<tr><td>3</td><td>Emergency double open and emergency double close.
0 is not supported, 1 is supported.</td></tr>
<tr><td>4</td><td>Wiegand reader light corresponding to Red and green through door magnetism and relay status.
0 is not supported, 1 is supported.</td></tr>
<tr><td>5</td><td>Function of long name.
0 is not supported, 1 is supported.</td></tr>
<tr><td>6</td><td>Wiegand format with sitecode.
0 is not supported, 1 is supported.</td></tr>
<tr><td>7</td><td>Multiple cards for one user. This parameter is used by the software to judge the service logic. When the value is 1, the software delivers the device setting MulCardUserFunOn=1.
0 is not supported, 1 is supported.</td></tr>
</table>
</td></tr>
</table>

	8	Auxiliary input. The exit button is controlled by the time period. 0 is not supported, 1 is supported.
	9	Automatic deletion of temporary users every day. 0 is not supported, 1 is supported.
	10	Wg test function is supported. 0 is not supported; 1 is supported
	11	Account command is support. 0 is not supported, 1 is supported.
	12	Different user has different verification method in different time period. 0 is not supported, 1 is supported.
	13	Control RS485 reader LED. 0 is not supported, 1 is supported.
	14	Multiple Wiegand format is supported, this function has been kept. 0 is not supported, 1 is supported.
	15	DoorID only represent for one door in the authority table. 0 is not supported, 1 is supported.
	16	Independent table of user super authority. 0 is not supported, 1 is supported.
	17	Add door attribute function is supported. 0 is not supported, 1 is supported.
	18	Reader attribute function. 0 is not supported, 1 is supported.
	19	Attribute table function of auxiliary input. 0 is not supported, 1 is supported.
	20	Attribute table function of auxiliary output. 0 is not supported, 1 is supported.

	21	Parameter table function of the door. 0 is not supported, 1 is supported.
	22	Anti-passback table function. 0 is not supported, 1 is supported.
	23	Interlock table function. 0 is not supported, 1 is supported.
	24	Wireless-serial function. 0 is not supported, 1 is supported.
	25	4G-usb communication function. 0 is not supported, 1 is supported.
	26	Table support DevID field (if there is DevID field in the control protocol). 0 is not supported, 1 is supported.
	27	Secondary machine is supported or not. 0 is not supported, 1 is supported.
	28	Dual network card is supported or not. 0 is not supported, 1 is supported.
	29	Authorization table is supported or not. 0 is not supported, 1 is supported.
	30	Identity card registration or not. 0 is not supported, 1 is supported.
	31	Switch between primary and secondary machine of the device is supported or not. 0 is not supported, 1 is supported.
	32	Mixed reader and Wiegand/RS485 reader is supported or not in the device. 0 is not supported, 1 is supported.
	33	Wiegand data management, byte switching, bit reversal is supported in the device or not. 0 is not supported, 1 is supported.
	34	Desfire card control function is supported or

		not in the device. 0 is not supported, 1 is supported.
	35	Sending QR code command is supported or not. 0 is not supported, 1 is supported.
	36	Sending advertisement command zksq200 addition is supported or not. 0 is not supported, 1 is supported.
	37	Indoor station function zksq200 addition is supported or not. 0 is not supported, 1 is supported.
	38	Uploading comparison photo on the device is supported or not. 0 is not supported, 1 is supported.
	39	Uploading integrated light visible face template is supported or not in the device. 0 is not supported, 1 is supported.
	40	Whether the device can be remote registered. 0 is not supported, 1 is supported.
	41	Whether the device supports setting the verification result style. 0 is not supported, 1 is supported.
	42	Whether the device supports setting the authentication server. 0 is not supported, 1 is supported.
	43	Whether the device supports delivering the resource files, such as voice files, boot screen, welcome page, and screensaver page. 0 is not supported, 1 is supported.
	44	Whether the device can connect to an AI device (for non-inbio5-series devices that do not support the reader property table, such as inbioPro).

		0 is not supported, 1 is supported.
	45	Whether the device supports the expansion board DM10. 0 is not supported, 1 is supported.
	46	Whether the device supports the expansion board AUX485. 0 is not supported, 1 is supported.
	47	Whether the device supports the expansion board EX0808. 0 is not supported, 1 is supported.
	48	Whether the device supports the function of allowing super users to pass through when the door is locked. 0 is not supported, 1 is supported.
	49	Whether the device supports OSDP protocol expansion board. 0 is not supported, 1 is supported.
	50	Whether the device supports the user focusing function. 0 is not supported, 1 is supported.
	51	Whether the device supports the display of complete information. 0 is not supported, 1 is supported.
	52	Whether the device supports the elevator control expansion board. 0 is not supported, 1 is supported.
	53	Whether the device supports the direct access function of elevator control. 0 is not supported, 1 is supported.
	54	Whether the device supports the elevator control emergency interface recovery function.

		0 is not supported, 1 is supported.
	55	Whether the device supports the delivery of the privilege group of elevator control direct access and floor selection. 0 is not supported, 1 is supported.
	56	Whether the device supports the elevator control optimization operation (remote command merging, repeated event record merging). 0 is not supported, 1 is supported.
	57	Whether the device supports voice module. 0 is not supported, 1 is supported.
	58	Whether the device supports setting negative floors. 0 is not supported, 1 is supported.
	59	Whether the device supports the card number on an identity card. 0 is not supported, 1 is supported.
	60	Whether the time rule supports valid period configuration. 0 is not supported, 1 is supported.
	61	Whether the device supports NTP configuration. 0 is not supported, 1 is supported.
	62	Whether the device supports the function of delivering permission group validity periods. 0 is not supported, 1 is supported.
	63	Whether the device supports the distribution of video intercom contact list related functions. 0 is not supported, 1 is supported.

Server Response

```
HTTP/1.1 200 OK
Server: ${XXX}
Set-Cookie: ${XXX}; Path=/; HttpOnly
Content-Type: application/push;charset=UTF-8
Content-Length: ${XXX}
Date: ${XXX}
RegistryCode=${XXX}
```

Server Configuration Information

Parameters	Description
RegistryCode	A random number generated by the server, up to 32 bytes. If registration fails, 406 instead of the registration code is returned.

Example:

Client Request

HTTP	POST /iclock/registry?SN=3383154200002 HTTP/1.1
Host	192.168.213.17:8088
User-Agent	iClock Proxy/1.09
Connection	starting
Accept	application/push
Accept-Charset	UTF-8
Accept-Language	zh-CN
Content-Type	application/push;charset=UTF-8
Content-Language	zh-CN
Content-Length	855

```
DeviceType=acc,~DeviceName=F20/M,FirmVer=Ver
8.0.1.3-20151229,PushVersion=Ver
2.0.22-20161201,CommType=ethernet,MaxPackageSize=2048000,LockCou
nt=1,ReaderCount=2,AuxInCount=0,AuxOutCount=0,MachineType=101,~I
sOnlyRFMachine=0,~MaxUserCount=50,~MaxAttLogCount=10,~MaxUserFin
gerCount=10,MThreshold=60,IPAddress=192.168.213.221,NetMask=255.
255.255.0,GATEIPAddress=192.168.213.1,~ZKFPVersion=10,IclockSvrF
un=1,OverallAntiFunOn=0,~REXInputFunOn=0,~CardFormatFunOn=0,~Sup
AuthrizeFunOn=0,~ReaderCFGFunOn=0,~ReaderLinkageFunOn=0,~RelaySt
ateFunOn=1,~Ext485ReaderFunOn=0,~TimeAPBFunOn=0,~CtlAllRelayFunO
n=0,~LossCardFunOn=0,SimpleEventType=1,VerifyStyles=ff7f0000,Eve
ntTypes=BF0FE03D3000010000000000070000000000000000000000000770020
01000000,DisableUserFunOn=0,DeleteAndFunOn=0,LogIDFunOn=0,DateFm
tFunOn=0,DelAllLossCardFunOn=0,AutoClearDay=0,FirstDelayDay=0,De
layDay=0,StopAllVerify=0,FvFunOn=0,FaceFunOn=0,FingerFunOn=1,Cam
eraOpen=1,AccSupportFunList=01010100001110000001110101000110l0,A
utoServerFunOn=1,DelayOpenDoorFunOn=1,UserOpenDoorDelayFunOn=1,M
ultiCardInterTimeFunOn=1,OutRelaySetFunOn=1,MachineTZFunOn=1,DST
FunOn=1,CardSiteCodeFunOn=1,MulCardUserFunOn=1,UserNameFunOn=1,S
tringPinFunOn=1,MaxLockCount=4,MaxZigbeeCount=3,MaxMCUCardBits=3
7,SupportReaderType=1,authKey=dassas
```

Server Response

```
HTTP/1.1 200 OK
Server: Apache-Coyote/1.1
Set-Cookie: JSESSIONID=30BFB04B2C8AECC72C01C03BFD549D15; Path=/; Http
Only
Content-Type: text/plain;charset=ISO-8859-1
Content-Length: 23
Date: Mon, 09 Jan 2017 01:31:59 GMT
RegistryCode=Uy47fxftP3
```

4.5 Download Configuration Parameters

Application scenario

After the registration request is successful in the previous step, the device needs to actively obtain the server configuration parameters and then the whole initialization process is finished.

Client Request

POST	POST /iclock/push?SN=$(SerialNumber) HTTP/1.1
Cookie	token=${XXX}, timestamp=${XXX}
Host	${ServerIP}:${ServerPort} …

Server Response

```
HTTP/1.1 200 OK
Server: ${XXX}
Content-Type: application/push;charset=UTF-8
Content-Length: ${XXX}
Date: ${XXX}
…
ServerVersion=${XXX}
ServerName=${XXX}
ErrorDelay=${XXX}
RequestDelay=${XXX}
TransTimes=${XXX}
TransInterval=${XXX}
TransTables=${XXX
Realtime=${XXX}
SessionID=${XXX}
TimeoutSec=${XXX}
BioPhotoFun=${XXX}
BioDataFun=${XXX}
MultiBioDataSupport=${XXX}
MultiBioPhotoSupport=${XXX}
```

```
QRCodeDecryptKey=${XXX}
QRCodeDecryptType=${XXX}
```

Annotation

HTTP Request Method	GET method
URI	/iclock/push
HTTP Protocol Version	1.1

Client Configuration Information

Parameter	Description
ServerVersion	Server Version
ServerName	Server Name
ErrorDelay	Interval time between client reconnecting server after network failure (second), suggested value is 30 to 300 seconds, if there is no configuration, the client default value is 30 seconds
RequestDelay	Interval time between client obtain command requests (second), if no configuration the client default value is 30 seconds
TransTimes	Timely check time points for new data transfer, if no configuration the client default value is "12:30; 14:30"
TransInterval	Check whether there is a time interval for new data to be transmitted (minute), if no configuration the client default value is 2 minutes
TransTables	Need to check and upload new data, default is "User Transaction", need to automatically upload user and access control records
Realtime	Whether the client transmits new records in real time. 1 means that once there is new data then transferred it to the server, 0 means that it is transmitted according to the specified time of TransTimes and TransInterval, if there is no

	configuration, the default value is 1.
SessionID	PUSH Communication session ID, from now on all the requests of device will need this field to calculate new Token
TimeoutSec	Set the network timeout time, if the client is not configured, the default value is 10 seconds
BioPhotoFun	Specify whether to support the comparison photo parameter
BioDataFun	Specify whether to support the visible light face template parameter.
MultiBioDataSupport	Supports multi-modal bio-template parameters. The type is defined bit by bit. Different types are separated by colons, 0 means not supported, 1 means supported. The supported version number, such as: 0: 1: 1: 0: 0: 0: 0: 0: 0: 0, indicating support for fingerprint template and near-infrared face template
MultiBioPhotoSupport	Supports multi-modal biometric photo parameters. The type is defined bit by bit. Different types are separated by colons, 0 means not supported, 1 means supported. The supported version number, such as: 0: 1: 1: 0: 0: 0: 0: 0: 0: 0, indicating support for fingerprint photo and near-infrared face photo.
NewVFStyles	If the server supports the new verification method, it will reply to the value of this parameter uploaded by the device in the registration interface. If the server does not support the new verification method, the device can be compatible with the old verification method
IRTempUnitTrans	Specifies the unit of temperature upload (specifies the temperature unit of ConvTemperature in real-time events) 0: The temperature is uploaded in Celsius 1: The temperature is uploaded in Fahrenheit
QRCodeDecryptType	QR code decryption method, currently supports three methods, value 1, 2, 3. 1: Use scheme one, use today's date as the key and use AES256 algorithm to encrypt (the key is fixed) 2: Use scheme two, the system randomly generates a key and uses AES256 algorithm for encryption (the key is not

	fixed) 3: Use scheme three, the system randomly generates a key and uses RSA1024 algorithm for encryption (public and private keys are not fixed)
QRCodeDecryptKey	QR code key
SvrNSLevel	National standard level of software 0-4. **Note:** The device and software can only be connected if they are of the same level.
Note: The token calculation method: The client encrypts the values of RegistryCode, SerialNumber, and SessionID with the MD5 algorithm and then converts the calculated value to a hexadecimal string. The hexadecimal string is the token.	

Example:

Client Request

HTTP	POST /iclock/push?SN=3383154200002 HTTP/1.1
Cookie	token=9d41643bfba01fe8b49143c21defc20a
Host	192.168.213.17:8088
User-Agent	iClock Proxy/1.09
Connection	starting
Accept	application/push
Accept-Charset	UTF-8
Accept-Language	zh-CN
Content-Type	application/push;charset=UTF-8
Content-Language	zh-CN
Content-Length	0

Server Response

```
HTTP/1.1 200 OK
Server: Apache-Coyote/1.1
Content-Type: text/plain;charset=ISO-8859-1
Content-Length: 218
```

```
Date: Mon, 09 Jan 2017 01:31:59 GMT
ServerVersion=3.0.1
ServerName=ADMS
PushVersion=3.0.1
ErrorDelay=60
RequestDelay=2
TransTimes=00:0014:00
TransInterval=1
TransTables=User Transaction
Realtime=1
SessionID=30BFB04B2C8AECC72C01C03BFD549D15
TimeoutSec=10
```

4.6 Upload Device Parameters

Application scenario

To enable the device to actively push its parameters, set PushOptionsFlag to 1.

Client Request

POST	POST /iclock/cdata?SN=$(SerialNumber)&table=options HTTP/1.1
Cookie	token=${XXX}, timestamp=${XXX}
Host	${ServerIP}:${ServerPort} …
Parm1=${XXX}, Parm2=${XXX}, Parm3=${XXX}, Parm3=${XXX}, Parm4=${XXX}...	

Server Response

```
HTTP/1.1 200 OK
Server: ${XXX}
Content-Length: ${XXX}
Date: ${XXX}
OK
```

Annotation

HTTP Request Method	POST method
URI	/iclock/ cdata
HTTP Protocol Version	1.1

Client Configuration Information

Parameter	Description
Parm1	A required parameter, same to Parm2 and other parameters.
Note: The token calculation method: The client encrypts the values of RegistryCode, SerialNumber, and SessionID with the MD5 algorithm and then converts the calculated value to a hexadecimal string. The hexadecimal string is the token.	

Elevator Control Parameters

After the voice module function is turned on, the following parameters need to be uploaded:

Parameter	Description
VoiceModuleType	Voice module control mode, default: 0. 0: No verification required 1: Verification required
VoiceModuleCancelTime	The output time when the voice module cancels the button. Default: 2(s).
VoiceModuleCancelCount	The output times when the voice module cancels the button. Default: 1.

Example:

Client Request

POST	POST /iclock/cdata?SN=AJI6181160001&table=options HTTP/1.1
Cookie	token=50104b7ee71d8bb2ea10d8f1736b1bf8
Host	192.168.52.44:8088
User-Agent	iClock Proxy/1.09
Connection	starting
Accept	application/push
Accept-Charset	UTF-8
Accept-Language	zh-CN
Content-Type	application/push;charset=UTF-8
Content-Language	zh-CN
~DeviceName	ProBio,MAC=00:17:61:11:ad:4f,TransactionCount=23,~MaxAttLogCount=10,UserCount=7,~MaxUserCount=100,PhotoFunOn=1,~MaxUserPhotoCount=3000,FingerFunOn=1,FPVersion=10,~MaxFingerCount=40,FPCount=4,FaceFunOn=1,FaceVersion=7,~MaxFaceCount=2000,FaceCount=3,FvFunOn=0,FvVersion=3,~MaxFvCount=10,FvCount=0,PvFunOn=0,PvVersion=5,~MaxPvCount=,PvCount=0,Language=69,IPAddress=192.168.52.71,~Platform=ZMM220_TFT,~OEMVendor=ZKTeco Inc. ,FWVersion=Ver 8.0.4.1-20180102,PushVersion=Ver 2.0.34-20180822

Server Response

```
HTTP/1.1 200 OK
Server: Apache-Coyote/1.1
Content-Length: 2
Date: Wed, 22 Aug 2018 03:14:29 GMT
```

```
OK
```

5. Authorization

Authorization steps are as follows:

1. Only requests of the master control device are authorized. For the definition of the master control device, see Appendix 15.
2. The master control device actively pushes the sub-control authorization table to the software.

When the device is started, the authorization table is pushed to the software. Any change to the authorization table is pushed to the software.

Protocol:

```
POST/iclock/cdata?SN=123456789&type=registry&table=tabledata&tablename=DeviceAuthorize&count=1
```

Host: `113.108.97.187:8081`

```
DeviceAuthorizeSN=%?\tOnline=%?\tIsAuthorize=%?\tAuthorizeInfo=%?\tRegisterInfo=%?
```

Parameter Description

Parameter	Description
SN	The serial number of the secondary controller.
Online	Indicate whether the secondary controller is online.
IsAuthorize	Indicate whether it is authorized. 0 means Unauthorized, 1 means Authorization in Progress, and 2 means Authorized.
AuthorizeInfo	The simple device information pushed by the secondary controller when it is not authorized.
RegisterInfo	The device registration information pushed by the secondary controller after it is authorized.

3. The software authorizes the secondary controller based on the pushed authorization table.

The software delivers a command to authorize the secondary controller. Based on the authorization delivered by the software, BioIR9000 allows the specified secondary controller to push the registration information. The secondary controller pushes the registration information of the corresponding device to BioIR9000. BioIR9000 immediately pushes the

registration information of the secondary controller to the software.

Protocol: `C:295:DATA UPDATE DeviceAuthorize SN=%?\tIsAuthorize=1`

4. After receiving the registration information of the secondary controller, the software delivers the final authorization command.

After receiving the registration information of the secondary controller, the software delivers the final authorization command.Based on the authorization delivered by the software, BioIR9000 allows access from the specified secondary controller.Authorization is completed.

Protocol: `C:296:DATA UPDATE DeviceAuthorize SN=%?\tIsAuthorize=2`

5. The software deletes the authorized secondary controller.

Protocol: `C:295:DATA DELETE DeviceAuthorize SN=%?`

6. Heartbeat

Application scenario

It is used to retain the heartbeat with the server. During the upload of massive data, the ping command is used to retain the heartbeat. After the massive data is processed, the getrequest command is used to retain the heartbeat.

Client Request

POST	GET /iclock/ping?SN=$(SerialNumber) HTTP/1.1
Cookie	token=${XXX}
Host	${ServerIP}:${ServerPort}
Content-Length	${XXX} …

Server Response

```
TTP/1.1 200 OK
Server: Apache-Coyote/1.1
Content-Length: ${XXX}
Date: ${XXX}
OK
```

Annotation

HTTP Request Method	POST method
URI	/iclock/ ping
HTTP Protocol Version	1.1

Example:

Client Request

HTTP	GET /iclock/ping?SN=3383154200002 HTTP/1.1
Cookie	token=cb386eb5f8219329db63356fb262ddff
Host	192.168.213.17:8088
User-Agent	iClock Proxy/1.09
Connection	starting
Accept	application/push
Accept-Charset	UTF-8
Accept-Language	zh-CN
Content-Type	application/push;charset=UTF-8
Content-Language	zh-CN

Server Response

```
HTTP/1.1 200 OK
Server: Apache-Coyote/1.1
Content-Length: 2
Date: Tue, 10 Jan 2017 07:42:41 GMT
OK
```

7. Upload Commands

7.1 Upload Method

Real-time upload

Messages are uploaded immediately. The device supports this mode by default and you can control the upload mode on the server. For details, see the parameter Realtime in Initialize Information Interaction.

Upload at the interval

Messages are uploaded at an interval. You can set the specific interval on the server. For details, see the parameter TransInterval in Initialize Information Interaction.

Upload at a specific time

Messages are uploaded at a specific time point. You can set the specific time point on the server. For details, see the parameter TransTimes in Initialize Information Interaction.

7.2 Upload Real-time Events

Real-time events are events generated by the device in real time. Considering network delay, all the events generated in the 15s are generally called real-time events. For the code and description of real-time events, see Appendix 2.

Application scenario

When a real-time event occurs, the device needs to immediately upload the real-time event information to the software.

Client Request

HTTP	`POST /iclock/cdata?SN=$(SerialNumber)&table=rtlog HTTP/1.1`
Cookie	`token=${XXX}`
Host	`${ServerIP}:${ServerPort}`

Content-Length	${XXX} … ${DataRecord}

Server Response

```
HTTP/1.1 200 OK
Server: Apache-Coyote/1.1
Content-Length: ${XXX}
Date: ${XXX}
OK
```

Annotation

HTTP Request Method	POST method
URI	/iclock/ cdata
HTTP Protocol Version	1.1

Parameter Description

Parameter	Description
Table name	ACC_ATT_LOG
Request entity	time=${Time}${HT}pin=${Pin}${HT}cardno=${XXX}${HT}sitecode=${XXX}${HT}linkid=${XXX}${HT}eventaddr=${XXX}${HT}event=${XXX}${HT}inoutstatus=${XXX}${HT}verifytype=${XXX}${HT}index=(xxx){HT}maskflag=(xxx){HT}temperature=(xxx){HT}convtemperature=(xxx){HT}bitCount=$(xxx)
time	The time, in the format of XXXX-XX-XX XX:XX:XX.
pin	The user ID
cardno	The card number
sitecode	Site code

linkid	Linkage event ID. For example, the event that the person has not registered will not trigger the door to open. If the software is set to open the door when the person unregistered event triggers, then the door will be opened as long as the device has a linked event.
eventaddr	The event points. The default value is DoorId.
event	The event code. 0-255: The event code for the original PUSH protocol, see Appendix 2 for details. 4000-7000: The extended event code for the PUSH protocol, see Appendix 19 for details. 4000≤event<5000: Normal event 5000≤event<6000: Abnormal event 6000≤event<7000: Warning event
inoutstatus	The in/out status. 0 indicates In, and 1 indicates Out.
verifytype	If both the device and software support the upload of the new verification method rules: "And" verification method-The verification is successful, and the string of the current verification method is returned. If the verification fails, the reply will be compared with the current verification method, the string of the wrong verification method. For example, card + password + face, if the verification is successful, it will reply the verification method 0000101000000011. If the verification method is Card + fingerprint, but the face is verified, the reply is 0000000000000010. If the verification method is card + fingerprint, but the card and palmprint are verified, then reply 0000000000000100. "Or" verification method-Return the current verification type, face verification will reply 0000000000000010, palmprint verification will reply 0000000000000100, 0000000000010000 for fingerprint verification, 0000001000000000 for password verification, 0000010000000000 for User ID verification, and 0000100000000000 for card verification. If the device supports the new verification method but the software

	does not support it, verifytype uses the original verification method, see Appendix 3.
maskflag	Value 0 or 1, 1 means wearing a mask
temperature	The value is the temperature data with a decimal point, for example: 36.2.
convtemperature	The value is the temperature data with a decimal point. If the server does not send the IRTempUnitTrans parameter, then the unit of the temperature upload is subject to the IRTempUnit parameter.
bitcount	Wiegand bit number (only supported in WGRuleVer=V2.0 and above versions)
index	The access control record ID. Each access control record has a unique ID in the device.

Example:

Client Request

HTTP	POST /iclock/cdata?SN=3383154200002&table=rtlog HTTP/1.1
Cookie	token=cb386eb5f8219329db63356fb262ddff
Host	192.168.213.17:8088
User-Agent	iClock Proxy/1.09
Connection	starting
Accept	application/push
Accept-Charset	UTF-8
Accept-Language	zh-CN
Content-Type	application/push;charset=UTF-8
Content-Language	zh-CN
Content-Length	99
time=2017-01-10 11:49:32 pin=0 cardno=0 eventaddr=1 event=27 inoutstatus=1 verifytype=0 index=21	

Server Response

```
HTTP/1.1 200 OK
Server: Apache-Coyote/1.1
Content-Length: 2
Date: Tue, 10 Jan 2017 03:49:32 GMT
OK
```

7.3 Upload Real-time Status

The real-time status means the current status or logic status of the physical devices, such as the relay, door sensor, barrier, and alarm.

Application scenario

It is used to regularly upload the access control status to change to the access control status of the current device to the software.

Client Request

HTTP	POST /iclock/cdata?SN=$(SerialNumber)&table=rtstate HTTP/1.1
Cookie	token=${XXX}
Host	${ServerIP}:${ServerPort}
Content-Length	${XXX} ... ${DataRecord}

Server Response

```
HTTP/1.1 200 OK
Server: Apache-Coyote/1.1
Content-Length: ${XXX}
Date: ${XXX}
```

```
OK
```

Annotation

HTTP Request Method	POST method
URI	/iclock/ cdata
HTTP Protocol Version	1.1

Parameter Description

Parameter	Description
Table name	-rtstate
Request entity	${DataRecord}, the real-time event data in the format of: time=${XXX}${HT}sensor=AABB${HT}relay=CC${HT}alarm=DDEEFFGGHHIIJJKK
time	The current time, in the format of XXXX-XX-XX XX:XX:XX
sensor	Indicate the door sensor status. Each door occupies two binary bits, AA indicates doors 1-4 and BB indicates doors 5-8. Door 1 occupies the 1st and 2nd bits of the first byte, and so on. **0b00**: indicates that the current door sensor type is set to No Door Sensor. **0b01**: indicates that the current door is closed (with the door sensor). **0b10**: indicates that the door is open (without the door sensor)
relay	Indicate the relay status. Each door occupies two binary bits. 0b0 indicates that the relay is connected and 0b1 indicates that the relay is disconnected. The first bit indicates the relay status of door 1, and so on.
alarm	Indicate the alarm status. Eight doors occupy 16 characters (DDEEFFGGHHIIJJKK), each door using 2-characters of hexadecimal

	string. It can indicate up to 8 alarms. DD indicates the alarm status of the door 1, and so on. The alarms are defined as follows: **First bit**: Accidental door opening event. **Second bit**: Tamper alarm. **Third bit**: Duress password alarm. **Fourth bit**: Duress fingerprint alarm. **Fifth bit**: Door sensor timeout alarm. **Sixth bit**: Mains power failure alarm. **Seventh bit**: Battery power failure alarm. **Eighth bit**: Reader disassembly alarm.

Example:

Client Request

HTTP	`POST /iclock/cdata?SN=3383154200002&table=rtstate HTTP/1.1`
Cookie	`token=cb386eb5f8219329db63356fb262ddff`
Host	`192.168.213.17:8088`
User-Agent	`iClock Proxy/1.09`
Connection	`starting`
Accept	`application/push`
Accept-Charset	`UTF-8`
Accept-Language	`zh-CN`
Content-Type	`application/push;charset=UTF-8`
Content-Language	`zh-CN`
Content-Length	`69`

```
time=2017-01-10 15:42:40 sensor=00 relay=00 alarm=0200000000000000
```

Server Response

```
HTTP/1.1 200 OK
Server: Apache-Coyote/1.1
Content-Length: 2
Date: Tue, 10 Jan 2017 07:42:41 GMT
OK
```

7.4 Upload Returned Result of the Command

Application scenario

It is used to return the execution result to the server after the command delivered by the server is executed.

Client Request

HTTP	POST /iclock/devicecmd?SN=$(SerialNumber) HTTP/1.1
Cookie	token=${XXX}
Host	${ServerIP}:${ServerPort}
Content-Length	35 … ${DataRecord}

Server Response

```
HTTP/1.1 200 OK
Server: Apache-Coyote/1.1
Content-Type: text/plain;charset=ISO-8859-1
Content-Length: ${XXX}
Date: ${XXX}
OK
```

Annotation

HTTP Request Method	POST method
URI	/iclock/devicecmd
HTTP Protocol Version	1.1

Parameter Description

Parameter	Description
Request entity	${DataRecord}, the returned command result, in the format of: ID=${XXX}&Return=${XXX}&CMD=${XXX}&SN=${XXX}
ID	The command ID which is carried by the cache command delivered by the software.
Return	The returned command execution result. It is returned based on the multi-level control parameters. See Appendix 15. If the master controller receives the sub-controller command delivered by the server and returns -5000, the command has been received and waits to be executed. If the master controller command, non-multi-level device command, or sub-controller command delivered by the server is executed, the returned value is a general error code (see Appendix 1). A value equal to or greater than 0 means that the command is successfully executed. A value is smaller than 0 means that the command execution failed.
CMD	The command type.
SN	The SN of the sub-controller. SN is returned only when the command is executed by a sub-controller. For the definition of the sub-controller, see Appendix 15.
${LF} is used to connect multiple records.	

Example:

Client Request

HTTP	POST /iclock/devicecmd?SN=3383154200002 HTTP/1.1
Cookie	1637f0b091af92b0cc66b8eede0ae48e
Host	192.168.213.17:8088
User-Agent	iClock Proxy/1.09
Connection	starting
Accept	application/push
Accept-Charset	UTF-8
Accept-Language	zh-CN
Content-Type	application/push;charset=UTF-8
Content-Language	zh-CN
Content-Length	35
ID=1&Return=0&CMD=DATA UPDATE	

Server Response

```
HTTP/1.1 200 OK
Server: Apache-Coyote/1.1
Content-Type: text/plain;charset=ISO-8859-1
Content-Length: 2
Date: Wed, 11 Jan 2017 01:30:08 GMT
OK
```

7.5 Upload User Information

Application scenario

The device actively uploads the user information. Generally, after a user is registered, the device automatically uploads the user information to the server.

Client Request

HTTP	POST/iclock/cdata?SN=${SerialNumber}&table=tabledata&tablename=user&count=${XXX} HTTP/1.1
Cookie	token=${XXX}
Host	${ServerIP}:${ServerPort}
Content-Length	${XXX} ${DataRecord}

Annotation

HTTP Request Method	POST method
URI	/iclock/ cdata
HTTP Protocol Version	1.1

Parameter Description

Parameter	Description
Table type	tabledata
Table name	user
count	The number of users of which information is uploaded in the current message.
Request entity	${DataRecord}, the user data, in the format of: User uid=${XXX}$(HT)cardno=${XXX}$(HT)pin=${XXX}$(HT)password=${XXX}$(HT)group=${XXX}$(HT)starttime=${XXX}$(HT)endtime=${XXX}$(HT)name=${XXX}$(HT)privilege=${XXX}$(HT)disable=${XXX}$(HT)verify=${XXX}.
uid	The user's ID in the device.
cardno	The card number, which is an unsigned integer. Values delivered by

	the software can be uploaded in two formats: a. Hexadecimal data, in the format of [%02x%02x%02x%02x], which means the first to the fourth bytes from left to right. For example, if the card number is 123456789, then Card=[15CD5B07]. b. A string. If the card number is 123456789, then Card=123456789
pin	The user ID.
password	The user password, up to 6 bits.
group	The access control group of the user.
starttime	The start time of the user validity period: If the DateFmtFunOn value is 1, for example, starttime=583512660, starttime is calculated according to the algorithm in Appendix 5 and the specific time in the format of YYYYMMDDHHMMSS is obtained according to the method in Appendix 6. If the DateFmtFunOn value is 0, the format is YYYYMMDD.
endtime	The end time of the user validity period: If the DateFmtFunOn value is 1, for example, endtime=583512660, endtime is calculated according to the algorithm in Appendix 5 and the specific time in the format of YYYYMMDDHHMMSS is obtained according to the method in Appendix 6. If the DateFmtFunOn value is 0, the format is YYYYMMDD.
name	The user name. When the device language is Chinese, it is encoded with the GB2312 character set; otherwise, it is encoded with the UTF-8 character set.
privilege	The user privilege
disable	Whether it is a blacklist. 0: whitelist; 1: blacklist
verify	The supported verification mode
${LF} is used to connect multiple records.	

Server Response

```
HTTP/1.1 200 OK
Server: Apache-Coyote/1.1
```

```
Content-Length: ${XXX}
Date: ${XXX}
User: ${XXX} (The number of users received by the server this time).
```

Example:

Client Request

HTTP	POST/iclock/cdata?SN=3383154200002&table=tabledata&tablename=user&count=1 HTTP/1.1
Cookie	token=af65a75608cf5b80fbb3b48f0b4df95a
Host	192.168.213.17:8088
User-Agent	iClock Proxy/1.09
Connection	starting
Accept	application/push
Accept-Charset	UTF-8
Accept-Language	zh-CN
Content-Type	application/push;charset=UTF-8
Content-Language	zh-CN
Content-Length	104
user uid=5 cardno= pin=4 password= group=1 starttime=0 endtime=0 name= privilege=0 disable=0 verify=0	

Server Response

```
HTTP/1.1 200 OK
Server: Apache-Coyote/1.1
Content-Length: 6
Date: Thu, 12 Jan 2017 00:49:31 GMT
user=1
```

7.6 Upload the Identity Card Information

Application scenario

The device actively uploads the identity card information. Generally, after an identity card is registered, the device automatically uploads the identity card to the server.

Client Request

HTTP	POST/iclock/cdata?SN=${SerialNumber}&table=tabledata &tablename=identitycard&count=${XXX} HTTP/1.1
Cookie	token=${XXX}
Host	${ServerIP}:${ServerPort}
Content-Length	${XXX} ${DataRecord}

Annotation

HTTP Request Method	POST method
URI	/iclock/ cdata
HTTP Protocol Version	1.1

Parameter Description

Parameter	Description
Table type	tabledata
Table name	identitycard
count	The number of users of which information is uploaded in the current message.
Request entity	${DataRecord}, the user data, in the format of identitycard:

<table>
<tr><td></td><td>pin=${XXX}$(HT)SN_Num=${XXX}$(HT)ID_Num=${XXX}$(HT)DN_Num=${XXX}$(HT)Name=${XXX}$(HT)Gender=${XXX}$(HT)Nation=${XXX}$(HT)Birthday=${XXX}$(HT)Valid_Info=${XXX}$(HT)Address=${XXX}$(HT)Additional_Info=${XXX}$(HT)Issuer=${XXX}$(HT)Photo=${XXX}$(HT)PhotoJPG=${XXX}$(HT)FP_Template1=${XXX}$(HT)FP_Template2=${XXX}$(HT)Reserve=${XXX}$(HT)Notice=${XXX}</td></tr>
<tr><td>pin</td><td>The user's ID in the device.</td></tr>
<tr><td>SN_Num</td><td>The physical number of the identity card.</td></tr>
<tr><td>ID_Num</td><td>The number of the citizen identity card.</td></tr>
<tr><td>DN_Num</td><td>The SN of the resident identity card (the card management number).</td></tr>
<tr><td>Name</td><td>The name, UTF-8-encoded.</td></tr>
<tr><td>Gender</td><td>The gender code
1: Male
2: Female</td></tr>
<tr><td>Nation</td><td>The nationality code
<table>
<tr><th>Nation code</th><th>Nation</th></tr>
<tr><td>0</td><td>Decoding Error</td></tr>
<tr><td>1</td><td>Han</td></tr>
<tr><td>2</td><td>Mongol</td></tr>
<tr><td>3</td><td>Hui</td></tr>
<tr><td>4</td><td>Tibetan</td></tr>
<tr><td>5</td><td>Uyghur</td></tr>
<tr><td>6</td><td>Miao</td></tr>
<tr><td>7</td><td>Yi</td></tr>
<tr><td>8</td><td>Zhuang</td></tr>
<tr><td>9</td><td>Buyi</td></tr>
<tr><td>10</td><td>Korean</td></tr>
</table></td></tr>
</table>

	11	Manchu	
	12	Dong	
	13	Yao	
	14	Bai	
	15	Tujia	
	16	Hani	
	17	Kazak	
	18	Dai	
	19	Li	
	20	Lisu	
	21	Va	
	22	She	
	23	Gaoshan	
	24	Lahu	
	25	Shui	
	26	Dongxiang	
	27	Naxi	
	28	Jingpo	
	29	Kirgiz	
	30	Tu	
	31	Daur	
	32	Mulao	
	33	Qiang	
	34	Blang	
	35	Salar	
	36	Maonan	
	37	Kelao	
	38	Xibe	

	39 Achang 40 Pumi 41 Tajik 42 Nu 43 Ozbek 44 Russian 45 Ewenki 46 De'ang 47 Baoan 48 Yugu 49 Jing 50 Tatar 51 Drung 52 Oroqen 53 Hezhe 54 Monba 55 Lhoba 56 Jino 57 Encoding Error 97 Other 98 Foreign Origin
Birthday	The date of birth, in the format of yyyyMMdd.
Valid_Info	The validity period, start time and end time, in the format of yyyyMMddyyyyMMdd.
Address	The address, UTF-8-encoded.
Address_Info	The additional information read by the machine, UTF-8-encoded.
Issuer	The issuing authority, UTF-8-encoded.

Photo	The photo data stored in the identity card, which is encrypted and converted to the base64-based data before transmission.
PhotoJPG	The photo data stored in the identity card, which is decrypted and converted to the base64-based data before transmission.
FP_Template1	Features of the finger 1.
FP_Template2	Features of the finger 2.
Reserve	A reserved field.
Notice	The remarks, UTF-8-encoded.
${LF} is used to connect multiple records.	

Server Response

```
HTTP/1.1 200 OK
Server: Apache-Coyote/1.1
Content-Length: ${XXX}
Date: ${XXX}
identitycard: ${XXX}
```
(The number of identity cards received by the server this time).

Example:

Client Request

HTTP	POST/iclock/cdata?SN=3383154200002&table=tabledata&tablename=identitycard&count=1 HTTP/1.1
Cookie	token=af65a75608cf5b80fbb3b48f0b4df95a
Host	192.168.213.17:8088
User-Agent	iClock Proxy/1.09
Connection	starting
Accept	application/push
Accept-Charset	UTF-8
Accept-Language	zh-CN

Content-Type	application/push;charset=UTF-8
Content-Language	zh-CN
Content-Length	104

```
identitycard pin=1  SN_Num=26121292460088826975
IDNum=210102199212182xxxx  DNNum=  Name=Zhangsan Gender=1
Nation=1   Birthday=19911218  ValidInfo=2013091220230912
Address=xxxxx, Tangxia Town, Dongguan, Guangdong  AdditionalInfo=
Issuer=xxx Public Security Bureau  Photo=V0xmAH4AMgAA/apmyEd8
PhotoJPG=Y0tKqWNPzyEd8Pn0EhRsgAAjeWPxiUzLaPU1w=
FPTemplate1=QwGIEgELUQAAAAAAAAAAAAAACgBmnIcAf///////0wZqfwuJK78
PzJg/lw
FPTemplate2=QwGIEgEQUAAAAAAAAAAAAAAADIBmkbyAP///////3UYCPxXQs38
qEVW/rpLovwyUBv8WV8 Reserve=   Notice=
```

Server Response

```
HTTP/1.1 200 OK
Server: Apache-Coyote/1.1
Content-Length: 6
Date: Thu, 12 Jan 2017 00:49:31 GMT
identitycard=1
```

7.7 Upload Fingerprint Template

Application scenario

The device actively uploads the fingerprint template. Generally, after a user fingerprint is registered, the device automatically uploads the fingerprint to the server.

Client Request

HTTP	POST/iclock/cdata?SN=${SerialNumber}&table=tabledata &tablename=templatev10&count=${XXX} HTTP/1.1
Cookie	token=${XXX}
Host	${ServerIP}:${ServerPort}

Content-Length	${XXX} ...

Annotation

HTTP Request Method	POST method
URI	/iclock/ cdata
HTTP Protocol Version	1.1

Parameter Description

Parameter	Description
Table type	tabledata
Table name	user
count	The number of fingerprint templates in one request.
Request entity	${DataRecord}, the user data, in the format of: templatev10 size=${XXX}$(HT)uid=${XXX}$(HT)pin=${XXX}$(HT)fingerid=${XXX}$(HT)valid=${XXX}$(HT)template=${XXX}$(HT)resverd=${XXX}$(HT)endtag=${XXX}}
templatev10	Indicate that the data type is a fingerprint template. The supported fingerprint algorithm version is no later than 10.0
size	The size of the base64-encoded fingerprint template
uid	The fingerprint template ID in the device
pin	The ID of the user corresponding to the fingerprint template
fingerid	The fingerprint ID of each user, which is used to distinguish different fingers. 0 to 9 indicate a common fingerprint. When 16 is added to a common fingerprint ID, such as 6+16=22, it indicates a duress fingerprint. 16 must be added when duress fingerprints in the device are uploaded to the software.

valid	The duress fingerprint mark in the device. 0 indicates an invalid fingerprint, 1 indicates a common fingerprint, and 3 indicates a duress fingerprint. The software distinguishes common fingerprints from duress fingerprints by fingerid. If the valid value is 3 when the fingerprint is uploaded to the software, change the value to 1 first.
template	The fingerprint template data. Before a fingerprint template is transmitted, the original binary fingerprint template must be base64-encoded.
resverd	-
endtag	-
${LF} is used to connect multiple records.	

Server Response

```
HTTP/1.1 200 OK
Server: Apache-Coyote/1.1
Content-Length: ${XXX}
Date: ${XXX}
templatev10: ${XXX}
(The number of fingerprint templates using the 10.0 algorithm received
 by the server).
```

Example:

Client Request

HTTP	POST/iclock/cdata?SN=3383154200002&table=tabledata&tablename=templatev10&count=2 HTTP/1.1
Cookie	token=af65a75608cf5b80fbb3b48f0b4df95a
Host	192.168.213.17:8088
User-Agent	iClock Proxy/1.09
Connection	starting
Accept	application/push

Accept-Charset	UTF-8
Accept-Language	zh-CN
Content-Type	application/push;charset=UTF-8
Content-Language	zh-CN
Content-Length	3900

templatev10 size=1808 uid=4 pin=4 fingerid=7 valid=1
template=TAlTUzIxAAAFSksECAUHCc7QAAAdS2kBAAAAhfcvU0o3AIAPswD/AIt
FJQBWAG0PcABfSo0PogBhAE0PeEpnAIEPqACjAAlFcAB0AAQPvwCBSoEPvACTANE
PGUqcAOoPSgBkAGtEOwCpAO0OAACpSo8PFgC9AKwPWUrBAGcPLAAPAFpFjQDNAHY
PEADIShoPagDRAKcPf0riAFcPLQAtAFxElADtAGEPpwDqSkoP3ADxAOMP/krzACM
PmgA8ABRFHgD8ANkO+AAFS00PLAACAZUOx0oDAZgOdADPATdFvAAOAY8OnAAKSzo
PyAAVAU8NT0oWAT8PowDkAQNF3wAhAYENUQApSxIPQQAwAfsPBUs5AYIOUAD4AS5
FEAE9AYwO8QBES0AOewBHAd8PR0pRATIPGwCWAV9FahcnAzN/9wRDQT+H/+9bh+J
rUMtkBbIAFH4s/XNDXH1GAPr51YZQy/ePqYVZhap9qDfHAOf5oYvaCB7Ep/zegJt
8RIE7Q5aEgYEmBCORoDUjdV8bLQ8ngAM0PAbG+bJ+fPtnQvN1aSNCFEMVE8ef9Yf
5wf6EDm9SzPaZDwXjRIJsQVAPUFOtDxPsIL0D6F4YXgQDkBNH4/z/Cbem5bMP9Ie
GuYiBgmuMtLZgD933wXjoBPSxOQsdEdJ0DPv/ohfewPsIB5+ItLY8C9rwjP9s9Qh
ALATt95EDJPVnMm9+lfqW9t8CUEfP+RPgYRVY/RRJCPmaApf7MO20sNDvgQ2LD25
rZFsYBTkbRgk3805c8PEpCg70yxgjYAodP0PuHtgjJQQAAvIoUg3FOQVahMBKQ/8
FxYoHQz4IAHoHCY5TCkokD/04wMAF/m2KYAsAKhYDllFJRAELK/TBMTpHx30TAAk
y/cP1/lYRwP/AcQYAlzKGi8NpCwBaNsNCxQF2BgBSOH29wQFKtjsTfQ0A0VDxHUD
AVf8LAO1U9XE8WwQAuVvJQQlKn16Mc8GAB2QCSqdhAzZMDMWYZ8xThHfACQC8Y4P
KaMEKAIFmw0/EBsEHAKtnCTvB+hMIAHloff9GcQlKdHQGXsH/BDb6QgF2gobCwUb
BAEp/hQA2CgC3h4U0wsE/BAAsSWdeWQESjun//4L+bbVC/0wGAL9XDPiLWwYAv5c
TgcEQSguc58AvwPZgXYr8ZBEAwatWwnfBw8H/wsCF1QC25o1xg3HB/lEJBf+sDP9
DSxbFGr+jaf5MNv7/OlVitxcABb7i/pP/NwtHRMBNBQCdwWzOEwDRypeEBcKDtMO
TZwQAKQ5gakQBkMwDLv8FMsSKmgcA2cwW8jwASmfQacKICMWJ1T3AxMHBkwTF19R
QQw0AaNVkB8GBFsBbDwB35KXFicLEUMHCEgBU6nKNxcbBwcN8VsI5RwFm6977wDr
8K3L+BACZ7QPnDgXY8WfFyMLCAcDGi8P/wP8DADryG7UKAF7zU8M6w8fCgwsAZPN
MBMRsw8MGAN3zJJDBH0sU9KBqc2+vw8QuxXLB/8EExfzzbW4FAJ33DDh2AUqU+S3
GwgPFnPhZwQUQHABWBIwBWogBMKEIEPwDVTFZBxA/BEy2TgJaKgVWgyYF1ZIDasK
UGRDTClv+xInDxP+ywcIBgvqIw//DwH8Q1XQIfonBwYh+weMJFRMTQMPAloHCEEJ
QQnDEwQMQiRo4igQQpx8DwAb8GlsUIaTB/nSuwoUliIPA/8HCr/sCWqYlDJAzCdW
YK13DjF4gERXrpMR9d8CIhMKGBmLHi/7Bwf39+cEQSgo2nQQQVEDogwBaf0oewnM
E1f9hN0tSQgALQ8QABUFEUg== resverd= endtag=

templatev10 size=1928 uid=3 pin=4 fingerid=6 valid=1

```
template=TOVTUzIxAAAFpqkECAUHCc7QAAAdp2kBAAAAhUs1zKYXAPUPzgD2AP+
pCgE6AAYOFgBDpgEPtQBSAL0Pk6ZVAPYP8ACRAAOpmABoAPQPDABopgcPNgCCACs
O2KaHAIYPqQBNAPiprACTAHYPMQCRphMPrwCeAMcPN6alAOoOEQBvAOetQQCqAGo
O2wC3pmQLpACyALMPI6a4AOoLZgAEAPWpvwDJAA0PxAHRphwOqADWAM4PEqbcAOI
PuQAnAISpowDlAAMPjwDjpukPYQDqACsPq6buABEP/wA6ACapaAAAAfMP1AADp2U
PkQALAbIPqqYPASAPbgDRAfGplQAVARYPvAAZp/8PFQAgAaIOaKYmAf8P9wDiASi
plQAqAS8O3AA2p+IOpgAzAf0OZ6Y0Af4P3ADyATGpbAA4AXgO9gBMp+UPZQBMAa8
Ps6ZUAS4NoACZAUKowQBdASMNAP/CUtp/vf1BByeG26fyCgMPNQlT/Pcuh4T6eDJ
1AAWPJr76DgheASIATKdzfbr5ooDnkUOs2wXKhueK1HenJnIRtIu5/vcJQdG9e1Y
ABY/KDyqpOY81BtWObIDopPyGtXy5+1/5rCaDgI4FOH90hADY4HbWACpp8YdopDA
L1gYqiyoIuCjcBxKZpv/aD+61xAiRjWEDt/rjJGd/XXCZebyCnFJM+HmGof5z9nJ
eGArF/loCyBBINiQLpvoiC3LzwKdABrr0Vf8jhlImPWNxew5YEA/jDiMQpfjR9YA
BJQWs6WH7HQwgAHxNUIXVB0OFoADELtwBvvgeDiYPYF8QGeYwVX1/AVIm3An2JXo
OSYtYp7yWOginEHYIkdnIeFkSOgFmfULb5PbRbuJuqQ/ATAP6LRo+JMcS/VHj3Dv
wsHjtYw+9ASBQAQMF4k4Apr8BdML/wMAAx6Fq/0sLAOfCgMFk/GvAbwMAKgxyZgw
A8BR3wKNWbmcQAQQeg34HUsVZcVQSAQsmRXFh+8P/a2UGAdcsf1llEgEMNICv/4F
YxUPBVAUAEkYFZysKARVHkwbAjP4GALlQ9y+CCAU0VHBq/1cNxbFQ1v/CwGvB/zj
Gx6UB9FUJwgrFnWZSwcH9//5RyACQz3DBWsDB/gD+N6wBnWr0wP86wPlmSgcAzW0
A9EYWpxR+l4WWwQXAxWRkbQQArYQ/MAimpIl6wsBpl8BloAGtigBK/sgArDZ2c37
AwMC+AwVekxDBAwAZUlf7rwGxk/r+//BUHacNk5NtecM7wcRma2vA/cXDBBYEsJW
ew/+HwFxqbviYDgCpl3cEa3bPbgkAs5r97jf7qQGrn3p+agXBxlhdCACzoAaRNgC
mobB6wsDBzwCtFPw1//7AV8YAkB92wg8Aabw1wDiJ/v82wAQA08BnYP4KAGLDab1
kUqoBdMV0icI7wcVn/vwJAMHKzP5E8AcAPNBkcwYLBQrSBv7/OP+GBgUC13qEwAw
AadkMWP7/wDZCDcW25CCIc8NnwBLFTeFBVf9GwP7+OjtRtwFk5uvAODr9+1lH/8I
xCwBy5wxb/j7/VQgAmO1oZHFXBwCy7cz9+ljB/xAAqu5JxMdZw39w/sIjyQC0VBI
4NcBGGsQS8gRrwnnAfsKx/4P8NREAbf7w8P34Z/81wEDBA9QCBIT/EBCrC4xUwYJ
kT/0/CRCzyBf4Zv8ywggQsdceNllzBxCXFgzkOQ22fxv9/f78O/3FoxFwIv3/Ks0
QdI4B/xUfCBCsKmFZwET/BRD77yk4oxH2Ky1VBdWYKYv8HwQQZTQy//+iEaU2Ojk
E1aozlikGEGk5bQdRAraCOYDHQ//BENicMEwGEH8/rMLFlAcQjkN6xzrA/rcRNUf
nwVs5wvhm/fz+/jYD1Y9O6vwDEINMXgQFFcVPaf/A/RrVAVxPwYH/wUb9jv74mv5
LX1JCAM5DBKYBC0VS      resverd=       endtag=
```

Server Response

```
HTTP/1.1 200 OK
Server: Apache-Coyote/1.1
Content-Length: 13
Date: Thu, 12 Jan 2017 00:49:32 GMT
```

```
templatev10=2
```

7.8 Upload Comparison Photo

Application scenario

The visible light device automatically uploads the registered user comparison photos to the server.

Client Request

HTTP	POST/iclock/cdata?SN=${SerialNumber}&table=tabledata &tablename=biophoto&count=${XXX} HTTP/1.1
Cookie	token=${XXX}
Host	${ServerIP}:${ServerPort}
Content-Length	${XXX} … ${DataRecord}

Annotation

HTTP Request Method	POST method
URI	/iclock/ cdata
HTTP Protocol Version	1.1

Parameter Description

Parameter	Required/ Optional	Description
count	${XXX}	The number of comparison photos in one request.
Host header	${Required}	-
Other headers	${Optional}	-

<table>
<tr><td>Content-Length header</td><td>${Required}</td><td>-</td></tr>
<tr><td>Request entity</td><td>${DataRecord}</td><td>The comparison photo data, in the format of:
biophoto${SP}pin=${XXX}${HT}filename=${XXX}${HT}type=${XXX}${HT}size=${XXX}${HT}content=${XXX}</td></tr>
<tr><td>pin</td><td>${XXX}</td><td>The user ID.</td></tr>
<tr><td>filename</td><td>${XXX}</td><td>The file name of the biometric photo. Currently, only photos in JPG format are supported.</td></tr>
<tr><td>type</td><td>${XXX}</td><td>The biometric type.
<table>
<tr><th>Value</th><th>Meaning</th></tr>
<tr><td>0</td><td>General</td></tr>
<tr><td>1</td><td>Fingerprint</td></tr>
<tr><td>2</td><td>Face (Near Infrared)</td></tr>
<tr><td>3</td><td>Voice Print</td></tr>
<tr><td>4</td><td>Iris</td></tr>
<tr><td>5</td><td>Retina</td></tr>
<tr><td>6</td><td>Palm Print</td></tr>
<tr><td>7</td><td>Finger Vein</td></tr>
<tr><td>8</td><td>Palm</td></tr>
<tr><td>9</td><td>Visible Light Face</td></tr>
</table></td></tr>
<tr><td>size</td><td>${XXX}</td><td>The length of the base64-encoded biometric photo</td></tr>
<tr><td>content</td><td>${XXX}</td><td>The original binary biometric photo must be base64-encoded before transmission</td></tr>
<tr><td colspan="3">${LF} is used to connect multiple records.</td></tr>
</table>

Server Response

```
HTTP/1.1 200 OK
Content-Length: ${XXX}
…
biophoto: ${XXX}
```

Note

Parameter	Description
HTTP status line	It is defined by the standard HTTP.
HTTP response header	-
Content-Length header	Content-Length header: According to HTTP 1.1, this header is generally used to specify the length of the response entity. If you do not know the length, you can also set Transfer-Encoding to chunked. Both Content-Length and Transfer-Encoding are headers defined by standard HTTP, which are not described in detail here
Response entity	When the server receives the data and processes it normally, it returns `biophoto=${XXX}`, in which `${XXX}` indicates the number of records that are successfully processed. When an error occurs, the server returns the error description.

Example:

Client Request

HTTP	POST/iclock/cdata?SN=0316144680030&table=tabledata&tablename=biophoto&count=1 HTTP/1.1
Host	58.250.50.81:8011
User-Agent	iClock Proxy/1.09
Connection	close
Accept	*/*

Content-Length	95040
biophoto pin=123 filename=123.jpg type=9 size=95040 content=AAAA........	

Server Response

```
HTTP/1.1 200 OK
Server: nginx/1.6.0
Content-Type: text/plain
Content-Length: 4
Connection: close
Pragma: no-cache
Cache-Control: no-store
Date: Thu, 30 Jul 2015 07:25:38 GMT
biophoto=1
```

7.9 Upload Snapshot

Application scenario

The visible light device automatically uploads snapshots to the server.

Client Request

HTTP	POST/iclock/cdata?SN=${SerialNumber}&table=tabledata &tablename=ATTPHOTO&count=${XXX} HTTP/1.1
Cookie	token=${XXX}
Host	${ServerIP}:${ServerPort}
Content-Length	${XXX} … ${DataRecord}

Annotation

HTTP Request Method	POST method
URI	/iclock/ cdata
HTTP Protocol Version	1.1

Parameter Description

Parameter	Required/ Optional	Description
count	${XXX}	The number of comparison photos in one request.
Host header	${Required}	-
Other headers	${Optional}	-
Content-Length header	${Required}	–
Request entity	${DataRecord}	The attendance photo data, in the format of: pin=${XXX}${HT}sn=${XXX}${HT}size=${XXX}${HT}photo=${XXX}
pin	${XXX}	The name of the attendance photo.
sn	${XXX}	The device SN.
size	${XXX}	The original size of the snapshot
photo	${XXX}	The original binary biometric photo must be base64-encoded before transmission.
${LF} is used to connect multiple records.		

Server Response

```
HTTP/1.1 200 OK
Content-Length: ${XXX}
…
```

```
ATTPHOTO: ${XXX}
```

Note

Parameter	Description
HTTP status line	It is defined by the standard HTTP.
HTTP response header	-
Content-Length header	Content-Length header: According to HTTP 1.1, this header is generally used to specify the length of the response entity. If you do not know the length, you can also set Transfer-Encoding to chunked. Both Content-Length and Transfer-Encoding are headers defined by standard HTTP, which are not described in detail here
Response entity	When the server receives the data and processes it normally, it returns `biophoto=${XXX}`, in which `${XXX}` indicates the number of records that are successfully processed. When an error occurs, the server returns the error description.

Example:

Client Request

HTTP	POST/iclock/cdata?SN=1809140006&table=tabledata&tablename=ATTPHOTO&count=1 HTTP/1.1
Host	58.250.50.81:8011
User-Agent	iClock Proxy/1.09
Connection	close
Accept	*/*
Content-Length	30327
pin=20181003143617-22.jpg sn=1809140006 size=22701 photo=/9j/4AAQSkZJRgABAQAAAQABAAD/2wBDABsSFBcUERsXFhceHBsgKEIrKCUlKFE6PTBCYFVlZF9VXVtqeJmBanGQc1tdhbWGkJ6jq62rZ4C8ybqmx5moq6T/2w	

```
BDARweHigjKE4rK06kbl1upKSkpKSkpKSkpKSkpKSkpKSkpKSkpKSkpKSkpK
SkpKSkpKSkpKSkpKSkpKSkpKT/wAARCAUAAtADASIAAhEBAxEB/8QAHwAAAQUBAQ
EBAQEAAAAAAAAAAAECAwQFBgcICQoL/8QAtRAAAgEDAwIEAwUFBAQAAAF9AQIDAA
QRBRIhMUEGE1FhByJxFDKBkaEII0KxwRVS0fAkM2JyggkKFhcYGRolJicoKSo0NT
Y3ODk6Q0RFRkdISUpTVFVWV1hZWmNkZWZnaGlqc3R1dnd4eXqDhIWGh4iJipKTlJ
WWl5iZmqKjpKWmp6ipqrKztLW2t7i5usLDxMXGx8jJytLT1NXW19jZ2uHi4+Tl5u
fo6erx8vP09fb3+Pn6/8QAHwEAAwEBAQEBAQEBAQAAAAAAAAECAwQFBgcICQoL/8
QAtREAAgECBAQDBAcFBAQAAQJ3AAECAxEEBSExBhJBUQdhcRMiMoEIFEKRobHBCS
MzUvAVYnLRChYkNOEl8RcYGRomJygpKjU2Nzg5OkNERUZHSElKU1RVVldYWVpjZG
VmZ2hpanN0dXZ3eHl6goOEhYaHiImKkpOUlZaXmJmaoqOkpaanqKmqsrO0tba3uL
m6wsPExcbHyMnK0tPU1dbX2Nna4uPk5ebn6Onq8vP09fb3+Pn6/9oADAMBAAIRAx
EAPwBXXIPFpVpI+fbHWnqWZ
```

Server Response

```
HTTP/1.1 200 OK
Server: nginx/1.6.0
Content-Type: text/plain
Content-Length: 10
Connection: close
Pragma: no-cache
Cache-Control: no-store
Date: Thu, 30 Jul 2015 07:25:38 GMT
ATTPHOTO=1
```

7.10 Upload User Photo

Application scenario

The device automatically uploads user photos to the server.

Client Request

HTTP	POST/iclock/cdata?SN=${SerialNumber}&table=tabledata &tablename=userpic&count=${XXX} HTTP/1.1

Cookie	token=${XXX}
Host	${ServerIP}:${ServerPort}
Content-Length	${XXX} … ${DataRecord}

Annotation

HTTP Request Method	POST method
URI	/iclock/ cdata
HTTP Protocol Version	1.1

Parameter Description

Parameter	Required/ Optional	Description
count	${XXX}	The number of user photos in one request.
Host header	${Required}	-
Other headers	${Optional}	-
Content-Length header	${Required}	–
Request entity	${DataRecord}	the user photo data, in the format of: userpic pin=${XXX}${HT}filename=${XXX}${HT}size=${XXX}${HT}content=${XXX}
pin	${XXX}	The user ID.
filename	${XXX}	The file name of the user photo. Currently, only photos in JPG format are supported.
size	${XXX}	The length of the base64-encoded user photo.
content	${XXX}	The original binary user photo must be

		base64-encoded before transmission.
${LF} is used to connect multiple records.		

Server Response

```
HTTP/1.1 200 OK
Content-Length: ${XXX}
…
userpic: ${XXX}
```

Note

Parameter	Description
HTTP status line	It is defined by the standard HTTP.
HTTP response header	-
Content-Length header	According to HTTP 1.1, this header is generally used to specify the length of the response entity. If you do not know the length, you can also set Transfer-Encoding to chunked. Both Content-Length and Transfer-Encoding are headers defined by standard HTTP, which are not described in detail here.
Response entity	When the server receives the data and processes it normally, it returns userpic=${XXX}, in which ${XXX} indicates the number of photos that are successfully processed. When an error occurs, the server returns the error description.

Example:

Client Request

HTTP	POST/iclock/cdata?SN=1809140006&table=tabledata&

	tablename=userpic&count=1 HTTP/1.1
Host	58.250.50.81:8011
User-Agent	iClock Proxy/1.09
Connection	close
Accept	*/*
Content-Length	29848

```
userpic pin=1 filename=1.jpg       size=22320
content=/9j/4AAQSkZJRgABAQAAAQABAAD/2wBDABsSFBcUERsXFhceHBsgKEIr
KCUlKFE6PTBCYFVlZF9VXVtqeJmBanGQc1tdhbWGkJ6jq62rZ4C8ybqmx5moq6T/
2wBDARweHigjKE4rK06kbl1upKSkpKSkpKSkpKSkpKSkpKSkpKSkpKSkpKSk
pKSkpKSkpKSkpKSkpKSkpKT/wAARCAOoAnQDASIAAhEBAxEB/8QAHwAAAQUB
AQEBAQEAAAAAAAAAAAECAwQFBgcICQoL/8QAtRAAAgEDAwIEAwUFBAQAAAF9AQID
AAQRBRIhMUEGE1FhByJxFDKBkaEII0KxwRVS0fAkM2JyggkKFhcYGRolJicoKSo0
NTY3ODk6Q0RFRkdISUpTVFVWV1hZWmNkZWZnaGlqc3R1dnd4eXqDhIWGh4iJipKT
lJWWl5iZmqKjpKWmp6ipqrKztLW2t7i5usLDxMXGx8jJytLT1NXW19jZ2uHi4+Tl
5ufo6erx8vP09fb3+Pn6/8QAHwEAAwEBAQEBAQEBAQAAAAAAAAECAwQFBgcICQoL
/8QAtREAAgECBAQDBAcFBAQAAQJ3AAECAxEEBSExBhJBUQdhcRMiMoEIFEKRobHB
CSMzUvAVYnLRChYkNOEl8RcYGRomJygpKjU2Nzg5OkNERUZHSElKU1RVVldYWVpj
ZGVmZ2hpanN0dXZ3eHl6goOEhYaHiImKkpOUlZaXmJmaoqOkpaanqKmqsrO0tba3
uLm6wsPExcbHyMnK0tPU1dbX2Nna4uPk5ebn6Onq8vP09fb3+Pn6/9oADAMBAAIR
AxEAPwDUoooqQCiigAooooAK
```

Server Response

```
HTTP/1.1 200 OK
Server: Apache-Coyote/1.1
Content-Type: text/plain;charset=utf-8
Content-Length: 24
Connection: close
Date: Thu, 08 Nov 2018 06:16:41 GMT
userpic=1
```

7.11 Upload Integrated Template

Application scenario

New biometric templates, such as integrated palm templates, will be uploaded and downloaded in a unified format and will be distinguished by the data type.

Client Request

HTTP	POST/iclock/cdata?SN=${SerialNumber}&table=tabledata &tablename=biodata&count=${XXX} HTTP/1.1
Cookie	token=${XXX}
Host	${ServerIP}:${ServerPort}
Content-Length	${XXX} … ${DataRecord}

Annotation

HTTP Request Method	POST method
URI	/iclock/ cdata
HTTP Protocol Version	1.1

Parameter Description

Parameter	Required/ Optional	Description
count	${XXX}	The number of integrated templates in one request.
Host header	${Required}	-
Other headers	${Optional}	-

<table>
<tr><td>Content-Length header</td><td>${Required}</td><td>-</td></tr>
<tr><td>Request entity</td><td>${DataRecord}</td><td>The integrated template, in the format of:
biodatapin=${XXX}${HT}no=${XXX}${HT}index=${XXX}${HT}valid=${XXX}${HT}duress=${XXX}${HT}type=${XXX}${HT}majorver=${XXX}${HT}minorver=${XXX}${HT}format=${XXX}${HT}tmp=${XXX}</td></tr>
<tr><td>pin</td><td>${XXX}</td><td>The user ID.</td></tr>
<tr><td>no</td><td>${XXX}:</td><td>The number of the specific biont. The default value is 0.
<table>
<tr><th>Biometric type</th><th>Description</th></tr>
<tr><td>Fingerprint</td><td>The number ranges from 0 to 9, corresponding to the little finger, ring finger, middle finger, index finger, and thumb of the left hand, and thumb, index finger, middle finger, ring finger, and little finger of the right hand respectively.</td></tr>
<tr><td>Finger Vein</td><td>Same as the fingerprints.</td></tr>
<tr><td>Face</td><td>All is 0.</td></tr>
<tr><td>Iris</td><td>0 is the iris of the left eye and 1 is the iris of the right eye.</td></tr>
<tr><td>Palm</td><td>0 is the palm of the left hand and 1 is the palm of the right hand.</td></tr>
<tr><td>Visible Light Palm</td><td>0 is the palm of the left hand and 1 is the palm of the right hand.</td></tr>
</table></td></tr>
</table>

<table>
<tr><td></td><td></td><td></td></tr>
<tr><td>index</td><td>${XXX}</td><td>The number of the specific biont template. Multiple templates may be stored for one finger. It is calculated from 0.</td></tr>
<tr><td>valid</td><td>${XXX}</td><td>Whether it is valid. The default value is 1.
0 Invalid
1 Valid</td></tr>
<tr><td>duress</td><td>${XXX}</td><td>Whether it is duress. The default value is 0.
0 Not Duress
1 Duress</td></tr>
<tr><td>type</td><td>${XXX}</td><td>The biometric type.
<table><tr><th>Value</th><th>Meaning</th></tr><tr><td>0</td><td>General</td></tr><tr><td>1</td><td>Fingerprint</td></tr><tr><td>2</td><td>Face (Near Infrared)</td></tr><tr><td>3</td><td>Voice Print</td></tr><tr><td>4</td><td>Iris</td></tr><tr><td>5</td><td>Retina</td></tr><tr><td>6</td><td>Palm Print</td></tr><tr><td>7</td><td>Finger Vein</td></tr><tr><td>8</td><td>Palm</td></tr><tr><td>9</td><td>Visible Light Face</td></tr><tr><td>10</td><td>Visible Light Palm</td></tr></table></td></tr>
<tr><td>majorver</td><td>${XXX}</td><td>It is the major version number. For example, for the fingerprint algorithm 10.3, the major version number is 10 and the minor version number is 3.
<table><tr><th>Biometric type</th><th>Version No.</th></tr><tr><td>Fingerprint</td><td>9.0, 10.3, and 12.0</td></tr></table></td></tr>
</table>

<table>
<tr><td></td><td></td><td>
<table>
<tr><td>Finger Vein</td><td>3.0</td></tr>
<tr><td>Face</td><td>5.0, 7.0, and 8.0</td></tr>
<tr><td>Palm</td><td>1.0</td></tr>
<tr><td>Visible Light Face</td><td>58.0 and 38.0</td></tr>
<tr><td>Visible Light Palm</td><td>32.2</td></tr>
</table>
</td></tr>
<tr><td>minorver</td><td>${XXX}</td><td>The minor version number. For example, for the fingerprint algorithm 10.3, the major version number is 10 and the minor version number is 3.
<table>
<tr><th>Biometric type</th><th>Version No.</th></tr>
<tr><td>Fingerprint</td><td>9.0, 10.3, and 12.0</td></tr>
<tr><td>Finger Vein</td><td>3.0</td></tr>
<tr><td>Face</td><td>5.0, 7.0, and 8.0</td></tr>
<tr><td>Palm</td><td>1.0</td></tr>
<tr><td>Visible Light Face</td><td>58.0 and 38.0</td></tr>
<tr><td>Visible Light Palm</td><td>32.2</td></tr>
</table>
pin=${XXX}: The name of the attendance photo.</td></tr>
<tr><td>format</td><td>${XXX}</td><td>The template format. The fingerprint template formats include ZK, ISO, and ANSI.
<table>
<tr><th>Biometric type</th><th>Value</th><th>Format</th></tr>
<tr><td>Fingerprint</td><td>0
1
2</td><td>ZK
ISO
ANSI</td></tr>
<tr><td>Finger Vein</td><td>0</td><td>ZK</td></tr>
<tr><td>Face</td><td>0</td><td>ZK</td></tr>
</table>
</td></tr>
</table>

		Palm: 0: ZK Visible Light Palm: 0: ZK
tmp	${XXX}	The template data. The original binary fingerprint template must be base64-encoded.
${LF} is used to connect multiple records.		

Server Response

```
HTTP/1.1 200 OK
Content-Length: ${XXX}
…
biodata: ${XXX}
```

Note

Parameter	Description
HTTP status line	It is defined by the standard HTTP.
HTTP response header	-
Content-Length header	According to HTTP 1.1, this header is generally used to specify the length of the response entity. If you do not know the length, you can also set Transfer-Encoding to chunked. Both Content-Length and Transfer-Encoding are headers defined by standard HTTP, which are not described in detail here.
Response entity	When the server receives the data and processes it normally, it returns userpic=${XXX}, in which ${XXX} indicates the number of photos that are successfully processed. When an error occurs, the server returns the error description.

Example:

Client Request

HTTP	POST/iclock/cdata?SN=5165181600015&table=tabledata&tablename=biodata&count=1 HTTP/1.1
Host	58.250.50.81:8011
User-Agent	iClock Proxy/1.09
Connection	starting
Accept	application/push
Accept-Charset	UTF-8
Accept-Language	zh-CN
Content-Type	application/push;charset=UTF-8
Content-Language	zh-CN
Content-Length	3564
biodata pin=1 no=0 index=0 valid=1 duress=0 type=8 majorver=5 minorver=0 format=0 tmp=apUBD+cAC44IAAEAAJfKWlBWAQQFAAAAAAAAEAcHAcQAAAA3PJc4PO4W/AxrTnYNXQ7cJ0+ObT0vBE81KyGPkzSHuh7ul+wP647kWNnMTDA8WMwlanRcNTpzTjc2hzUGZ488FweSBlaN0kY2GdZeNHlVXCsx81nI5M0kZ2YzJD2kMxaWrBIWM9hWXEt4VzlLNP4JzMdLF6OUY5Kn8GPepNRjUp+0Z2BPOkd5STsfeZuXaxaKzU0WvnUk06a2L1PrIu1jzwLvYf3U6wBrBG80yLbMc+g0DPL9MClyPTCNk50Fzws8xBsN	

Server Response

```
HTTP/1.1 200 OK
Server: Apache-Coyote/1.1 is not blacklisted
Content-Length: 9
Date: Fri, 19 Oct 2018 06:55:30 GMT
biodata=1
```

7.12 Upload Error Log

Application scenario

The device automatically uploads error logs to the server. The value of PushProtVer delivered by the server is greater to or equal to 3.1.2.

Client Request

HTTP	POST/iclock/cdata?SN=${SerialNumber}&table=tabledata &tablename=errorlog&count=${XXX} HTTP/1.1
Cookie	token=${XXX}
Host	${ServerIP}:${ServerPort}
Content-Length	${XXX} … ${DataRecord}

Annotation

HTTP Request Method	POST method
URI	/iclock/ cdata
HTTP Protocol Version	1.1

Parameter Description

Parameter	Required/ Optional	Description
count	${XXX}	The number of error logs in one request.
Host header	${Required}	-
Other headers	${Optional}	-
Content-Length	${Required}	-

header		
Request entity	${DataRecord}	The error log, in the format of: errcode=${XXX}$(HT)errmsg=${XXX}$(HT)dataorigin=${XXX}$(HT)cmdid=${XXX}$(HT)additional=${XXX}
errcode	${XXX}	The error code. See Appendix 17.
errmsg	${XXX}	The error message.
dataorigin	${XXX}	The data source. Dev indicates that the data is sourced from the device, and cmd indicates that the data is delivered by the device through a command.
cmdid	${XXX}	The number of the command delivered by the software.
additional	${XXX}	The base64-encoded additional information. The native data is in JSON format.
${LF} is used to connect multiple records.		

Server Response

```
HTTP/1.1 200 OK
Content-Length: ${XXX}
…
errorlog: ${XXX}
```

Note

Parameter	Description
HTTP status line	It is defined by the standard HTTP.
HTTP response header	-
Content-Length header	According to HTTP 1.1, this header is generally used to specify the length of the response entity. If you do not know the length, you can also set Transfer-Encoding to chunked. Both Content-Length

	and Transfer-Encoding are headers defined by standard HTTP, which are not described in detail here.
Response entity	When the server receives the data and processes it normally, it returns `errorlog=${XXX}`, in which `${XXX}` indicates the number of error logs that are successfully processed. When an error occurs, the server returns the error description.

Example:

Client Request

HTTP	POST/iclock/cdata?SN=5165181600015&table=tabledata&tablename=errorlog&count=1 HTTP/1.1
Host	58.250.50.81:8011
User-Agent	iClock Proxy/1.09
Connection	starting
Accept	application/push
Accept-Charset	UTF-8
Accept-Language	zh-CN
Content-Type	application/push;charset=UTF-8
Content-Language	zh-CN
Content-Length	3564
errorlog errcode=D01E0001 errmsg= dataorigin=cmd cmdid=123 additional=	

Server Response

```
HTTP/1.1 200 OK
Server: Apache-Coyote/1.1 is not blacklisted
Content-Length: 9
Date: Fri, 19 Oct 2018 06:55:30 GMT
errorlog=1
```

7.13 Upload Cancel Alarm Events (only supported by the channel controller)

Application scenario

The device is in the alarm state, and the software sends a control command to the device to turn off the alarm. The event must be uploaded to the software as soon as the cancellation is complete.

Client Request

HTTP	POST /iclock/cdata?SN=$(SerialNumber)&table=rtlog HTTP/1.1
Cookie	token=${XXX}
Host	${ServerIP}:${ServerPort}
Content-Length	${XXX} … ${DataRecord}

Server Response

```
HTTP/1.1 200 OK
Server: Apache-Coyote/1.1
Content-Length: ${XXX}
Date: ${XXX}
OK
```

Annotation

HTTP Request Method	POST method
URI	/iclock/ cdata
HTTP Protocol Version	1.1

Parameter Description

Parameter	Description
Table name	rtlog
Request entity	${DataRecord}, the real-time event data in the format of: time=${Time}${HT}pin=${Pin}${HT}cardno=${XXX}${HT}sitecode=${XXX}${HT}linkid=${XXX}${HT}eventaddr=${XXX}${HT}event=${XXX}${HT}inoutstatus=${XXX}${HT}verifytype=${XXX}${HT}index=${XXX}${HT}Alarmtype=${XXX}\r\n
time	The time, in the format of XXXX-XX-XX XX:XX:XX.
pin	The user ID
cardno	The card number
sitecode	Location code
linkid	Linkage event ID. For example, in the event, if the person has not registered, the door will not open. If the software is configured to open the door when the unregistered person event occurs, the door will open as long as the device has a linked event.
eventaddr	The event points. The default value is DoorId.
event	The event code. , see Appendix 2.
inoutstatus	The in/out status. 0 indicates In, and 1 indicates Out.
verifytype	The current verification method.
index	The access control record ID. Each access control record has a unique ID in the device.
Alarmtype	Alarm type, currently there is only one value of 200 for the channel

Example:

Client Request

HTTP	POST /iclock/cdata?SN=3383154200002&table=rtlog HTTP/1.1
Cookie	token=cb386eb5f8219329db63356fb262ddff
Host	192.168.213.17:8088
User-Agent	iClock Proxy/1.09
Connection	starting
Accept	application/push
Accept-Charset	UTF-8
Accept-Language	zh-CN
Content-Type	application/push;charset=UTF-8
Content-Language	zh-CN
Content-Length	99
time=2017-01-10 11:49:32 pin=0 cardno=0 eventaddr=1 event=27 inoutstatus=1 verifytype=0 index=21	

Server Response

```
HTTP/1.1 200 OK
Server: Apache-Coyote/1.1
Content-Length: 2
Date: Tue, 10 Jan 2017 03:49:32 GMT
OK
```

7.14 Upload Channel Infrared Status and Device Status Count (only supported by the channel controller)

Application scenario

1. When the device is started for the first time, it is pushed to the software.

2. When the infrared state of the channel changes or the number of in and out of the gate and the number of operations changes, the device needs to be uploaded to the software immediately.

Client Request

HTTP	POST /iclock/cdata?SN=$(SerialNumber)&table=irstate HTTP/1.1
Cookie	token=${XXX}
Host	${ServerIP}:${ServerPort}
Content-Length	${XXX} … ${DataRecord}

Server Response

```
HTTP/1.1 200 OK
Server: Apache-Coyote/1.1
Content-Length: ${XXX}
Date: ${XXX}
OK
```

Annotation

HTTP Request Method	POST method
URI	/iclock/ cdata
HTTP Protocol Version	1.1

Parameter Description

Parameter	Description
Table name	irstate
Request entity	${DataRecord}, the infrared status and device count data in the format of: irstate=${XXX}${HT}irsensecount=${XXX}${HT}incount=${XXX}${HT}outcount=${XXX}${HT}alarmcount=${XXX}${HT}runcount=${XXX}${CR}${LF}
irstate	The infrared sensor status
irsensecount	Number of infrared sensors
incount	Number of in of the gate
outcount	Number of out of the gate
alarmcount	Number of alarms
runcount	Operation times of the gate

Example:

Client Request

HTTP	POST /iclock/cdata?SN=3383154200002&table=irstate HTTP/1.1
Cookie	token=cb386eb5f8219329db63356fb262ddff
Host	192.168.213.17:8088
User-Agent	iClock Proxy/1.09
Connection	starting
Accept	application/push
Accept-Charset	UTF-8
Accept-Language	zh-CN

Content-Type	application/push;charset=UTF-8
Content-Language	zh-CN
Content-Length	99
irstate=1 irsensecount=2 incount=2 outcount=2 alarmcount=1 runcount=1	

Server Response

```
HTTP/1.1 200 OK
Server: Apache-Coyote/1.1
Content-Length: 2
Date: Tue, 10 Jan 2017 03:49:32 GMT
OK
```

8. Download Commands

8.1 Download Cache Command

Application scenario

The client obtains generated commands from the server.

Client Request

HTTP	GET /iclock/getrequest?SN=${SerialNumber} HTTP/1.1
Cookie	token=${XXX}
Host	${ServerIP}:${ServerPort} …

Annotation

HTTP Request Method	GET method
URI	/iclock/ getrequest
HTTP Protocol Version	1.1

Server Response

```
HTTP/1.1 200 OK
Server: Apache-Coyote/1.1
Content-Type: text/plain;charset=UTF-8
Content-Length: ${XXX}
Date: ${XXX}
${CmdRecord}
```

Note

Parameter	Description
${CmdRecord}	The server response entity: the command, in the format of C: ${CmdID}:$(CmdDesc)${SP}$(CmdDetail)
CmdID	The command ID
CmdDesc	The command description. Commands are divided into data commands, control commands, and configuration commands.
CmdDetail	The command details. For more details about commands delivered by the server, see Details of Server Commands.
${LF} is used to connect multiple records.	

Example:

Client Request

HTTP	GET /iclock/getrequest?SN=3383154200002 HTTP/1.1
Cookie	token=1637f0b091af92b0cc66b8eede0ae48e
Host	192.168.213.17:8088
User-Agent	iClock Proxy/1.09
Connection	starting
Accept	application/push
Accept-Charset	UTF-8
Accept-Language	zh-CN

Server Response

```
HTTP/1.1 200 OK
Server: Apache-Coyote/1.1
Content-Type: text/plain;charset=UTF-8
```

```
Content-Length: 99
Date: Wed, 11 Jan 2017 01:30:03 GMT
C:223:SET OPTIONS IPAddress=192.168.213.222,GATEIPAddress=192.168.21
3.1,NetMask=255.255.255
```

9. Server Commands

For the device operation, the server must generate a command and then send the command to the device after the device sends a request. The execution results of commands are already described in Upload Returned Result of the Command. The formats of the requests sent by the client and the server response are already described in Download Cache Command.

9.1 Data Commands

There are several data commands used to create, modify, and remove database objects. The following describes the command format in detail.

9.1.1 Update

It is used to add or update data to the device. If the device does not have the primary key data, you can run this command to add the data to the device; otherwise, you can run this command to update the data to the device.

Command format

Single Data Record Format:

```
C:${CmdID}:DATA$(SP)UPDATE$(SP)table name$(SP)field
name1=value1$(HT)field name2=value2$(HT)field name3=value3
```

Multiple Data Record Format:

```
C:${CmdID}:DATA$(SP)UPDATE$(SP)table name$(SP)field
name1=value1$(HT)field name2=value2$(HT)field name3=value3$(LF)field
name4=value4$(HT)field name5=value5$(HT)field name6=value6
```

Wherein, 123 indicates the first data record, 456 indicates the second data record, and they are separated by the line feeder $(LF).

User Information

Application scenario

The server delivers the user information to the client. It is generally used to synchronize the user information edited on the server to the client.

Command format

The server delivers the command:

```
C:${CmdID}:DATA$(SP)UPDATE$(SP)user$(SP)CardNo=${XXX}$(HT)Pin=${XXX}$(HT)
Password=${XXX}$(HT)Group=${XXX}$(HT)StartTime=${XXX}$(HT)EndTime=${XXX}$
(HT)Name=${XXX}$(HT)Privilege=${XXX}
```

The client uploads the execution result:

```
ID=${CmdID}&Return=${XXX}&CMD=DATA UPDATE
```

Annotation

Parameter	Description
user	The table name, which is the user.
CardNo	The card number, which is an unsigned integer. Values delivered by the software can be delivered in two formats: • Hexadecimal data, in the format of [%02x%02x%02x%02x], which means the first to the fourth bytes from left to right. For example, if the card number is 123456789, then Card=[15CD5B07]. • A string. If the card number is 123456789, then Card=123456789. • When MulCardUserFunOn is enabled (refer to the value of AccSupportFunList), this table field is invalid, and the number of MulCardUser protocol storage card is used. (This field is invalid if the device supports multi-card open door (DoorMultiCardOpenDoor=1), then please deliver the Multi-card Opening Information command in 9.1.1 to issue the card number.)
Pin	User ID
Password	Password
Group	User group (the user belongs to group 1 by default)
StartTime	The start time of the user validity period: • If the DateFmtFunOn value is 1, for example, StartTime=583512660, StartTime is calculated according to the algorithm in Appendix 5. • If the DateFmtFunOn value is 0, the format is YYYYMMDD. • If the value is 0, this user's start time has no limit.
EndTime	The end time of the user validity period: • If the DateFmtFunOn value is 1, for example, EndTime=583512660, EndTime

<table>
<tr><td></td><td>is calculated according to the algorithm in Appendix 5.
• If the DateFmtFunOn value is 0, the format is YYYYMMDD.
• If the value is 0, this user's end time has no limit.</td></tr>
<tr><td>Name</td><td>It is the user name. When the device language is Chinese, it is encoded with the GB2312 character set; otherwise, it is encoded with the UTF-8 character set.</td></tr>
<tr><td>Privilege</td><td>User privilege values have the following meanings respectively:<table><tr><td>0</td><td>Common User</td></tr><tr><td>2</td><td>Registrar</td></tr><tr><td>6</td><td>Administrator</td></tr><tr><td>10</td><td>User-defined</td></tr><tr><td>14</td><td>Super Administrator</td></tr></table></td></tr>
<tr><td colspan="2">${LF} is used to connect multiple records.</td></tr>
</table>

Example

The software delivers the information about a user, of which the user ID is 1 and the password is 234. The user has no card, belongs to the default group, and has the privilege of a common user.

No validity period is set for the user:

```
C:295:DATA UPDATE user CardNo=  Pin=1   Password=234    Group=0
StartTime=0 EndTime=0   Name=   Privilege=0
```

The client uploads the successful execution result:

```
ID=295&return=0&CMD=DATA UPDATE
```

Extended User Information

Application scenario

The server delivers the extended user information to the client. It is generally used to synchronize the user information edited on the server to the client.

The command format is as follows:

The server delivers the command:

```
C:${CmdID}:DATA$(SP)UPDATE$(SP)extuser$(SP)Pin=${XXX}$(HT)FunSwitch=${XXX}$(HT)FirstName=${XXX}$(HT)LastName=${XXX}$(HT)PersonalVS=${XXX}
```

The client uploads the execution result:

```
ID=${CmdID}&return=${XXX}&CMD=DATA
```

Annotation

Parameter	Description
extuser	The table name, which is the extended user
Pin	User ID
FunSwitch	It is the function switch. Each bit controls a specific function. For example: 00000000 00000000 00000000 10001000 The bits have the following meanings from low to high: 1. The switch for the driving duration of the extended lock. 2. The switch for the temporary user (visitor) i.e., • FunSwitch=1 (00000001) means enabling the function of driving duration for the extended lock. • FunSwitch=2 (00000010) means enabling the temporary user function. • FunSwitch=3 (00000011) means enabling the above two functions simultaneously.
FirstName	Reserved, not used yet
LastName	Reserved, not used yet
PersonalVS	The personal verification mode
${LF} is used to connect multiple records.	

Example

The server delivers:

```
C:500:DATA UPDATE extuser Pin=1 FunSwitch=1 FirstName= LastName=
```

```
PersonalVS=0
```

The client uploads the successful execution result:

```
ID=500&Return=0&CMD=DATA
```

MulCardUser Information

Application scenario

The server delivers MulCardUser information to the client. It is generally used to synchronize the MulCardUser information edited on the server to the client.

The command format is as follows:

The server delivers the command:

```
C:${CmdID}:DATA$(SP)UPDATE$(SP)mulcarduser$(SP)Pin=${XXX}$(HT)CardNo=${XXX}$(HT)LossCardFlag=${XXX}$(HT)CardType=${XXX}}$(HT)CardBit=${XXX}}$(HT)SiteCode=${XXX}}$(HT)IMIE=${XXX}
```

The client uploads the execution result:

```
ID=${CmdID}&return=${XXX}&CMD=DATA
```

Annotation

Parameter	Description
mulcarduser	The table name, which is the mulcarduser.
Pin	User ID
CardNo	The card number. It is stored as a string. The value delivered by the software is a hexadecimal string. For example, CardNo=DF349C3BEA5277ED.
LossCardFlag	The card loss flag. 0 indicates that the card is in normal use, and 1 indicates that the card is lost.
CardType	The main card or sub-card. 0 indicates the main card, and 1 indicates a sub-card.
CardBit	Wiegand bit number (only supported in WGRuleVer=V2.0 and above versions)
SiteCode	Site code (only supported in WGRuleVer=V2.0 and above versions)
IMIE	IMIE identification code (only supported in WGRuleVer=V2.0 and above

	versions)
${LF} is used to connect multiple records.	

Example

The server delivers:

```
(1) WGRuleVer=V1.0
    C:501:DATA UPDATE mulcarduser Pin=1 CardNo=CC9932DD LossCardFlag=0
    CardType=0
(2) WGRuleVer=V2.0
    C:501:DATA UPDATE mulcarduser Pin=1 CardNo=CC9932DD LossCardFlag=0
    CardType=0 CardBit=26 SiteCode=10010 IMIE=356591106
```

The client uploads the successful execution result:

```
ID=501&return=0&CMD=DATA
```

Identity Card Information

Application scenario

The server delivers the identity card information to the client. It is generally used to synchronize the identity card information edited on the server to the client.

The command format is as follows:

The server delivers the command:

```
C:${CmdID}:DATA$(SP)UPDATE$(SP)identitycard$(SP)Pin=${XXX}$(HT)SN_Num=${XXX}$(HT)ID_Num=${XXX}$(HT)DN_Num=${XXX}$(HT)Name=${XXX}$(HT)Gender=${XXX}$(HT)Nation=${XXX}$(HT)Birthday=${XXX}$(HT)Valid_Info=${XXX}$(HT)Address=${XXX}$(HT)Additional_Info=${XXX}$(HT)Issuer=${XXX}$(HT)Photo=${XXX}$(HT)PhotoJPG=${XXX}$(HT)FP_Template1=${XXX}$(HT)FP_Template2=${XXX}$(HT)Reserve=${XXX}$(HT)Notice=${XXX}
```

The client uploads the execution result:

```
ID=${CmdID}&return=${XXX}&CMD=DATA
```

Annotation

Parameter	Description
identitycard	The table name, which is the identity card information
Pin	User ID
SN_Num	The physical number of the identity card
ID_Num	The number of the citizen identity card
DN_Num	The SN of the resident identity card (the card management number).
Name	The name of the user
Gender	The gender code

Nation

The nationality code:

Nationality Code	Nation	Nationality Code	Nation
0	Decoding Error	30	Tu
1	Han	31	Daur
2	Mongol	32	Mulao
3	Hui	33	Qiang
4	Tibetan	34	Blang
5	Uyghur	35	Salar
6	Miao	36	Maonan
7	Yi	37	Kelao
8	Zhuang	38	Xibe
9	Buyi	39	Achang
10	Korean	40	Pumi
11	Manchu	41	Tajik
12	Dong	42	Nu
13	Yao	43	Ozbek
14	Bai	44	Russian
15	Tujia	45	Ewenki
16	Hani	46	De'ang
17	Kazak	47	Baoan
18	Dai	48	Yugu
19	Li	49	Jing
20	Lisu	50	Tatar
21	Va	51	Drung
22	She	52	Oroqen
23	Gaoshan	53	Hezhe
24	Lahu	54	Monba

	25 Shui 55 Lhoba 26 Dongxiang 56 Jino 27 Naxi 57 Encoding Error 28 Jingpo 97 Other 29 Kirgiz 98 Foreign Origin
Birthday	The date of birth
Valid_Info	Validity period
Address	The address of the User
Additional_Info	The additional address read by the machine
Issuer	The issuing authority
Photo	The photo data is encrypted and converted to the base64-based data before transmission.
PhotoJPG	The photo data is decrypted and converted to the base64-based data before transmission.
FP_Template1	Features of the finger 1.
FP_Template2	Features of the finger 2.
Reserve	A reserved field
Notice	The remarks

Example

The server delivers:

```
C:502:DATA UPDATE identitycard Pin=1    SN_Num=26121292460088826975
ID_Num=210102199212182xxxx   DN_Num=  Name=Zhang San Gender=1    Nation=1
Birthday=19911218   ValidInfo=2013091220230912  Address=xxxxx, Tangxia
Town, Dongguan, Guangdong  AdditionalInfo= Issuer=xxx Public Security Bureau
Photo=V0xmAH4AMgAA/apmyEd8
PhotoJPG=Y0tKqWNPzyEd8Pn0EhRsgAAjeWPxiUzLaPU1w=
FPTemplate1=QwGIEgELUQAAAAAAAAAAAAAACgBmnIcAf///////0wZqfwuJK78PzJg/lw
FPTemplate2=QwGIEgEQUAAAAAAAAAAAAAAADIBmkbyAP///////3UYCPxXQs38qEVW/rpLo
vwyUBv8WV8 Reserve=    Notice=
```

The client uploads the successful execution result:

```
ID=502&return=0&CMD=DATA
```

User's Access Control Privilege

Application scenario

The server delivers the user's access control privilege to the client. Generally, the privilege is delivered together with the user information.

The command format is as follows:

The server delivers the command:

```
C:${CmdID}:DATA$(SP)UPDATE$(SP)userauthorize$(SP)Pin=${XXX}$(HT)Authorize
TimezoneId=${XXX}$(HT)AuthorizeDoorId=${XXX}$(HT)DevID=${XXX}$(HT)StartTi
me=${XXX}$(HT)EndTime=${XXX}
```

The client uploads the execution result:

```
ID=${CmdID}&return=${XXX}&CMD=DATA
```

Annotation

<table>
<tr><th>Parameter</th><th>Description</th></tr>
<tr><td>userauthorize</td><td>The table name, which is the user's access control privilege</td></tr>
<tr><td>AuthorizeTimezoneId</td><td>The ID of the user's time rule</td></tr>
<tr><td>AuthorizeDoorId</td><td>The ID of the user's door
It indicates the doors of the device to which the privilege can be applied. The value is encoded in binary. Each door is represented by a binary bit. If the bit value is 1, the user has the privilege of this door. For details, see the following:
<table>
<tr><th>AuthorizeDoorId</th><th>Indication</th></tr>
<tr><td>1</td><td>LOCK1</td></tr>
<tr><td>2</td><td>LOCK2</td></tr>
<tr><td>3</td><td>LOCK1 and LOCK2</td></tr>
<tr><td>4</td><td>LOCK3</td></tr>
</table>
</td></tr>
</table>

	5 LOCK1 and LOCK3 6 LOCK2 and LOCK3 7 LOCK1, LOCK2, and LOCK3 8 LOCK4 9 LOCK1 and LOCK4 10 LOCK2 and LOCK4 11 LOCK1, LOCK2, and LOCK4 12 LOCK3 and LOCK4 13 LOCK1, LOCK3, and LOCK4 14 LOCK2, LOCK3, and LOCK4 15 LOCK1, LOCK2, LOCK3, and LOCK4 If the four doors are numbered as 1, 2, 3, and 4 respectively, the value 15 is obtained according to the following calculation formula: **1<<(1-1)+1<<(2-1)+1<<(3-1)+1<<(4-1)=15**
DevID	The device ID. The value of AccSupportFunList counts from 0, and the value of the 26th bit is used to control the function.
${LF} is used to connect multiple records.	
StartTime	The start time of user's validity period. Example: starttime=752688180, calculated by the algorithm in Appendix 5.
EndTime	The end time of user's validity period. Example: endtime=755193780, calculated by the algorithm in Appendix 5.

Example

The server delivers the information of a user of which user ID is 1, the time rule ID is 1, and the door ID is 1:

```
C:296:DATA UPDATE userauthorize Pin=1   AuthorizeTimezoneId=1
AuthorizeDoorId=1   DevID=1
```

The client uploads the successful execution result:

```
ID=296&return=0&CMD=DATA
```

Holiday

Application scenario

The server delivers the holiday data to the client. It is generally used to synchronize the holiday data edited on the server to the client.

The command format is as follows:

The server delivers the command:

```
C:${CmdID}:DATA$(SP)UPDATE$(SP)holiday$(SP)Holiday=${XXX}$(HT)HolidayType
=${XXX}$(HT)Loop=${XXX}
```

The client uploads the execution result:

```
ID=${CmdID}&return=${XXX}&CMD=DATA
```

Annotation

Parameter	Description
holiday	The table name, which is the holiday
Holiday	E.g., 20100101 means January 1, 2010.
HolidayType	The holiday type. Values are 1, 2, and 3, of which the meaning should be defined as needed.
Loop	1 indicates that the year is different but the month and the day are the same. 2 indicates that all the year, month, and day must be the same.
${LF} is used to connect multiple records.	

Example

The server delivers:

```
C:503:DATA UPDATE holiday Holiday=20100101  HolidayType=1   Loop=1
```

The client uploads the successful execution result:

```
ID=503&return=0&CMD=DATA
```

Time Rule

Application scenario

The server delivers a time rule to the client. Generally, it is used to automatically deliver the time rule edited on the server.

The command format is as follows:

The server delivers the command:

```
C:${CmdID}:DATA$(SP)UPDATE$(SP)timezone$(SP)TimezoneId=${XXX}$(HT)SunTime1=${XXX}$(HT)SunTime2=${XXX}$(HT)SunTime3=${XXX}$(HT)MonTime1=${XXX}$(HT)MonTime2=${XXX}$(HT)MonTime3=${XXX}$(HT)TueTime1=${XXX}$(HT)TueTime2=${XXX}$(HT)TueTime3=${XXX}$(HT)WedTime1=${XXX}$(HT)WedTime2=${XXX}$(HT)WedTime3=${XXX}$(HT)ThuTime1=${XXX}$(HT)ThuTime2=${XXX}$(HT)ThuTime3=${XXX}$(HT)FriTime1=${XXX}$(HT)FriTime2=${XXX}$(HT)FriTime3=${XXX}$(HT)SatTime1=${XXX}$(HT)SatTime2=${XXX}$(HT)SatTime3=${XXX}$(HT)Hol1Time1=${XXX}$(HT)Hol1Time2=${XXX}$(HT)Hol1Time3=${XXX}$(HT)Hol2Time1=${XXX}$(HT)Hol2Time2=${XXX}$(HT)Hol2Time3=${XXX}$(HT)Hol3Time1=${XXX}$(HT)Hol3Time2=${XXX}$(HT)Hol3Time3=${XXX}$(HT)StartTime=${XXX}$(HT)EndTime=${XXX}
```

The client uploads the execution result:

```
ID=${CmdID}&Return=${XXX}&CMD=DATA
```

Annotation

Parameter	Description
`timezone`	The table name, which is time group
`TimezoneId`	The ID of the time rule **SunTime1 - SatTime3:** From Sunday to Saturday, three time periods each day. **Hol1Time1 - Hol3Time3:** Three holidays, three time periods each holiday.
Note: Each time period lasts from StartTime to EndTime on the software. Before the time rule is delivered to the device, the software converts the time rule to an integer based on the convention StartTime<<16+EndTime. For example, the converted value of 8:30-12:00 is 830 << 16 + 1200, that is, 0x33e04b0.	

StartTime	The effective start time of the time period, calculated by the [Appendix 5] ZKTeco algorithm.
EndTime	The effective end time of the time period, calculated by the [Appendix 5] ZKTeco algorithm.

Example

The server delivers a time rule, in which the time rule ID is 2, the time period 1 on Monday is 00:01-00:03, and the time period 2 is 10:00-11:00:

```
C:307:DATA UPDATE timezone TimezoneId=2 SunTime1=0 SunTime2=0 SunTime3=0
MonTime1=65539 MonTime2=65537100 MonTime3=0 TueTime1=0 TueTime2=0
TueTime3=0 WedTime1=0 WedTime2=0 WedTime3=0 ThuTime1=0 ThuTime2=0
ThuTime3=0 FriTime1=0 FriTime2=0 FriTime3=0 SatTime1=0 SatTime2=0
SatTime3=0 Hol1Time1=0 Hol1Time2=0 Hol1Time3=0 Hol2Time1=0 Hol2Time2=0
Hol2Time3=0 Hol3Time1=0 Hol3Time2=0 Hol3Time3=0
```

The client uploads the successful execution result:

```
ID=307&return=0&CMD=DATA
```

Fingerprint Template

Application scenario

The server delivers a fingerprint template, if any, to the client when delivering the user information.

The command format is as follows:

The server delivers the command:

```
C:${CmdID}:DATA$(SP)UPDATE$(SP)templatev10
Pin=${XXX}$(HT)FingerID=$(HT)Valid=${XXX}$(HT)Template=${XXX}
```

The client uploads the execution result:

```
ID=${CmdID}&return=${XXX}&CMD=DATA
```

Annotation

Parameter	Description
templatev10	Indicate that the data type is fingerprint template. The supported fingerprint

<table>
<tr><td></td><td>algorithm version is lower than 10.0.</td></tr>
<tr><td>Pin</td><td>User ID</td></tr>
<tr><td>FingerID</td><td>The finger number. The value ranges from 0 to 9.</td></tr>
<tr><td>Valid</td><td>Describe whether the template is valid and is duress. The value meanings are described below:<table><tr><th>Value</th><th>Description</th></tr><tr><td>0</td><td>Invalid</td></tr><tr><td>1</td><td>Valid</td></tr><tr><td>3</td><td>Duress</td></tr></table></td></tr>
<tr><td>Template</td><td>Before a fingerprint template is transmitted, the original binary fingerprint template must be base64-encoded.</td></tr>
<tr><td colspan="2">${LF} is used to connect multiple records.</td></tr>
</table>

Example

The server delivers a common fingerprint of which the user ID is 1 and the fingerprint ID is 6:

```
C:333:DATA UPDATE templatev10 Pin=1 FingerID=6  Valid=1
Template=Si9TUzIxAAADbG8ECAUHCc7QAAAbbWkBAAAAg5Eac2wNAG4PmwDZAAVjrQAiAIIP
WwAzbAIPsgCOAMgOWmyOAFgPwwBWAI5i2wCUAJsPTACUbF0PrwCqAKIPgGy8AFAPwwAHAINjZ
ADMANQPPADdbKgPjgDhAIAOoGzmADAOlQApADJiqwD5ALIOYwAEbasM/wAJAWcPn2wVAYoNhA
DfAfJhbQAlAdYPCgAkbZ0PbAA2ASAP0Gw4AZQPI2jqbyCDiIW1BQuYSH1i7Y4XXYE5/d6Xgu2
4q3p2M/ymh0hngIH19lIorBJt/lI0bg4eU//33jP73D7onn2jDvmapxjD9F9QX4XA/ManEgtv
K04TZ3tIFlqVxQBEgd6C+e9BAGWC3AiC6WSCofgxIEd4mZdKCj8T0eComsm0hGr9H+4NKPCai
55NHQmD9bIWXnx79YNfpfkK94RjfiCZAiAvxAIRcMoEAHYBdLkEA88FhpMEAKvNAzBmAaQLgI
hitQQDAhFtXAYAn9z9/icPANEZjMAEmcOuZ3sKAJcdsn53AQQAsCIJ/48LA8QkfcLAwmavBwP
OLf3+/cFGzQCZXHuDwHIHAGcyAJP/UxMA91pbw8MUw//EaVzCOsMQbPpmmmnBwUHCwq3Bb2wV
AQe4oHfohcT/eMHBtREDMYrgRjExK44QA8KPfcPF/8IEwPyvwcBg/xcBwoykG32MwYNneK8HA
zmRXsJrwRTF2JPwkMLEwv/CQcDBk8HB/sEIAJ+TVAtZBwDHlxz9VQZshplgfhkBwpyorsHBfp
PCg6pna6sKAKqqcMUGwVVQCwCCwFDABFtPYAG/wJPJxwfDwqjAw0cJAKsD/fiXWFQZAQe+dId
q+sPDgMDCwLvAWmUBr8sQInYHCwPE0UPF/8N2BXsJbKzQHP3AwjrBaGUBrtktfMCuGwJj26lS
wUWXB8LArXZvZFkYADDcpAjBw8LCw8UGwv2uecH+wEQJxY3lKsD9KXYPAFHnw0b99/z//sEFw
FBwAAvpsG3CBcDBrKHDw3rAizrAWJMGAJLvPcUAVRl9DQinZMDABcDBrsPExsOJUDv/wV4EEK
AXF8GOBBO/Khf/xAkQqjLhkvz7/sBmBdWTS+zCVwgQaUoi//+S/0QNEONMSlfCrKDBaVJCAM5
```

```
DAmwBC0VS
```

The client uploads the successful execution result:

```
ID=333&return=0&CMD=DATA
```

First-card Opening Information

Application scenario

The server delivers the first-card opening data to the client. It is generally used to synchronize the first-card opening data edited on the server to the client.

The command format is as follows:

The server delivers the command:

```
C:${CmdID}:DATA$(SP)UPDATE$(SP)firstcard$(SP)DoorID=${XXX}$(HT)TimezoneID
=${XXX}$(HT)Pin=${XXX}$(HT)DevID=${XXX}
```

The client uploads the execution result:

```
ID=${CmdID}&return=${XXX}&CMD=DATA
```

Annotation

Parameter	Description
firstcard	The table name, which is the first-card opening table.
Pin	User ID
DoorID	Door ID
TimezoneID	Time zone ID
DevID	The device ID. The value of AccSupportFunList counts from 0, and the value of the 26th bit is used to control the function.
${LF} is used to connect multiple records.	

Example

The server delivers:

```
C:504:DATA UPDATE firstcard DoorID=1    TimezoneID=1    Pin=1   DevID=1
```

The client uploads the successful execution result:

```
ID=504&return=0&CMD=DATA
```

Multi-card Opening Information

Application scenario

The server delivers the multi-card opening data to the client. It is generally used to synchronize the multi-card opening data edited on the server to the client. (This command is valid only when the DoorMultiCardOpenDoor=1)

The command format is as follows:

The server delivers the command:

```
C:${CmdID}:DATA$(SP)UPDATE$(SP)multimcard$(SP)Index=${XXX}$(HT)DoorId=${XXX}$(HT)Group1=${XXX}$(HT)Group2=${XXX}$(HT)Group3=${XXX}$(HT)Group4=${XXX}$(HT)Group5=${XXX}$(HT)DevID=${XXX}
```

The client uploads the execution result:

```
ID=${CmdID}&return=${XXX}&CMD=DATA
```

Annotation

Parameter	Description
multimcard	The table name, which is the multi-card opening table
Index	Index
DoorId	Door ID
Group1	Group1
Group2	Group 2
Group3	Group 3
Group4	Group 4
Group5	Group 5
DevID	The device ID. The value of AccSupportFunList counts from 0, and the value of the 26th bit is used to control the function.

${LF} is used to connect multiple records.

Example

The server delivers:

```
C:505:DATA UPDATE multimcard Index=1    DoorId=1    Group1=1    Group2=3
Group3= Group4= Group5= DevID=1
```

The client uploads the successful execution result:

```
ID=505&return=0&CMD=DATA
```

Linkage Details

Application scenario

The server delivers the linkage details to the client. It is generally used to synchronize the linkage details edited on the server to the client.

The command format is as follows:

The server delivers the command:

```
C:${CmdID}:DATA$(SP)UPDATE$(SP)inoutfun$(SP)Index=${XXX}$(HT)EventType=${
XXX}$(HT)InAddr=${XXX}$(HT)OutType=${XXX}$(HT)OutAddr=${XXX}$(HT)OutTime=
${XXX}$(HT)Reserved=${XXX}$(HT)InDevID=${XXX}$(HT)OutDevID=${XXX}
```

The client uploads the execution result:

```
ID=${CmdID}&return=${XXX}&CMD=DATA
```

Annotation

Parameter	Description
inoutfun	The table name, which is the linkage details table.
Index	Index
EventType	Triggered event
InAddr	The position where the event is triggered, which is the door ID or the auxiliary input ID. The value ranges from 1 to 4. 0 means any position

OutType	The output type. 0 means lock and 1 means auxiliary output.
OutAddr	The position of the output action. The value of the lock output ranges from 1 to 4. The value of auxiliary output ranges from 1 to 4.
OutTime	The time of the output action. 0 means closed. 1 to 254 means the lock is opened for n seconds, and 255 means normal open.
Reserved	Reserved for alignment. The low four bits indicate the linkage type: 0 Door linkage 1 Reader linkage 2 IPC linkage The high four bits indicate the action type: 16 Lock 32 Unlock
InDevID	The device ID. The value of AccSupportFunList counts from 0, and the value of the 26th bit is used to control the function.
OutDevID	The device ID. The value of AccSupportFunList counts from 0, and the value of the 26th bit is used to control the function.
${LF} is used to connect multiple records.	

Example

The server delivers:

```
C:506:DATA UPDATE inoutfun Index=1  EventType=204  InAddr=0  OutType=0
OutAddr=1  OutTime=0  Reserved=0  InDevID=14  OutDevID=14
```

The client uploads the successful execution result:

```
ID=506&return=0&CMD=DATA
```

Scheduled Output Information

Application scenario

The server delivers the scheduled output information to the client. It is generally used to synchronize

the scheduled output information edited on the server to the client.

The command format is as follows:

The server delivers the command:

```
C:${CmdID}:DATA$(SP)UPDATE$(SP)outrelaysetting$(SP)Num=${XXX}$(HT)OutType
=${XXX}$(HT)ActionType=${XXX}$(HT)TimezoneId=${XXX}$(HT)DevID=${XXX}
```

The client uploads the execution result:

```
ID=${CmdID}&return=${XXX}&CMD=DATA
```

Annotation

Parameter	Description
outrelaysetting	The table name, which is the scheduled output table.
Num	The output point or the number of the primary key of the auxiliary output property table.
OutType	0 indicates door and 1 indicates auxiliary output.
ActionType	0 indicates none, 2 indicates normally closed, and 1 indicates normal open.
TimezoneId	The ID of the time zone.
DevID	The device ID. The value of AccSupportFunList counts from 0, and the value of the 26th bit is used to control the function.
${LF} is used to connect multiple records.	

Example

The server delivers:

```
C:507:DATA UPDATE outrelaysetting Num=1 OutType=1 ActionType=0
TimezoneId=1 DevID=14
```

The client uploads the successful execution result:

```
ID=507&return=0&CMD=DATA
```

DLST Information

Application scenario

The server delivers the DLST information to the client. It is generally used to synchronize the DLST information edited on the server to the client.

The command format is as follows:

The server delivers the command:

```
C:${CmdID}:DATA$(SP)UPDATE$(SP)DSTSetting$(SP)Year=${XXX}$(HT)StartTime=${XXX}$(HT)EndTime=${XXX}$(HT)Loop=${XXX}
```

The client uploads the execution result:

```
ID=${CmdID}&return=${XXX}&CMD=DATA
```

Annotation

Parameter	Description
DSTSetting	The table name, which is the DLST table.
Year	The year.
StartTime	The start time, in the format of MMIIWWHH, wherein MM indicates the month, II indicates the week, WW indicates the weekday, and HH indicates the hour.
EndTime	The end time, in the format of MMIIWWHH, wherein MM indicates the month, II indicates the week, WW indicates the weekday, and HH indicates the hour.
Loop	Whether it is a loop.
${LF} is used to connect multiple records.	

Example

The server delivers:

```
C:508:DATA UPDATE DSTSetting Year=2018 StartTime=03020517 EndTime=08020508 Loop=1
```

The client uploads the successful execution result:

```
ID=508&return=0&CMD=DATA
```

Device Property

Application scenario

The server delivers the device properties to the client. It is generally used to synchronize the device properties edited on the server to the client.

The command format is as follows:

The server delivers the command:

```
C:${CmdID}:DATA$(SP)UPDATE$(SP)DevProperty$(SP)ID=${XXX}$(HT)Type=${XXX}$(HT)BusType=${XXX}$(HT)MachineType=${XXX}$(HT)Address=${XXX}$(HT)Mac=${XXX}$(HT)IPAddress=${XXX}$(HT)SN=${XXX}$(HT)IsMaster=${XXX}
```

The client uploads the execution result:

```
ID=${CmdID}&return=${XXX}&CMD=DATA
```

Annotation

Parameter	Description
DevProperty	The table name, which is the device property table.
ID	Device ID.
Type	The device type. 0: Door Unit, 1: Zigbee, and 2: BioCom48. Currently, all is 0.
BusType	The bus type. 0: 485 A, 1: 485B, 2: ZIGBEE, and 3: TCP/IP.
MachineType	The machine type. (see Appendix 7).
Address	Device address
Mac	Device MAC address
IPAddress	Device IP address
SN	The device SN
IsMaster	Whether it is a master. 0: Master and 1: Slave.
${LF} is used to connect multiple records.	

Example

The server delivers:

```
C:509:DATA UPDATE DevProperty ID=14 Type=0 BusType=0 MachineType=28
Address=0 Mac= IPAddress=192.168.223.222 SN=DDG7050067042500078
IsMaster=0
```

The client uploads the successful execution result:

```
ID=509&return=0&CMD=DATA
```

Device Parameter

Application scenario

The server delivers the device parameters to the client. It is generally used to synchronize the device parameters edited on the server to the client.

The command format is as follows:

The server delivers the command:

```
C:${CmdID}:DATA$(SP)UPDATE$(SP)DevParameters$(SP)ID=${XXX}$(HT)Name=${XXX
}$(HT)Value=${XXX}
```

The client uploads the execution result:

```
ID=${CmdID}&return=${XXX}&CMD=DATA
```

Annotation

Parameter	Description
DevParameters	The table name, which is the device parameter table.
ID	Device ID.
Name	The parameter name (see Appendix 12)
Value	The parameter value
${LF} is used to connect multiple records.	

Example

The server delivers:

```
C:510:DATA UPDATE DevParameters ID=14   Name=IsSupportReaderEncrypt
Value=0
```

The client uploads the successful execution result:

```
ID=510&return=0&CMD=DATA
```

Door Property

Application scenario

The server delivers the door properties to the client. It is generally used to synchronize the door properties edited on the server to the client.

The command format is as follows:

The server delivers the command:

```
C:${CmdID}:DATA$(SP)UPDATE$(SP)DoorProperty$(SP)ID=${XXX}$(HT)DevID=${XXX
}$(HT)Address=${XXX}$(HT)Disable=${XXX}
```

The client uploads the execution result:

```
ID=${CmdID}&return=${XXX}&CMD=DATA
```

Annotation

Parameter	Description
DoorProperty	The table name, which is the door property table.
ID	Door ID
DevID	Slave ID
Address	The lock relay. 1 LOCK1 2 LOCK2 3 LOCK3 4 LOCK4

Disable	To Enable or disable. 0 Enable 1 Disable
${LF} is used to connect multiple records.	

Example

The server delivers:

```
C:511:DATA UPDATE DoorProperty ID=1 DevID=14    Address=1   Disable=0
```

The client uploads the successful execution result:

```
ID=511&return=0&CMD=DATA
```

Door Parameter

Application scenario

The server delivers the door parameters to the client. It is generally used to synchronize the door parameters edited on the server to the client.

The command format is as follows:

The server delivers the command:

```
C:${CmdID}:DATA$(SP)UPDATE$(SP)DoorParameters$(SP)ID=${XXX}$(HT)Name=${XXX}$(HT)Value=${XXX}$(HT)DevID=${XXX}
```

The client uploads the execution result:

```
ID=${CmdID}&return=${XXX}&CMD=DATA
```

Annotation

Parameter	Description
DoorParameters	The table name, which is the door parameter table.
ID	Door ID.
Name	The parameter name (see Appendix 10)
Value	The parameter value

DevID	The ID of the device property
${LF} is used to connect multiple records.	

Example

The server delivers:

```
C:512:DATA UPDATE DoorParameters ID=1   Name=MaskFlag   Value=0 DevID=14
```

The client uploads the successful execution result:

```
ID=512&return=0&CMD=DATA
```

Reader Property

Application scenario

The server delivers the reader properties to the client. It is generally used to synchronize the reader properties edited on the server to the client.

The command format is as follows:

The server delivers the command:

```
C:${CmdID}:DATA$(SP)UPDATE$(SP)ReaderProperty$(SP)ID=${XXX}$(HT)DoorID=${XXX}$(HT)Type=${XXX}$(HT)Address=${XXX}$(HT)IPAddress=${XXX}$(HT)Port=${XXX}$(HT)MAC=${XXX}InOut=${XXX}$(HT)Disable=${XXX}$(HT)VerifyType=${XXX}$(HT)Multicast=${XXX}$(HT)DevID=${XXX}$(HT)OfflineRefuse=${XXX}
```

The client uploads the execution result:

```
ID=${CmdID}&return=${XXX}&CMD=DATA
```

Annotation

Parameter	Description
ReaderProperty	The table name, which is the device parameter table.
ID	Reader ID.
DoorID	Door ID

Type	The reader type, which is calculated by binary bits. 0 Disable 1 485 Reader 2 Wiegand Reader 3 485 Reader or Wiegand Reader 4 Network Reader 8 Zigbee Reader
Address	The reader address, applicable for the Wiegand reader and 485 reader.
IPAddress	Reader IP address.
Port	Reader port
MAC	Reader MAC address
InOut	The input or output reader: 0 Input 1 Output
Disable	To Enable or disable: 0 Enable 1 Disable
VerifyType	Reader verification mode
Multicast	Multicast address
DevID	Device ID
OfflineRefuse	Whether offline access is supported. 0 Accept 1 Refuse
${LF} is used to connect multiple records.	

Example

The server delivers:

```
C:513:DATA UPDATE ReaderProperty ID=1   DoorID=1    Type=2  Address=1
IPAddress=  Port=4376   MAC=    InOut=0 Disable=0   VerifyType=0
```

```
Multicast=  DevID=14    OfflineRefuse=0
```

The client uploads the successful execution result:

```
ID=513&return=0&CMD=DATA
```

Reader Parameter

Application scenario

The server delivers the reader parameters to the client. It is generally used to synchronize the reader parameters edited on the server to the client.

The command format is as follows:

The server delivers the command:

```
C:${CmdID}:DATA$(SP)UPDATE$(SP)ReaderParameters$(SP)ID=${XXX}$(HT)DevID=$
{XXX}$(HT)Name=${XXX}$(HT)Value=${XXX}
```

The client uploads the execution result:

```
ID=${CmdID}&return=${XXX}&CMD=DATA
```

Annotation

Parameter	Description
ReaderParameters	The table name, which is the reader parameter table
ID	Reader ID.
DevID	Device ID
Name	The parameter name (see Appendix 11)
Value	The parameter value
${LF} is used to connect multiple records.	

Example

The server delivers:

```
C:514:DATA UPDATE ReaderParameters ID=1 DevID=1 Name=MaskFlag  Value=1
```

The client uploads the successful execution result:

```
ID=514&return=0&CMD=DATA
```

Auxiliary Input Property

Application scenario

The server delivers the auxiliary input properties to the client. It is generally used to synchronize the auxiliary input properties edited on the server to the client.

The command format is as follows:

The server delivers the command:

```
C:${CmdID}:DATA$(SP)UPDATE$(SP)AuxIn$(SP)ID=${XXX}$(HT)DevID=${XXX}$(HT)Address=${XXX}$(HT)Disable=${XXX}
```

The client uploads the execution result:

```
ID=${CmdID}&return=${XXX}&CMD=DATA
```

Annotation

Parameter	Description
AuxIn	The table name, which is the auxiliary input property table.
ID	Auxiliary input ID.
DevID	Device ID
Address	Auxiliary input: 1 AuxIn1 2 AuxIn2
Disable	Enable or disable: 0 Enable 1 Disable
${LF} is used to connect multiple records.	

Example

The server delivers:

```
C:515:DATA UPDATE AuxIn ID=1    DevID=14    Address=1   Disable=0
```

The client uploads the successful execution result:

```
ID=515&return=0&CMD=DATA
```

Auxiliary Output Property

Application scenario

The server delivers the auxiliary output properties to the client. It is generally used to synchronize the auxiliary output properties edited on the server to the client.

The command format is as follows:

The server delivers the command:

```
C:${CmdID}:DATA$(SP)UPDATE$(SP)AuxOut$(SP)ID=${XXX}$(HT)DevID=${XXX}$(HT)
Address=${XXX}$(HT)Disable=${XXX}
```

The client uploads the execution result:

```
ID=${CmdID}&return=${XXX}&CMD=DATA
```

Annotation

Parameter	Description
AuxOut	The table name, which is the auxiliary output property table
ID	Reader ID.
DevID	Device ID
Address	The auxiliary output relay: 1 AuxOut1
Disable	Enable or disable: 0 Enable 1 Disable
${LF} is used to connect multiple records.	

Example

The server delivers:

```
C:516:DATA UPDATE AuxOut ID=1   DevID=14    Address=1   Disable=0
```

The client uploads the successful execution result:

```
ID=516&return=0&CMD=DATA
```

Default Wiegand Format

Application scenario

The server delivers the default Wiegand format to the client. It is generally used to synchronize the default Wiegand format edited on the server to the client.

The command format is as follows:

The server delivers the command:

```
C:${CmdID}:DATA$(SP)UPDATE$(SP)DefaultWGFormat$(SP)ID=${XXX}$(HT)CardBit=
${XXX}$(HT)SiteCode=${XXX}$(HT)FormatName=${XXX}$(HT)CardFormat=${XXX}
```

The client uploads the execution result:

```
ID=${CmdID}&return=${XXX}&CMD=DATA
```

Annotation

Parameter	Description
DefaultWGFormat	The table name, which is the default Wiegand format table
ID	The index.
CardBit	The number of bits
SiteCode	Site code
FormatName	The user-defined Wiegand name
CardFormat	Wiegand format
${LF} is used to connect multiple records.	

Example

The server delivers:

```
C:517:DATA UPDATE DefaultWGFormat ID=2  CardBit=26  SiteCode=0
```

```
FormatName=Wiegand 26
CardFormat=pccccccccccccccccccccccccp:eeeeeeeeeeeeeeooooooooooooo
```

The client uploads the successful execution result:

```
ID=517&return=0&CMD=DATA
```

Wiegand Format

Application scenario

The server delivers the Wiegand format to the client. It is generally used to synchronize the Wiegand format edited on the server to the client.

The command format is as follows:

The server delivers the command:

```
C:${CmdID}:DATA$(SP)UPDATE$(SP)WGFormat$(SP)ID=${XXX}$(HT)CardBit=${XXX}$
(HT)SiteCode=${XXX}$(HT)FormatName=${XXX}$(HT)CardFormat=${XXX}
```

The client uploads the execution result:

```
ID=${CmdID}&return=${XXX}&CMD=DATA
```

Annotation

Parameter	Description
WGFormat	The table name, which is the default Wiegand format table
ID	The index.
CardBit	The number of bits
SiteCode	Site code
FormatName	The user-defined Wiegand name
CardFormat	Wiegand format
${LF} is used to connect multiple records.	

Example

The server delivers:

```
C:518:DATA UPDATE WGFormat ID=2 CardBit=26  SiteCode=0  FormatName=Wiegand
26 CardFormat=pccccccccccccccccccccccp:eeeeeeeeeeeeeooooooooooooo
```

The client uploads the successful execution result:

```
ID=518&return=0&CMD=DATA
```

Wiegand Format of Wiegand Reader

Application scenario

The server delivers the Wiegand format of the Wiegand reader to the client. It is generally used to synchronize the Wiegand format of the Wiegand reader edited on the server to the client.

The command format is as follows:

The server delivers the command:

```
C:${CmdID}:DATA$(SP)UPDATE$(SP)ReaderWGFormat$(SP)ReaderID=${XXX}$(HT)Dev
ID=${XXX}$(HT)WGFormatID=${XXX}$(HT)ParityVerifyDisable=${XXX}$(HT)Revers
alType=${XXX}
```

The client uploads the execution result:

```
ID=${CmdID}&return=${XXX}&CMD=DATA
```

Annotation

Parameter	Description
ReaderWGFormat	The table name, which is the table of the Wiegand format of the Wiegand reader ReaderID: The reader ID.
DevID	The device ID. The value of AccSupportFunList counts from 0, and the value of the 26th bit is used to control the function.
WGFormatID	The index of the Wiegand format table
ParityVerifyDisable	Whether to disable the parity check: 0 Enable 1 Disable
ReversalType	1 Invert all bits

	2	Invert 0 and 1
	3	Exchange low bits with high bits
	4	Exchange a low bit with a high bit
${LF} is used to connect multiple records.		

Example

The server delivers:

```
C:519:DATA UPDATE ReaderWGFormat ReaderID=1 DevID=14   WGFormatID=2
ParityVerifyDisable=0   ReversalType=0
```

The client uploads the successful execution result:

```
ID=519&return=0&CMD=DATA
```

Input Control (by Time Period) Information

Application scenario

The server delivers the input control (by time period) information to the client. It is generally used to synchronize the input control (by time period) information edited on the server to the client.

The command format is as follows:

The server delivers the command:

```
C:${CmdID}:DATA$(SP)UPDATE$(SP)InputIOSetting$(SP)Number=${XXX}$(HT)InTyp
e=${XXX}$(HT)TimeZoneId=${XXX}$(HT)DevID=${XXX}
```

The client uploads the execution result:

```
ID=${CmdID}&return=${XXX}&CMD=DATA
```

Annotation

Parameter	Description
InputIOSetting	The table name, which is the table of input control by time period.
Number	The input ID, which is the door ID in the door property table or the ID in the auxiliary input property table.

InType	0 Exit button 1 Auxiliary input
TimeZoneId	The ID of the time zone
DevID	The device ID. The value of AccSupportFunList counts from 0, and the value of the 26th bit is used to control the function.
${LF} is used to connect multiple records.	

Example

The server delivers:

```
C:520:DATA UPDATE InputIOSetting Number=1   InType=0    TimeZoneId=1
DevID=14
```

The client uploads the successful execution result:

```
ID=520&return=0&CMD=DATA
```

Verification Modes in Different Time Periods

Application scenario

The server delivers the verification modes in different time periods to the client. It is generally used to synchronize the verification modes edited on the server to the client.

The command format is as follows:

The server delivers the command:

```
C:${CmdID}:DATA$(SP)UPDATE$(SP)DiffTimezoneVS$(SP)TimezoneID=${XXX}$(HT)S
unTime1=${XXX}$(HT)SunTime1VSUser=${XXX}$(HT)SunTime1VSDoor=${XXX}$(HT)Su
nTime2=${XXX}$(HT)SunTime2VSUser=${XXX}$(HT)SunTime2VSDoor=${XXX}$(HT)Sun
Time3=${XXX}$(HT)SunTime3VSUser=${XXX}$(HT)SunTime3VSDoor=${XXX}$(HT)MonT
ime1=${XXX}$(HT)MonTime1VSUser=${XXX}$(HT)MonTime1VSDoor=${XXX}$(HT)MonTi
me2=${XXX}$(HT)MonTime2VSUser=${XXX}$(HT)MonTime2VSDoor=${XXX}$(HT)MonTim
e3=${XXX}$(HT)MonTime3VSUser=${XXX}$(HT)MonTime3VSDoor=${XXX}$(HT)TueTime
1=${XXX}$(HT)TueTime1VSUser=${XXX}$(HT)TueTime1VSDoor=${XXX}$(HT)TueTime2
=${XXX}$(HT)TueTime2VSUser=${XXX}$(HT)TueTime2VSDoor=${XXX}$(HT)TueTime3=
${XXX}$(HT)TueTime3VSUser=${XXX}$(HT)TueTime3VSDoor=${XXX}$(HT)WedTime1=$
{XXX}$(HT)WedTime1VSUser=${XXX}$(HT)WedTime1VSDoor=${XXX}$(HT)WedTime2=${
XXX}$(HT)WedTime2VSUser=${XXX}$(HT)WedTime2VSDoor=${XXX}$(HT)WedTime3=${X
```

```
XX}$(HT)WedTime3VSUser=${XXX}$(HT)WedTime3VSDoor=${XXX}$(HT)ThuTime1=${XX
X}$(HT)ThuTime1VSUser=${XXX}$(HT)ThuTime1VSDoor=${XXX}$(HT)ThuTime2=${XXX
}$(HT)ThuTime2VSUser=${XXX}$(HT)ThuTime2VSDoor=${XXX}$(HT)ThuTime3=${XXX}
$(HT)ThuTime3VSUser=${XXX}$(HT)ThuTime3VSDoor=${XXX}$(HT)FriTime1=${XXX}$
(HT)FriTime1VSUser=${XXX}$(HT)FriTime1VSDoor=${XXX}$(HT)FriTime2=${XXX}$(
HT)FriTime2VSUser=${XXX}$(HT)FriTime2VSDoor=${XXX}$(HT)FriTime3=${XXX}$(H
T)FriTime3VSUser=${XXX}$(HT)FriTime3VSDoor=${XXX}$(HT)SatTime1=${XXX}$(HT
)SatTime1VSUser=${XXX}$(HT)SatTime1VSDoor=${XXX}$(HT)SatTime2=${XXX}$(HT)
SatTime2VSUser=${XXX}$(HT)SatTime2VSDoor=${XXX}$(HT)SatTime3=${XXX}$(HT)S
atTime3VSUser=${XXX}$(HT)SatTime3VSDoor=${XXX}$(HT)Hol1Time1=${XXX}$(HT)H
ol1Time1VSUser=${XXX}$(HT)Hol1Time1VSDoor=${XXX}$(HT)Hol1Time2=${XXX}$(HT
)Hol1Time2VSUser=${XXX}$(HT)Hol1Time2VSDoor=${XXX}$(HT)Hol1Time3=${XXX}$(
HT)Hol1Time3VSUser=${XXX}$(HT)Hol1Time3VSDoor=${XXX}$(HT)Hol2Time1=${XXX}
$(HT)Hol2Time1VSUser=${XXX}$(HT)Hol2Time1VSDoor=${XXX}$(HT)Hol2Time2=${XX
X}$(HT)Hol2Time2VSUser=${XXX}$(HT)Hol2Time2VSDoor=${XXX}$(HT)Hol2Time3=${
XXX}$(HT)Hol2Time3VSUser=${XXX}$(HT)Hol2Time3VSDoor=${XXX}$(HT)Hol3Time1=
${XXX}$(HT)Hol3Time1VSUser=${XXX}$(HT)Hol3Time1VSDoor=${XXX}$(HT)Hol3Time
2=${XXX}$(HT)Hol3Time2VSUser=${XXX}$(HT)Hol3Time2VSDoor=${XXX}$(HT)Hol3Ti
me3=${XXX}$(HT)Hol3Time3VSUser=${XXX}$(HT)Hol3Time3VSDoor=${XXX}
```

The client uploads the execution result:

```
ID=${CmdID}&return=${XXX}&CMD=DATA
```

Annotation

Parameter	Description
DiffTimezoneVS	The table name, which is the table of verification modes in different time periods.
TimezoneID	Index ID.
SunTime1-SatTime3	From Sunday to Saturday, three time periods each day.
Hol1Time1-Hol3Time3	Three holidays, three time periods each holiday
SunTime1VSUser-SatTime3VSUser	From Sunday to Saturday, three time periods each day, the verification modes in different time periods when the user is different.
Hol1Time1VSUser-Hol3Time3VSUser	Three holidays, three time periods each holiday, the

	verification modes in different time periods when the user is different
SunTime1VSDoor-SatTime3VSDoor	From Sunday to Saturday, three time periods each day, the verification modes in different time periods when the door is different
Hol1Time1VSDoor-Hol3Time3VSDoor	Three holidays, three time periods each holiday, the verification modes in different time periods when the door is different
Note: Each time period lasts from StartTime to EndTime on the software. Before the time rule is delivered to the device, the software converts the time rule to an integer based on the convention StartTime<<16+EndTime. For example, the converted value of 8:30-12:00 is 830 << 16 + 1200, that is, 0x33e04b0.	
${LF} is used to connect multiple records.	

Example

The server delivers:

```
C:521:DATA UPDATE DiffTimezoneVS TimezoneID=1   SunTime1=2359
SunTime1VSUser=255  SunTime1VSDoor=255  SunTime2=0  SunTime2VSUser=255
SunTime2VSDoor=255  SunTime3=0  SunTime3VSUser=255  SunTime3VSDoor=255
MonTime1=2359   MonTime1VSUser=7    MonTime1VSDoor=5    MonTime2=0
MonTime2VSUser=255  MonTime2VSDoor=255  MonTime3=0  MonTime3VSUser=255
MonTime3VSDoor=255  TueTime1=2359   TueTime1VSUser=255
TueTime1VSDoor=255  TueTime2=0  TueTime2VSUser=255  TueTime2VSDoor=255
TueTime3=0  TueTime3VSUser=255  TueTime3VSDoor=255  WedTime1=2359
WedTime1VSUser=255  WedTime1VSDoor=255  WedTime2=0  WedTime2VSUser=255
WedTime2VSDoor=255  WedTime3=0  WedTime3VSUser=255  WedTime3VSDoor=255
ThuTime1=2359   ThuTime1VSUser=255  ThuTime1VSDoor=255  ThuTime2=0
ThuTime2VSUser=255  ThuTime2VSDoor=255  ThuTime3=0  ThuTime3VSUser=255
ThuTime3VSDoor=255  FriTime1=2359   FriTime1VSUser=255
FriTime1VSDoor=255  FriTime2=0  FriTime2VSUser=255  FriTime2VSDoor=255
FriTime3=0  FriTime3VSUser=255  FriTime3VSDoor=255  SatTime1=2359
SatTime1VSUser=255  SatTime1VSDoor=255  SatTime2=0  SatTime2VSUser=255
SatTime2VSDoor=255  SatTime3=0  SatTime3VSUser=255  SatTime3VSDoor=255
Hol1Time1=2359  Hol1Time1VSUser=255 Hol1Time1VSDoor=255 Hol1Time2=0
Hol1Time2VSUser=255 Hol1Time2VSDoor=255 Hol1Time3=0 Hol1Time3VSUser=255
```

```
Hol1Time3VSDoor=255 Hol2Time1=2359  Hol2Time1VSUser=255
Hol2Time1VSDoor=255 Hol2Time2=0 Hol2Time2VSUser=255 Hol2Time2VSDoor=255
Hol2Time3=0 Hol2Time3VSUser=255 Hol2Time3VSDoor=255 Hol3Time1=2359
Hol3Time1VSUser=255 Hol3Time1VSDoor=255 Hol3Time2=0 Hol3Time2VSUser=255
Hol3Time2VSDoor=255 Hol3Time3=0 Hol3Time3VSUser=255 Hol3Time3VSDoor=255
```

The client uploads the successful execution result:

```
ID=521&return=0&CMD=DATA
```

Verification Modes for Different Doors in Different Periods

Application scenario

The server delivers the verification modes for different doors in different time periods to the client. It is generally used to synchronize the verification modes edited on the server to the client.

The command format is as follows:

The server delivers the command:

```
C:${CmdID}:DATA$(SP)UPDATE$(SP)DoorVSTimezone$(SP)DoorID=${XXX}$(HT)Timez
oneID=${XXX}$(HT)DevID=${XXX}
```

The client uploads the execution result:

```
ID=${CmdID}&return=${XXX}&CMD=DATA
```

Annotation

Parameter	Description
DoorVSTimezone	The table name, which is the table of verification modes for different doors in different time periods.
DoorID	Door ID
TimezoneID	Correspond to DiffTimezoneVS
DevID	The device ID. The value of AccSupportFunList counts from 0, and the value of the 26th bit is used to control the function.
${LF} is used to connect multiple records.	

Example

The server delivers:

```
C:522:DATA UPDATE DoorVSTimezone DoorID=4   TimezoneID=1    DevID=14
```

The client uploads the successful execution result:

```
ID=522&return=0&CMD=DATA
```

Verification Modes for Different Users in Different Periods

Application scenario

The server delivers the verification modes for different users in different time periods to the client. It is generally used to synchronize the verification modes edited on the server to the client.

The command format is as follows:

The server delivers the command:

```
C:${CmdID}:DATA$(SP)UPDATE$(SP)PersonalVSTimezone$(SP)PIN=${XXX}$(HT)Door
ID=${XXX}$(HT)TimezoneID=${XXX}$(HT)DevID=${XXX}
```

The client uploads the execution result:

```
ID=${CmdID}&return=${XXX}&CMD=DATA
```

Annotation

Parameter	Description
PersonalVSTimezone	The table name, which is the table of verification modes for different users in different time periods.
PIN	User ID
DoorID	Door ID
TimezoneID	Correspond to DiffTimezoneVS
DevID	The device ID. The value of AccSupportFunList counts from 0, and the value of the 26th bit is used to control the function.
${LF} is used to connect multiple records.	

Example

The server delivers:

```
C:523:DATA UPDATE PersonalVSTimezone PIN=1 DoorID=4 TimezoneID=1
DevID=14
```

The client uploads the successful execution result:

```
ID=523&return=0&CMD=DATA
```

Super User Privilege

Application scenario

The server delivers the super user privilege to the client. It is generally used to synchronize the super user privilege edited on the server to the client.

The command format is as follows:

The server delivers the command:

```
C:${CmdID}:DATA$(SP)UPDATE$(SP)SuperAuthorize$(SP)Pin=${XXX}$(HT)DoorID=$
{XXX}$(HT)DevID=${XXX}
```

The client uploads the execution result:

```
ID=${CmdID}&return=${XXX}&CMD=DATA
```

Annotation

Parameter	Description
SuperAuthorize	The table name, which is the super user privilege table.
Pin	User ID
DoorID	Door ID
DevID	The device ID. The value of AccSupportFunList counts from 0, and the value of the 26th bit is used to control the function.
${LF} is used to connect multiple records.	

Example

The server delivers:

```
C:524:DATA UPDATE SuperAuthorize Pin=1  DoorID=4    DevID=14
```

The client uploads the successful execution result:

```
ID=523&return=0&CMD=DATA
```

Comparison Photo (supported by visible light devices only)

Application scenario

The server delivers comparison photos.

The command format is as follows:

The server delivers the command:

```
C:${CmdID}:DATA${SP}UPDATE${SP}biophoto${SP}PIN=${XXX}${HT}Type=${XXX}${HT}Size=${XXX}${HT}Content=${XXX}${HT}Format=${XXX}${HT}Url=${XXX}${HT}PostBackTmpFlag=${XXX}
```

The client uploads the execution result:

```
ID=${CmdID}&return=${XXX}&CMD=DATA
```

Annotation

<table>
<tr><th>Parameter</th><th>Description</th></tr>
<tr><td>PIN=${XXX}</td><td>User ID.</td></tr>
<tr><td>Type=${XXX}</td><td>Biometric type:
<table>
<tr><th>Value</th><th>Meaning</th></tr>
<tr><td>0</td><td>General</td></tr>
<tr><td>1</td><td>Fingerprint</td></tr>
<tr><td>2</td><td>Face (Near Infrared)</td></tr>
<tr><td>3</td><td>Voice Print</td></tr>
<tr><td>4</td><td>Iris</td></tr>
<tr><td>5</td><td>Retina</td></tr>
<tr><td>6</td><td>Palm Print</td></tr>
</table></td></tr>
</table>

	7 Finger Vein 8 Palm 9 Visible Light Face
Size=${XXX}	The length of the base64-encoded binary biometric photo.
Content=${XXX}	The original binary biometric photo must be base64-encoded before transmission.
Url=${XXX}	The path where the file is stored on the server. Currently, only photos in JGP format are supported.
Format=${XXX}	The delivery mode: 0 base64 1 URL
PostBackTmpFlag=${XXX}	Whether to return the template data after image conversion: 0 not required 1 required No PostBackTmpFlag parameter, it is not required to return by default
Note: When URL is a relative path, use the relative path directly	
${LF} is used to connect multiple records.	

User Photo

Application scenario

The server delivers the user photo.

The command format is as follows:

The server delivers the command:

```
C:${CmdID}:DATA${SP}UPDATE${SP}userpic${SP}pin=${XXX}${HT}size=${XXX}${HT}format=${xxx}${HT}url=${xxx}${HT}content=${XXX}
```

The client uploads the execution result:

```
ID=${CmdID}&return=${XXX}&CMD=DATA
```

Annotation

Parameter	Description
pin=${XXX}	User ID.
Size=${XXX}	The length of the base64-encoded binary user photo.
format=${xxx}	The delivery mode: 0 base64 1 URL
url=${xxx}	Server user photo storage address.
${LF} is used to connect multiple records.	

Example

The server delivers:

```
C:524:DATA UPDATE userpic pin=2 size=138620
content=/9j/4AAQSkZJRgABAQAAAQABAAD/2wBDAAEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBA
QEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQH/2wBDAQEBAQEBAQEBAQ
EBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQH
/wAARCAGQAoADASIAAhEBAxEB/8QAHwAAAQUBAQEBAQ…...
```

The client uploads the successful execution result:

```
ID=524&return=0&CMD=DATA
```

Unified Template

Application scenario

The server delivers the unified template.

The command format is as follows:

The server delivers the command:

```
C:${CmdID}:DATA${SP}UPDATE${SP}biodata${SP}Pin=${XXX}${HT}No=${XXX}${HT}I
ndex=${XXX}${HT}Valid=${XXX}${HT}Duress=${XXX}${HT}Type=${XXX}${HT}MajorV
er=${XXX}${HT}MinorVer=${XXX}${HT}Format=${XXX}${HT}Tmp=${XXX}
```

The client uploads the execution result:

```
ID=${CmdID}&return=${XXX}&CMD=DATA
```

Annotation

<table>
<tr><th>Parameter</th><th>Description</th></tr>
<tr><td>Pin=${XXX}</td><td>User ID.</td></tr>
<tr><td>No=${XXX}</td><td>The number of specific biont. The default value is 0.
<table>
<tr><td>Fingerprint</td><td>The number ranges from 0 to 9, corresponding to the little finger, ring finger, middle finger, index finger, and thumb of the left hand, and thumb, index finger, middle finger, ring finger, and little finger of the right hand respectively.</td></tr>
<tr><td>Finger Vein</td><td>It is same as of the fingerprint.</td></tr>
<tr><td>Face</td><td>All is 0.</td></tr>
<tr><td>Iris</td><td>0 is the iris of the left eye and 1 is the iris of the right eye.</td></tr>
<tr><td>Palm</td><td>0 is the palm of the left hand and 1 is the palm of the right hand.</td></tr>
<tr><td>Visible Light Palm</td><td>0 is the palm of the left hand and 1 is the palm of the right hand.</td></tr>
</table></td></tr>
<tr><td>Index=${XXX}</td><td>The number of the specific biont template. Multiple templates may be stored for one finger. It is calculated from 0</td></tr>
<tr><td>Valid=${XXX}</td><td>Whether it is valid. The default value is 1.
0 Invalid
1 Valid</td></tr>
<tr><td>Duress=${XXX}</td><td>Whether it is duress. The default value is 0.
0 Not Duress
1 Duress</td></tr>
</table>

<table>
<tr><td>Type=${XXX}</td><td>Biometric type:<table><tr><th>Value</th><th>Meaning</th></tr><tr><td>0</td><td>General</td></tr><tr><td>1</td><td>Fingerprint</td></tr><tr><td>2</td><td>Face (Near Infrared)</td></tr><tr><td>3</td><td>Voice Print</td></tr><tr><td>4</td><td>Iris</td></tr><tr><td>5</td><td>Retina</td></tr><tr><td>6</td><td>Palm Print</td></tr><tr><td>7</td><td>Finger Vein</td></tr><tr><td>8</td><td>Palm</td></tr><tr><td>9</td><td>Visible Light Face</td></tr><tr><td>10</td><td>Visible Light Palm</td></tr></table></td></tr>
<tr><td>MajorVer=${XXX}</td><td>The major version number. For example, for the fingerprint algorithm 10.3, the major version number is 10 and the minor version number is 3.
Fingerprint 9.0, 10.3 and 12.0
Finger Vein 3.0
Face 5.0, 7.0 and 8.0
Palm 1.0
Visible Light Face 58.0 and 38.0
Visible Light Palm 32.2</td></tr>
<tr><td>MinorVer=${XXX}</td><td>The minor version number. For example, for the fingerprint algorithm 10.3, the major version number is 10 and the minor version number is 3.
Fingerprint 9.0, 10.3 and 12.0
Finger Vein 3.0
Face 5.0, 7.0 and 8.0
Palm 1.0
Visible Light Face 58.0 and 38.0</td></tr>
</table>

<table>
<tr><td></td><td>Visible Light Palm 32.2</td></tr>
<tr><td>pin=${XXX}</td><td>Name of attendance photo.</td></tr>
<tr><td>Format=${XXX}</td><td>The template format.
The fingerprint template formats include ZK, ISO, and ANSI.
• Fingerprint
<table><tr><th>Value</th><th>Format</th></tr><tr><td>0</td><td>ZK</td></tr><tr><td>1</td><td>ISO</td></tr><tr><td>2</td><td>ANSI</td></tr></table>• Finger Vein
<table><tr><th>Value</th><th>Format</th></tr><tr><td>0</td><td>ZK</td></tr></table>• Face
<table><tr><th>Value</th><th>Format</th></tr><tr><td>0</td><td>ZK</td></tr></table>• Palm
<table><tr><td>Value</td><td>Format</td></tr><tr><td>0</td><td>ZK</td></tr></table>• Visible Light Palm
<table><tr><td>Value</td><td>Format</td></tr><tr><td>0</td><td>ZK</td></tr></table></td></tr>
<tr><td>Tmp=${XXX}</td><td>The template data. The original binary fingerprint template must be base64-encoded.</td></tr>
<tr><td colspan="2">${LF} is used to connect multiple records.</td></tr>
</table>

Example

The server delivers:

```
C:524:DATA UPDATE biodata Pin=2 No=0 Index=0 Valid=1 Duress=0 Type=9
Majorver=5 Minorver=8 Format=0 Tmp=AAAAA......
```

The client uploads the successful execution result:

```
ID=524&return=0&CMD=DATA
```

Note: If using a subcommand to deliver an integrated template, the same command must use the same type of template.

Example:

```
C:8748:DATA UPDATE biodata Pin=1 No=0 Type=9 Index=0 Valid=1 Duress=0 ...
Tmp=XXX
                    Pin=2 No=0 Type=9 Index=0 Valid=1 Duress=0 ... Tmp=XXX
                    Pin=3 No=0 Type=9 Index=0 Valid=1 Duress=0 ... Tmp=XXX
```

Elevator Control Privilege Group

Application scenario

The server delivers the elevator control privilege information to the client, which is generally used to edit the elevator control privilege group information on the server and then synchronize it to the client.

The command format is as follows:

The server delivers the command:

```
C:${CmdID}:DATA${SP}UPDATE${SP}usereleauthorize${SP}Pin=${XXX}${HT}Author
izeTimezoneId=${XXX}${HT}AuthorizeFloorId=${XXX}${HT}AuthorizeFloors=${XX
X}${HT}DevID=${XXX}
```

The client uploads the execution result:

```
ID=${CmdID}&return=${XXX}&CMD=DATA
```

Annotation

Parameter	Description
Pin=${XXX}	User ID.
AuthorizeTimezoneId=${XXX}	Effective time period number.
AuthorizeFloorId=${XXX}	Direct floor number.
AuthorizeFloors=${XXX}	Selected floor number, using bits to indicate whether the

	specified floor is authorized or not, the maximum is 32 bytes, which can represent 256 floors, in the order of low to high. A bit value of 0 means not authorized, 1 means authorized. For example, if floor 8 and floor 9 are authorized, the delivered value is 180.
DevID=${XXX}	Device ID.

Example

The server delivers:

```
C:524:DATA UPDATE biodata Pin=2 AuthorizeTimezoneId=1 AuthorizeFloorId=1
AuthorizeFloors=180
```

The client uploads the successful execution result:

```
ID=524&return=0&CMD=DATA
```

Expansion Board Property

Application scenario

The server delivers the expansion board properties to the client, which is generally used to add or delete the expansion board on the server and synchronize it to the client.

The command format is as follows:

The server delivers the command:

```
C:${CmdID}:DATA${SP}UPDATE${SP}ExtBoardID${SP}DevID=${XXX}${HT}DevType=${
XXX}${HT}BusType=${XXX}${HT}Address=${XXX}${HT}Disable=${XXX}
```

The client uploads the execution result:

```
ID=${CmdID}&return=${XXX}&CMD=DATA
```

Annotation

Parameter	Description
ExtBoardID=${XXX}	Expansion board property ID.
DevID=${XXX}	Device ID.

DevType=${XXX}	The expansion board type:		
	DM10	101	Access control expansion board, 1 door, 1 auxiliary output, 1 auxiliary input, 2 Wiegand
	WR485	102	Wiegand to RS485 reader board
	Aux485	103	IO expansion board, 4 auxiliary output
	EX308	104	8 floor expansion board used by elevator control, 8 auxiliary output
	EX316	105	16 floor expansion board used by elevator control, 16 auxiliary output
	EX0808	110	8 auxiliary output, 8 auxiliary input
BusType=${XXX}	Temporarily useless, just fill in 0.		
Address=${XXX}	RS485 address of the expansion board. The value is 1~8, depending on the dialing address of the expansion board.		
Disable=${XXX}	Whether to disable.		

Example

The server delivers:

```
C:524:DATA UPDATE ExtBoardID=1 DevID=1 DevType=101 BusType=0 Address=4
Disable=0
```

The client uploads the successful execution result:

```
ID=524&return=0&CMD=DATA
```

Expansion Board Configuration

Application scenario

The server delivers the expansion board configuration information to the client, which is generally used to add or delete the expansion board on the server and synchronize it to the client.

The command format is as follows:

The server delivers the command:

```
C:${CmdID}:DATA${SP}UPDATE${SP}ExtBoardID${SP}ReactType=${XXX}${HT}Entity
ID=${XXX}${HT}SeqNo=${XXX}
```

The client uploads the execution result:

```
ID=${CmdID}&return=${XXX}&CMD=DATA
```

Annotation

Parameter	Description
ExtBoardID=${XXX}	Expansion board property ID.
ReactType=${XXX}	The interface types supported by the expansion board: 1 Door 2 Auxiliary iutput 3 Auxiliary output 4 Reader
EntityID=${XXX}	The interface number is determined by ReactType. When the expansion board is added in the software, the corresponding interface number is based on the hardware address (from small to large) according to the number of interfaces supported by the expansion board type.
SeqNo=${XXX}	Interface hardware address, which is determined by ReactType. When the expansion board is added in the software, the hardware address matches the interface number.

Example

The server delivers:

```
C:524:DATA UPDATE ExtBoardID=1 ReactType=1 EntityID=1 SeqNo=1
```

The client uploads the successful execution result:

```
ID=524&return=0&CMD=DATA
```

Time Rule and Passing Mode (only supported by the channel controller)

Application scenario

The server delivers a time rule and corresponding passing mode to the client. Generally, it is used to automatically deliver the rule edited on the server.

The command format is as follows:

The server delivers the command:

```
C:${CmdID}:DATA$(SP)UPDATE$(SP)gatepassrule$(SP)PasswayTimeZoneId=${XXX}$
(HT)SunTime1=${XXX}$(HT)SunPassMode1=${XXX}$(HT)SunTime2=${XXX}$(HT)SunPa
ssMode2=${XXX}$(HT)SunTime3=${XXX}$(HT)SunPassMode3=${XXX}$(HT)SunTime4=$
{XXX}$(HT)SunPassMode4=${XXX}$(HT)SunTime5=${XXX}$(HT)SunPassMode5=${XXX}
$(HT)MonTime1=${XXX}$(HT)MonPassMode1=${XXX}$(HT)MonTime2=${XXX}$(HT)MonP
assMode2=${XXX}$(HT)MonTime3=${XXX}$(HT)MonPassMode3=${XXX}$(HT)MonTime4=
${XXX}$(HT)MonPassMode4=${XXX}$(HT)MonTime5=${XXX}$(HT)MonPassMode5=${XXX
}$(HT)TueTime1=${XXX}$(HT)TuePassMode1=${XXX}$(HT)TueTime2=${XXX}$(HT)Tue
PassMode2=${XXX}$(HT)TueTime3=${XXX}$(HT)TuePassMode3=${XXX}$(HT)TueTime4
=${XXX}$(HT)TuePassMode4=${XXX}$(HT)TueTime5=${XXX}$(HT)TuePassMode5=${XX
X}$(HT)WedTime1=${XXX}$(HT)WedPassMode1=${XXX}$(HT)WedTime2=${XXX}$(HT)We
dPassMode2=${XXX}$(HT)WedTime3=${XXX}$(HT)WedPassMode3=${XXX}$(HT)WedTime
4=${XXX}$(HT)WedPassMode4=${XXX}$(HT)WedTime5=${XXX}$(HT)WedPassMode5=${X
XX}$(HT)ThuTime1=${XXX}$(HT)ThuPassMode1=${XXX}$(HT)ThuTime2=${XXX}$(HT)T
huPassMode2=${XXX}$(HT)ThuTime3=${XXX}$(HT)ThuPassMode3=${XXX}$(HT)ThuTim
e4=${XXX}$(HT)ThuPassMode4=${XXX}$(HT)ThuTime5=${XXX}$(HT)ThuPassMode5=${
XXX}$(HT)FriTime1=${XXX}$(HT)FriPassMode1=${XXX}$(HT)FriTime2=${XXX}$(HT)
FriPassMode2=${XXX}$(HT)FriTime3=${XXX}$(HT)FriPassMode3=${XXX}$(HT)FriTi
me4=${XXX}$(HT)FriPassMode4=${XXX}$(HT)FriTime5=${XXX}$(HT)FriPassMode5=$
{XXX}$(HT)SatTime1=${XXX}$(HT)SatPassMode1=${XXX}$(HT)SatTime2=${XXX}$(HT
)SatPassMode2=${XXX}$(HT)SatTime3=${XXX}$(HT)SatPassMode3=${XXX}$(HT)SatT
ime4=${XXX}$(HT)SatPassMode4=${XXX}$(HT)SatTime5=${XXX}$(HT)SatPassMode5=
${XXX}
```

The client uploads the execution result:

```
ID=${CmdID}&return=${XXX}&CMD=DATA
```

Annotation

Parameter	Description
timezone	The table name, which is time group
TimezoneId	The ID of the time rule **xxxTime1~xxxTime5:** From Sunday to Saturday, five time periods each day. **xxxPassMode1~xxxPassMode5:** The passing mode corresponding to each time period.
Note: Each time period lasts from StartTime to EndTime on the software. Before the time rule is delivered to the device, the software converts the time rule to an integer based on the convention StartTime<<16+EndTime. For example, the converted value of 8:30-12:00 is 830 << 16 + 1200, that is, 0x33e04b0.	

NTP Server Information

Application scenario

The server delivers the NTP server information to the device, which is generally used for the device to use the NTP server to automatically synchronize the time.

The command format is as follows:

The server delivers the command:

```
C:${CmdID}:DATA$(SP)UPDATE$(SP)ntpserverlist$(SP)ServerAddress=${XXX}
```

The client uploads the execution result:

```
ID=${CmdID}&return=${XXX}&CMD=DATA
```

Annotation

Parameter	Description
ServerAddress	NTP server address

Wiegand Format Information

Application scenario

The server delivers the Wiegand format information data to the client, typically used for automatic synchronization of edited Wiegand format information data from the server to the client.

The command format is as follows:

The server delivers the command:

```
C:${CmdID}:DATA$(SP)UPDATE$(SP)WGFormat$(SP)ID=${XXX}$(HT)CardBit=${XXX}$(HT)
SiteCode=${XXX}$(HT)FormatName=${XXX}$(HT)CardFormat=${XXX}$(HT)
ExtSiteCode=${XXX}$(HT)ExtParityFormat=${XXX}$(HT)IsDefault=${XXX}
```

The client uploads the execution result:

```
ID=${CmdID}&return=${XXX}&CMD=DATA
```

Annotation

Parameter	Description
WGFormat	Table name, referring to the Wiegand format table
ID	Index
CardBit	Bit number
SiteCode	Site code
FormatName	Wiegand custom name
CardFormat	Wiegand format
ExtSiteCode	Extension, supports multiple site codes separated by semicolons (only supported in WGRuleVer=V2.0 and above versions)
ExtParityFormat	Extension, supports multiple parity formats separated by semicolons (only supported in WGRuleVer=V2.0 and above versions)
IsDefault	Whether the current Wiegand format participates in the Wiegand automatic matching rule, with a value of 0 or 1. 0 represents not participating in the Wiegand automatic matching rule, and 1 represents participating in the Wiegand automatic matching rule. (Only supported in WGRuleVer=V2.0 and above versions)
${LF} is used to connect multiple records.	

Example

The server delivers:

```
(1) WGRuleVer=V1.0
    C:518:DATA UPDATE WGFormat ID=2 CardBit=26 SiteCode=0
    FormatName=Wiegand26
    CardFormat=pccccccccccccccccccccccp:eeeeeeeeeeeeeooooooooooooo
(2) WGRuleVer=V2.0
    C:518:DATA UPDATE WGFormat ID=2 CardBit=26 SiteCode=0 FormatName=
    Wiegand26
    CardFormat=pccccccccccccccccccccccp:eeeeeeeeeeeeeooooooooooooo
    ExtSiteCode=1000;2000
    ExtParityFormat=xeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeex;xooooooooooooooo
    ooooooooooooooxo
    IsDefault=0
```

The client uploads the successful execution result:

```
ID=518&return=0&CMD=DATA
```

Contact List Basic Information

Application scenario

The server delivers the contact list basic information to the client, which is generally used to automatically synchronize with the client after editing the contact list basic information on the server.

The command format is as follows:

The server delivers the command:

```
C:${CmdID}:DATA$(SP)UPDATE$(SP)contact_info$(SP)ContactID=${XXX}$(HT)Pin=
${XXX}$(HT)
GroupName=${XXX}$(HT)DisplayName=${XXX}$(HT)PassCode=${XXX}
```

The client uploads the execution result:

```
ID=${CmdID}&return=${XXX}&CMD=DATA
```

Annotation

Parameter	Description
ContactID	Contact list unique ID
Pin	Related personnel User ID
GroupName	Grouping, can be used for classification and pre search (primary filtering)
DisplayName	Display name
PassCode	Remote door opening password

Example

The server delivers:

```
C:110:DATA UPDATE contact_info ContactID=1  Pin=1000   GroupName=1#101
DisplayName=Alfred  PassCode=123456
```

The client uploads the successful execution result:

```
ID=110&return=0&CMD=DATA
```

Contact Number List

Application scenario

The server delivers the contact number list to the client, a contact can have multiple numbers.

The command format is as follows:

The server delivers the command:

```
C:${CmdID}:DATA$(SP)UPDATE$(SP)contact_number_list$(SP)ContactID=${XXX}$(
HT)NumType=${XXX}$(HT)
Number=${XXX}$(HT)Priority=${XXX}$(HT)PatternID=${XXX}
```

The client uploads the execution result:

```
ID=${CmdID}&return=${XXX}&CMD=DATA
```

Annotation

Parameter	Description
ContactID	Contact list unique ID
NumType	Number type: 1. Phone number; 2. Account name; 3. IP address or domain name
Number	Phone number/Account name/ IP address or domain name
Priority	Priority. When a contact person has multiple contact methods, they should be dialed based on priority, starting from 1. The higher the number, the higher the priority.
PatternID	PatternID in the contact number rule list, 0 indicates none

Example

The server delivers:

```
C:110:DATA UPDATE contact_number_list ContactID=1  NumType=1
Number=15865742356 Priority=1  PatternID=1
```

The client uploads the successful execution result:

```
ID=110&return=0&CMD=DATA
```

Contact Number Rule List

Application scenario

The server delivers the contact number rule list to the client, specifying specific rules for any contact number.

The command format is as follows:

The server delivers the command:

```
C:${CmdID}:DATA$(SP)UPDATE$(SP)contact_number_pattern_list$(SP)PatternID=
${XXX}$(HT)Pattern=${XXX}$(HT)

Strip=${XXX}$(HT)Prepend=${XXX}
```

The client uploads the execution result:

```
ID=${CmdID}&return=${XXX}&CMD=DATA
```

Annotation

Parameter	Description
PatternID	Number rule ID
Pattern	Number rules, using regular expressions such as 9 *, 9 [1-9], etc
Strip	Prefix digit, which is a number greater than 0, for example, 2 means the prefix digit is 2.
Prepend	When the number dialed by the user meets the dialing rules, PBX needs to add a prefix number before the number dialed by the user before sending it out (keep unused).

Example

The server delivers:

```
C:110:DATA UPDATE contact_number_pattern_list PatternID=1  Pattern=9[1-9]
Strip=2  PatternID=1
```

The client uploads the successful execution result:

```
ID=110&return=0&CMD=DATA
```

9.1.2 Delete

Application scenario

This command is used to control the deletion of device data on the software, for example, the user and fingerprint template.

Format

```
C:$(CmdID):DATA$(SP)DELETE$(SP)$(TableName)$(SP)$(Cond)
```

Annotation

Parameter	Description
$(Cond)	The deletion condition list, in the format of: `Condition field name1=value1$(HT)condition field name2=value2$(HT)condition field name3=value3` When the deletion condition list is *, it means to clear the table. When the

	deletion condition list contains multiple conditions, the conditions are in AND relationship.
${LF} is used to connect multiple records.	

User Information

Application scenario

It is generally used to delete a user or all users.

Format

The server delivers the command:

```
C:$(CmdID):DATA$(SP)DELETE$(SP)user$(SP)$(Cond)
```

The client uploads the execution result:

```
ID=${CmdID}&return=${XXX}&CMD=DATA
```

Annotation

Parameter	Description
user	The table name, which is the user information
$(Cond)	Generally, Pin=x means deleting the user of which ID is x, and * means deleting all users.
Note: Generally, when deleting a user, the server deletes the user's verification mode and access control privilege together. Therefore, when delivering the command to delete a user, the server also delivers the command to delete the user's access control privilege and verification mode (such as a fingerprint).	
${LF} is used to connect multiple records.	

Example

Delete the user of which ID is 1 (can delete the user data together):

```
C:335:DATA DELETE userauthorize Pin=1
```

```
C:336:DATA DELETE templatev10 Pin=1

C:337:DATA DELETE user Pin=1
```

Wherein, commands 1 and 2 are used to delete the access control privilege and the fingerprint template respectively. Command 3 is used to delete the user.

The client returns the execution result:

```
ID=335&return=0&CMD=DATA

ID=336&return=0&CMD=DATA

ID=337&return=0&CMD=DATA
```

Extended User Information

Application scenario

It is generally used to delete an extended user or all extended users.

Format

The server delivers the command:

```
C:$(CmdID):DATA$(SP)DELETE$(SP)extuser$(SP)$(Cond)
```

The client uploads the execution result:

```
ID=${CmdID}&return=${XXX}&CMD=DATA
```

Annotation

Parameter	Description
extuser	The table name, which is the extended user information
$(Cond)	Generally, Pin=x means deleting the extended user of which ID is x, and * means deleting all extended users.

Example

Delete the user of which ID is 1:

```
C:337:DATA DELETE extuser Pin=1
```

The client returns the execution result:

```
ID=337&return=0&CMD=DATA
```

Multi-card User

Application scenario

It is generally used to delete a multi-card user or all multi-card users.

Format

The server delivers the command:

```
C:$(CmdID):DATA$(SP)DELETE$(SP)mulcarduser$(SP)$(Cond)
```

The client uploads the execution result:

```
ID=${CmdID}&return=${XXX}&CMD=DATA
```

Annotation

Parameter	Description
mulcarduser	The table name, which is the multi-card user.
$(Cond)	Generally, Pin=x means deleting the multi-card user of which ID is x, and * means deleting all multi-card users.

Example

Delete the user of which ID is 1:

```
C:337:DATA DELETE mulcarduser Pin=1
```

The client returns the execution result:

```
ID=337&return=0&CMD=DATA
```

User's Access Control Privilege

Application scenario

After deleting a user, the server also deletes the user's access control privilege.

Format

The server delivers the command:

```
C:(CmdID):DATA$(SP)DELETE$(SP)userauthorize$(SP)$(Cond)
```

The client uploads the execution result:

```
ID=${CmdID}&return=${XXX}&CMD=DATA
```

Annotation

Parameter	Description
userauthorize	The table name, which is the user's access control privilege.
$(Cond)	Generally, Pin=x means deleting the access control privilege of the user of which ID is x, and * means deleting the access control privilege of all users.

Example

Delete the access control privilege of the user of which ID is 1:

```
C:335:DATA DELETE userauthorize Pin=1
```

The client returns the execution result:

```
ID=335&return=0&CMD=DATA
```

Access Control Record

Application scenario

It is generally used to delete all the access control records of the device.

Format

The server delivers the command:

```
C:(CmdID):DATA$(SP)DELETE$(SP)transaction$(SP)$(Cond)
```

The client uploads the execution result:

```
ID=${CmdID}&return=${XXX}&CMD=DATA
```

Annotation

Parameter	Description
transaction	The table name, which is the access control record.
$(Cond)	* means deleting all the records.

Example

Delete all the access control records:

```
C:123:DATA DELETE transaction *
```

The client returns the execution result:

```
ID=123&return=0&CMD=DATA
```

Fingerprint Template

Application scenario

Generally, it is used to delete a specified fingerprint template of a device used on the server or delete a user as well as the user's fingerprint template.

Format

The server delivers the command:

```
C:${CmdID}:DATA$(SP)DELETE$(SP)templatev10$(SP)$(Cond)
```

The client uploads the execution result:

```
ID=${CmdID}&return=${XXX}&CMD=DATA
```

Annotation

Parameter	Description

templatev10	The table name, which is the user's fingerprint template.
$(Cond)	Pin=x means deleting all the fingerprint templates of the user of which ID is x. FingerID=x means deleting a specified fingerprint template. * means deleting all the records.

Example

Delete the fingerprint template of a user of which ID is 1 and fingerprint ID is 6:

```
C:342:DATA DELETE templatev10 Pin=1 FingerID=6
```

The client returns the execution result:

```
ID=342&return=0&CMD=DATA
```

Holiday

Application scenario

It is generally used to delete all the holidays of the device.

Format

The server delivers the command:

```
C:(CmdID):DATA$(SP)DELETE$(SP)holiday$(SP)$(Cond)
```

The client uploads the execution result:

```
ID=${CmdID}&return=${XXX}&CMD=DATA
```

Annotation

Parameter	Description
holiday	The table name, which is the holiday.
$(Cond)	* means deleting all the records.

Example

Delete all the holidays:

```
C:123: DATA DELETE holiday *
```

The client returns the execution result:

```
ID=123&return=0&CMD=DATA
```

Time Period

Application scenario

It is generally used to delete the data in a time period or the data in all time periods.

Format

The server delivers the command:

```
C:${CmdID}:DATA$(SP)DELETE$(SP)timezone$(SP)$(Cond)
```

The client uploads the execution result:

```
ID=${CmdID}&return=${XXX}&CMD=DATA
```

Annotation

Parameter	Description
`holiday`	The table name, which is the time period.
`$(Cond)`	TimezoneId=x means deleting the time period of which TimezoneId is x. * means deleting all the time periods.

Example

Delete a time period of which TimezoneId is 1:

```
C:342:DATA DELETE timezone TimezoneId=1
```

The client returns the execution result:

```
ID=342&return=0&CMD=DATA
```

First-card Opening Record

Application scenario

It is generally used to delete a first-card opening record or all the first-card opening records.

Format

The server delivers the command:

```
C:${CmdID}:DATA$(SP)DELETE$(SP)firstcard$(SP)$(Cond)
```

The client uploads the execution result:

```
ID=${CmdID}&return=${XXX}&CMD=DATA
```

Annotation

Parameter	Description
firstcard	The table name, which is the first-card opening record
$(Cond)	DoorID=x means deleting the first-card opening record of which DoorID is x. Pin=x means deleting the first-card opening record of which Pin is x. * means deleting all the first-card opening records.

Example

Delete the first-card opening record of which DoorID or Pin is 1:

```
C:342:DATA DELETE firstcard DoorID=1
```

```
C:343: DATA DELETE firstcard Pin=1
```

The client returns the execution result:

```
ID=342&return=0&CMD=DATA
```

```
ID=343&return=0&CMD=DATA
```

Multi-card Opening Record

Application scenario

It is generally used to delete a multi-card opening record or all the multi-card opening records.

Format

The server delivers the command:

```
C:${CmdID}:DATA$(SP)DELETE$(SP)multimcard$(SP)$(Cond)
```

The client uploads the execution result:

```
ID=${CmdID}&return=${XXX}&CMD=DATA
```

Annotation

Parameter	Description
multimcard	The table name, which is the multi-card opening table
$(Cond)	Index=x means deleting the multi-card opening record of which Index is x. * means deleting all the multi-card opening records.

Example

Delete the multi-card opening record of which Index is 1:

```
C:342:DATA DELETE multimcard Index=1
```

The client returns the execution result:

```
ID=342&return=0&CMD=DATA
```

Linkage Details

Application scenario

It is generally used to delete a linkage detail or all the linkage details.

Format

The server delivers the command:

```
C:${CmdID}:DATA$(SP)DELETE$(SP)inoutfun$(SP)$(Cond)
```

The client uploads the execution result:

```
ID=${CmdID}&return=${XXX}&CMD=DATA
```

Annotation

Parameter	Description

inoutfun	The table name, which is the linkage details table
$(Cond)	Index=x means deleting the multi-card opening record of which Index is x. * means deleting all the multi-card opening records.

Example

Delete the linkage details of which Index is 1:

```
C:342:DATA DELETE inoutfun Index=1
```

The client returns the execution result:

```
ID=342&return=0&CMD=DATA
```

Scheduled Output

Application scenario

It is generally used to delete a scheduled output or all the scheduled outputs.

Format

The server delivers the command:

```
C:${CmdID}:DATA$(SP)DELETE$(SP)outrelaysetting$(SP)$(Cond)
```

The client uploads the execution result:

```
ID=${CmdID}&return=${XXX}&CMD=DATA
```

Annotation

Parameter	Description
outrelaysetting	The table name, which is the scheduled output table
$(Cond)	Num=x means deleting the scheduled output of which Num is x. * means deleting all the scheduled outputs.

Example

Delete all the scheduled outputs:

```
C:342:DATA DELETE outrelaysetting Num=1
```

The client returns the execution result:

```
ID=342&return=0&CMD=DATA
```

DLST Record

Application scenario

It is generally used to delete all the DLST records.

Format

The server delivers the command:

```
C:${CmdID}:DATA$(SP)DELETE$(SP)DSTSetting$(SP)$(Cond)
```

The client uploads the execution result:

```
ID=${CmdID}&return=${XXX}&CMD=DATA
```

Annotation

Parameter	Description
DSTSetting	The table name, which is the DLST table
$(Cond)	* means deleting all the DLST records.

Example

Delete the scheduled output of which Num is 1:

```
C:342: DATA DELETE DSTSetting *
```

The client returns the execution result:

```
ID=342&return=0&CMD=DATA
```

Device Property

Application scenario

It is generally used to delete all the device properties.

Format

The server delivers the command:

```
C:${CmdID}:DATA$(SP)DELETE$(SP)DevProperty$(SP)$(Cond)
```

The client uploads the execution result:

```
ID=${CmdID}&return=${XXX}&CMD=DATA
```

Annotation

Parameter	Description
DevProperty	The table name, which is the device property table
$(Cond)	* means deleting all the device properties.

Example

Delete all the device properties:

```
C:342: DATA DELETE DevProperty *
```

The client returns the execution result:

```
ID=342&return=0&CMD=DATA
```

Device Parameter

Application scenario

It is generally used to delete all the device parameters.

Format

The server delivers the command:

```
C:${CmdID}:DATA$(SP)DELETE$(SP)DevParameters$(SP)$(Cond)
```

The client uploads the execution result:

```
ID=${CmdID}&return=${XXX}&CMD=DATA
```

Annotation

Parameter	Description
DevParameters	The table name, which is the device parameter table
$(Cond)	* means deleting all the device parameters.

Example

Delete all the device parameters:

```
C:342: DATA DELETE DevParameters *
```

The client returns the execution result:

```
ID=342&return=0&CMD=DATA
```

Door Property

Application scenario

It is generally used to delete all the door properties.

Format

The server delivers the command:

```
C:${CmdID}:DATA$(SP)DELETE$(SP)DoorProperty$(SP)$(Cond)
```

The client uploads the execution result:

```
ID=${CmdID}&return=${XXX}&CMD=DATA
```

Annotation

Parameter	Description
DoorProperty	The table name, which is the door property table
$(Cond)	* means deleting all the door properties.

Example

Delete all the door properties:

```
C:342:DATA DELETE DoorProperty *
```

The client returns the execution result:

```
ID=342&return=0&CMD=DATA
```

Door Parameter

Application scenario

It is generally used to delete all the door parameters.

Format

The server delivers the command:

```
C:${CmdID}:DATA$(SP)DELETE$(SP)DoorParameters$(SP)$(Cond)
```

The client uploads the execution result:

```
ID=${CmdID}&return=${XXX}&CMD=DATA
```

Annotation

Parameter	Description
DoorParameters	The table name, which is the door parameter table
$(Cond)	* means deleting all the door parameters.

Example

Delete all the door parameters:

```
C:342:DATA DELETE DoorParameters *
```

The client returns the execution result:

```
ID=342&return=0&CMD=DATA
```

Reader Property

Application scenario

It is generally used to delete all the reader properties.

Format

The server delivers the command:

```
C:${CmdID}:DATA$(SP)DELETE$(SP)ReaderProperty$(SP)$(Cond)
```

The client uploads the execution result:

```
ID=${CmdID}&return=${XXX}&CMD=DATA
```

Annotation

Parameter	Description
ReaderProperty	The table name, which is the reader property table
$(Cond)	* means deleting all the reader properties.

Example

Delete all the reader properties:

```
C:342:DATA DELETE ReaderProperty*
```

The client returns the execution result:

```
ID=342&return=0&CMD=DATA
```

Reader Parameter

Application scenario

It is generally used to delete all the reader parameters.

Format

The server delivers the command:

```
C:${CmdID}:DATA$(SP)DELETE$(SP)ReaderParameters$(SP)$(Cond)
```

The client uploads the execution result:

```
ID=${CmdID}&return=${XXX}&CMD=DATA
```

Annotation

Parameter	Description
ReaderParameters	The table name, which is the reader parameter table
$(Cond)	* means deleting all the reader parameters.

Example

Delete all the reader parameters:

```
C:342:DATA DELETE ReaderParameters *
```

The client returns the execution result:

```
ID=342&return=0&CMD=DATA
```

Auxiliary Input

Application scenario

It is generally used to delete all the auxiliary inputs.

Format

The server delivers the command:

```
C:${CmdID}:DATA$(SP)DELETE$(SP)AuxIn$(SP)$(Cond)
```

The client uploads the execution result:

```
ID=${CmdID}&return=${XXX}&CMD=DATA
```

Annotation

Parameter	Description
AuxIn	The table name, which is the auxiliary input table
$(Cond)	* means deleting all the auxiliary inputs.

Example

Delete all the auxiliary inputs:

```
C:342:DATA DELETE AuxIn *
```

The client returns the execution result:

```
ID=342&return=0&CMD=DATA
```

Auxiliary Output

Application scenario

It is generally used to delete all the auxiliary outputs.

Format

The server delivers the command:

```
C:${CmdID}:DATA$(SP)DELETE$(SP)AuxOut$(SP)$(Cond)
```

The client uploads the execution result:

```
ID=${CmdID}&return=${XXX}&CMD=DATA
```

Annotation

Parameter	Description
AuxOut	The table name, which is the auxiliary output table
$(Cond)	* means deleting all the auxiliary outputs.

Example

Delete all the auxiliary inputs:

```
C:342:DATA DELETE AuxOut *
```

The client returns the execution result:

```
ID=342&return=0&CMD=DATA
```

Default Wiegand Format

Application scenario

It is generally used to delete a default Wiegand format or all the default Wiegand formats.

Format

The server delivers the command:

```
C:${CmdID}:DATA$(SP)DELETE$(SP)DefaultWGFormat$(SP)$(Cond)
```

The client uploads the execution result:

```
ID=${CmdID}&return=${XXX}&CMD=DATA
```

Annotation

Parameter	Description
DefaultWGFormat	The table name, which is the default Wiegand format table
$(Cond)	ID=x means deleting the default Wiegand format of which ID is x. CardBit=x means deleting the default Wiegand format of which CardBit is x. * means deleting all the default Wiegand formats.

Example

Delete the Wiegand format of which ID is 1 or CardBit is 26:

```
C:342:DATA DELETE DefaultWGFormat ID=1
```

```
C:343:DATA DELETE DefaultWGFormat CardBit=26
```

The client returns the execution result:

```
ID=342&return=0&CMD=DATA
```

```
ID=343&return=0&CMD=DATA
```

Wiegand Format

Application scenario

It is generally used to delete a Wiegand format or all the Wiegand formats.

Format

The server delivers the command:

```
C:${CmdID}:DATA$(SP)DELETE$(SP)WGFormat$(SP)$(Cond)
```

The client uploads the execution result:

```
ID=${CmdID}&return=${XXX}&CMD=DATA
```

Annotation

Parameter	Description
WGFormat	The table name, which is the Wiegand format table
$(Cond)	ID=x means deleting the Wiegand format of which ID is x. CardBit=x means deleting the Wiegand format of which CardBit is x. * means deleting all the Wiegand formats.

Example

Delete the Wiegand format of which ID is 1 or CardBit is 26:

```
C:342:DATA DELETE WGFormat ID=1
C:343:DATA DELETE WGFormat CardBit=26
```

The client returns the execution result:

```
ID=342&Return=0&CMD=DATA
ID=343&return=0&CMD=DATA
```

Reader Wiegand Format

Application scenario

It is generally used to delete the Wiegand format of a reader or the Wiegand formats of all readers.

Format

The server delivers the command:

```
C:${CmdID}:DATA$(SP)DELETE$(SP)WGFormat$(SP)$(Cond)
```

The client uploads the execution result:

```
ID=${CmdID}&return=${XXX}&CMD=DATA
```

Annotation

Parameter	Description
WGFormat	The table name, which is the reader Wiegand format table
$(Cond)	ReaderID=x means deleting the Wiegand format of the reader of which ReaderID is x. DevID=x means deleting the Wiegand format of the reader of which DevID is x. * means deleting the Wiegand formats of all the readers.

Example

Delete the Wiegand format of the reader of which ReaderID is 1 or DevID is 24:

```
C:342:DATA DELETE WGFormat ReaderID=1
C:343:DATA DELETE WGFormat DevID=24
```

The client returns the execution result:

```
ID=342&Return=0&CMD=DATA
ID=343&return=0&CMD=DATA
```

Input Control (by Time Period)

Application scenario

It is generally used to delete all the input control (by time period) data.

Format

The server delivers the command:

```
C:${CmdID}:DATA$(SP)DELETE$(SP)InputIOSetting$(SP)$(Cond)
```

The client uploads the execution result:

```
ID=${CmdID}&return=${XXX}&CMD=DATA
```

Annotation

Parameter	Description
InputIOSetting	The table name, which is the table of input control by time period.
$(Cond)	* means deleting all the input control (by time period) data.

Example

Delete all the input control (by time period) data:

```
C:342:DATA DELETE InputIOSetting *
```

The client returns the execution result:

```
ID=342&return=0&CMD=DATA
```

Verification Modes in Different Time Periods

Application scenario

It is generally used to delete a verification mode in different time periods or all the verification modes in different time periods.

Format

The server delivers the command:

```
C:${CmdID}:DATA$(SP)DELETE$(SP)DiffTimezoneVS$(SP)$(Cond)
```

The client uploads the execution result:

```
ID=${CmdID}&return=${XXX}&CMD=DATA
```

Annotation

Parameter	Description
DiffTimezoneVS	The table name, which is the table of verification modes in different time

	periods.
$(Cond)	TimezoneID=x means deleting the verification mode in different time periods of which TimezoneID is x. * means deleting all the verification modes in different time periods.

Example

Delete the verification mode in different time periods of which TimezoneID is 1:

```
C:342:DATA DELETE DiffTimezoneVS TimezoneID=1
```

The client returns the execution result:

```
ID=342&return=0&CMD=DATA
```

Door's Verification Modes in Different Time Periods

Application scenario

It is generally used to delete a door's verification modes in different time periods or all the doors' verification modes in different time periods.

Format

The server delivers the command:

```
C:${CmdID}:DATA$(SP)DELETE$(SP)DoorVSTimezone$(SP)$(Cond)
```

The client uploads the execution result:

```
ID=${CmdID}&return=${XXX}&CMD=DATA
```

Annotation

Parameter	Description
DoorVSTimezone	The table name, which is the table of the door's verification modes in different time periods.
$(Cond)	DoorID=x means deleting the verification modes in different time periods of the door of which DoorID is x. * means deleting all the doors' verification modes in different time periods.

Example

Delete the verification modes in different time periods of the door of which DoorID is 1:

```
C:342:DATA DELETE DoorVSTimezone DoorID=1
```

The client returns the execution result:

```
ID=342&return=0&CMD=DATA
```

User's Verification Modes in Different Time Periods

Application scenario

It is generally used to delete a user's verification modes in different time periods or all the users' verification modes in different time periods.

Format

The server delivers the command:

```
C:${CmdID}:DATA$(SP)DELETE$(SP)PersonalVSTimezone$(SP)$(Cond)
```

The client uploads the execution result:

```
ID=${CmdID}&return=${XXX}&CMD=DATA
```

Annotation

Parameter	Description
PersonalVSTimezone	The table name, which is the table of the user's verification modes in different time periods.
$(Cond)	PIN=x means deleting the verification mode in different time periods of a user of which PIN is x. PIN=x and DoorID=x mean deleting the verification mode in different time periods of a user of which PIN is x and DoorID is x. * means deleting all the users' verification modes in different time periods.

Example

Delete the verification modes in different time periods of the user of which PIN is 1 or PIN and DoorID are 1:

```
C:342:DATA DELETE PersonalVSTimezone PIN=1
C:343:DATA DELETE PersonalVSTimezone PIN=1  DoorID=1
```

The client returns the execution result:

```
ID=342&return=0&CMD=DATA
ID=343&return=0&CMD=DATA
```

Super User Privilege

Application scenario

It is generally used to delete the privilege of a super user or all the super users.

Format

The server delivers the command:

```
C:${CmdID}:DATA$(SP)DELETE$(SP)SuperAuthorize$(SP)$(Cond)
```

The client uploads the execution result:

```
ID=${CmdID}&return=${XXX}&CMD=DATA
```

Annotation

Parameter	Description
SuperAuthorize	The table name, which is the super user privilege table.
$(Cond)	Pin=x means deleting the privilege of the super user of which Pin is x. * means deleting the privilege of all the super users.

Example

Delete the privilege of the super user of which Pin is 1:

```
C:342:DATA DELETE SuperAuthorize Pin=1
```

The client returns the execution result:

```
ID=342&return=0&CMD=DATA
```

Comparison Photo (supported by visible light devices only)

Application scenario

It is generally used to delete the user comparison photo.

Format

The server delivers the command:

```
C:${CmdID}:DATA${SP}DELETE${SP}biophoto${SP}PIN=${XXX}
```

The client uploads the execution result:

```
ID=${CmdID}&return=${XXX}&CMD=DATA
```

Annotation

Parameter	Description
PIN=${XXX}	User ID.

User Photo

Application scenario

It is used to delete the user photo.

Format

The server delivers the command:

```
C:${CmdID}:DATA${SP}DELETE${SP}userpic${SP}pin=${XXX}
```

The client uploads the execution result:

```
ID=${CmdID}&return=${XXX}&CMD=DATA
```

Annotation

Parameter	Description
pin=${XXX}	User ID.

Unified Template

Application scenario

It is used to delete the united template.

Format

The server delivers the command:

```
C:${CmdID}:DATA${SP}DELETE${SP}biodata${SP}Type=${XXX}
C:${CmdID}:DATA${SP}DELETE${SP}biodata${SP}Type=${XXX}${HT}Pin=${XXX}${HT}No=${XXX}
```

The client uploads the execution result:

```
ID=${CmdID}&return=${XXX}&CMD=DATA
```

Annotation

<table>
<tr><th>Parameter</th><th>Description</th></tr>
<tr><td>Type=${XXX}</td><td>The biometric type
<table>
<tr><th>Value</th><th>Meaning</th></tr>
<tr><td>0</td><td>General</td></tr>
<tr><td>1</td><td>Fingerprint</td></tr>
<tr><td>2</td><td>Face (Near Infrared)</td></tr>
<tr><td>3</td><td>Voice Print</td></tr>
<tr><td>4</td><td>Iris</td></tr>
</table></td></tr>
</table>

<table>
<tr><td></td><td><table>
<tr><td>5</td><td>Retina</td></tr>
<tr><td>6</td><td>Palm Print</td></tr>
<tr><td>7</td><td>Finger Vein</td></tr>
<tr><td>8</td><td>Palm</td></tr>
<tr><td>9</td><td>Visible Light Face</td></tr>
<tr><td>10</td><td>Visible Light Palm</td></tr>
</table></td></tr>
<tr><td>Pin=${XXX}</td><td>User ID.</td></tr>
<tr><td>No=${XXX}</td><td>The number of specific biont. The default value is 0.<table>
<tr><td>Fingerprint</td><td>The number ranges from 0 to 9, corresponding to the little finger, ring finger, middle finger, index finger, and thumb of the left hand, and thumb, index finger, middle finger, ring finger, and little finger of the right hand respectively.</td></tr>
<tr><td>Finger Vein</td><td>It is same as of the fingerprint.</td></tr>
<tr><td>Face</td><td>All is 0.</td></tr>
<tr><td>Iris</td><td>0 is the iris of the left eye and 1 is the iris of the right eye.</td></tr>
<tr><td>Palm</td><td>0 is the palm of the left hand and 1 is the palm of the right hand.</td></tr>
<tr><td>Visible Light Palm</td><td>0 is the palm of the left hand and 1 is the palm of the right hand.</td></tr>
</table></td></tr>
</table>

Contact List Basic Information

Application scenario

It is used to delete the contact list basic information. When deleting basic information, the corresponding information in the contact number list needs to be deleted together.

Format

The server delivers the command:

```
C:${CmdID}:DATA$(SP)DELETE$(SP)contact_info$(SP)$(Cond)
```

The client uploads the execution result:

```
ID=${CmdID}&return=${XXX}&CMD=DATA
```

Annotation

Parameter	Description
contact_info	Table name, referring to the basic information of the contact list.
$(Cond)	Commonly used "ContactID=x" to delete contact information with contact ID x, or "*" to delete all data.
Special note: Generally, when a server deletes basic information from the contact list, it needs to delete the corresponding information from the contact number list.	

Example

Delete contact information with ContactID=1:

```
C:200:DATA DELETE contact_info ContactID=1
```

The client returns the execution result:

```
ID=200&return=0&CMD=DATA
```

Contact Number List

Application scenario

It is used to delete the contact number.

Format

The server delivers the command:

```
C:${CmdID}:DATA$(SP)DELETE$(SP)contact_number_list$(SP)$(Cond)
```

The client uploads the execution result:

```
ID=${CmdID}&return=${XXX}&CMD=DATA
```

Annotation

Parameter	Description
contact_number_list	Table name, referring to the the contact number list information.
$(Cond)	Commonly used "ContactID=x" to delete contact information with contact ID x, or "*" to delete all data.

Example

Delete contact number with ContactID=1:

```
C:200:DATA DELETE contact_number_list ContactID=1
```

Delete contact information with ContactID=1 and NumType=1:

```
C:200:DATA DELETE contact_number_list ContactID=1  NumType=1
```

The client returns the execution result:

```
ID=200&return=0&CMD=DATA
```

Contact Number Rule List

Application scenario

It is used to delete the contact list rule.

Format

The server delivers the command:

```
C:${CmdID}:DATA$(SP)DELETE$(SP)contact_number_pattern_list$(SP)$(Cond)
```

The client uploads the execution result:

```
ID=${CmdID}&return=${XXX}&CMD=DATA
```

Annotation

Parameter	Description
contact_number_pattern_list	Table name, referring to the the contact number rule information.
$(Cond)	Commonly used "PatternID=x" to delete contact information with number rule ID x, or "*" to delete all data.

Example

Delete contact number rule with PatternID=1:

```
C:200:DATA DELETE contact_number_pattern_list PatternID=1
```

The client returns the execution result:

```
ID=200&return=0&CMD=DATA
```

9.1.3 Count

Application scenario

It is used to count the number of records in a specified table or the number of records that meet the specified condition in a specified table.

Format

The server delivers the command:

```
C:${CmdID}:DATA$(SP)COUNT$(SP)$(TableName)$(SP)$(Cond)
```

The client uploads the statistical result:

```
POST
/iclock/querydata?SN=$(SerialNumber)&type=count&cmdid=${CmdID}&tablename=
user&count=${XXX}&packcnt=${XXX}&packidx=${XXX} HTTP/1.1
```

…

```
$(TableName)=${XXX}
```

The client uploads the execution result:

```
ID=${CmdID}&Return=${XXX}&CMD=DATA
```

Annotation

Parameter	Description
$(Cond)	The filter condition list. Condition field name1 = value1 Condition field name2 = value2 Condition field name3 = value3 When the filter condition list is x, the whole table is counted. When

	the list contains multiple filter conditions, the conditions are in AND relationship.
/iclock/querydata	The URI of the data that is uploaded by the client and meets the condition.
type=count:	The type of data to be uploaded by the client is statistics.
count	The number of data entries returned.
packcnt	The total number of packages. When there is a great amount of data, the data needs to be divided into different packages. This value indicates how many packages the data is divided into.
packidx	The package ID, which indicates which of all packages is currently being sent.
$(TableName)=${XXX}	The left parameter indicates the table name, and the right value indicates the number of data entries in the table that meet the condition.
COUNT	The sub-command COUNT contains three interaction processes: The client gets the command, the server returns the COUNT command, the client returns the statistical result, the server returns OK, the client returns the command execution result, and then the server returns OK. (Only the key points are listed below).

User Count

Application scenario

It is generally used to get the number of users of the device.

Format

The server delivers the command:

```
C:${CmdID}:DATA$(SP)COUNT$(SP)user$(SP)$(Cond)
```

The client uploads the statistical result:

```
user=${XXX}
```

The client uploads the execution result:

```
ID=${CmdID}&return=${XXX}&CMD=DATA
```

Annotation

Parameter	Description
user	The table name, which is the number of obtained user information tables.
$(Cond)	Generally no condition is set, which means getting the number of all users

Example

The server delivers the command:

```
C:288:DATA COUNT user
```

The client uploads the statistical result:

```
POST
/iclock/querydata?SN=3383154200002&type=count&cmdid=288&tablename=user&co
unt=1&packcnt=1&packidx=1 HTTP/1.1
Cookie: token=af65a75608cf5b80fbb3b48f0b4df95a
Host: 192.168.213.17:8088
Content-Length: 6
…
user=5
```

The client uploads the execution result:

```
ID=288&return=0&CMD=DATA COUNT
```

Access Control Record Count

Application scenario

It is used to get the number of access control records from the client.

Note

The interaction process and command format are the same as those of the command to get the user count. You only need to replace the table name with "transaction".

Fingerprint Template Count

Application scenario

It is used to get the number of fingerprint templates from the client.

Note

The interaction process and command format are the same as those of the command to get the user count. You only need to replace the table name with "transaction".

Comparison Photo Count (supported by visible light devices only)

Application scenario

It is generally used to get the number of comparison photos from the device.

Format

The server delivers the command:

```
C:${CmdID}:DATA$(SP)COUNT$(SP)biophoto$(SP)Type=$(XXX)$(HT)$(Cond)
```

The client uploads the statistical result:

```
biophoto=${XXX}
```

The client uploads the execution result:

```
ID=${CmdID}&return=${XXX}&CMD=DATA COUNT
```

Annotation

<table>
<tr><th>Parameter</th><th>Description</th></tr>
<tr><td>biophoto</td><td>The table name, which is the number of obtained comparison photos.</td></tr>
<tr><td>Type=${XXX}</td><td>The biometric type
<table>
<tr><th>Value</th><th>Meaning</th></tr>
<tr><td>0</td><td>General</td></tr>
<tr><td>1</td><td>Fingerprint</td></tr>
</table>
</td></tr>
</table>

	2 Face (Near Infrared) 3 Voice Print 4 Iris 5 Retina 6 Palm Print 7 Finger Vein 8 Palm 9 Visible Light Face
$(Cond)	Generally, no condition is set, which means getting the number of all comparison photos.

Example

The server delivers the command:

Does not support hybrid identification protocol, only supports the number of visible light face comparison photos:

```
C:80:DATA COUNT biophoto
```

After supporting the hybrid identification protocol, you must specify the type of comparison photo.

```
C:80:DATA COUNT biophoto Type=9
```

The client uploads the statistical result:

```
POST
/iclock/querydata?SN=1809140005&type=count&cmdid=80&tablename=biophoto&co
unt=1&packcnt=1&packidx=1 HTTP/1.1

Cookie: token=af65a75608cf5b80fbb3b48f0b4df95a

Host: 192.168.213.17:8088

Content-Length: 10

…

biophoto=2
```

The client uploads the execution result:

```
ID=80&Return=0&CMD=DATA COUNT
```

Unified Template Count

Application scenario

It is generally used to get the number of device unified templates of the specified type.

Format

The server delivers the command:

```
C:${CmdID}:DATA$(SP)COUNT$(SP)biodata$(SP)Type=$(XXX)$(HT)$(Cond)
```

The client uploads the statistical result:

```
biodata=${XXX}
```

The client uploads the execution result:

```
ID=${CmdID}&return=${XXX}&CMD=DATA COUNT
```

Annotation

<table>
<tr><th>Parameter</th><th>Description</th></tr>
<tr><td>biophoto</td><td>The table name, which is the number of obtained comparison photos.</td></tr>
<tr><td>Type=${XXX}</td><td>The biometric type
<table>
<tr><th>Value</th><th>Meaning</th></tr>
<tr><td>0</td><td>General</td></tr>
<tr><td>1</td><td>Fingerprint</td></tr>
<tr><td>2</td><td>Face (Near Infrared)</td></tr>
<tr><td>3</td><td>Voice Print</td></tr>
<tr><td>4</td><td>Iris</td></tr>
<tr><td>5</td><td>Retina</td></tr>
<tr><td>6</td><td>Palm Print</td></tr>
<tr><td>7</td><td>Finger Vein</td></tr>
<tr><td>8</td><td>Palm</td></tr>
<tr><td>9</td><td>Visible Light Face</td></tr>
<tr><td>10</td><td>Visible Light Palm</td></tr>
</table></td></tr>
</table>

$(Cond)	No condition is set, which means getting the number of all comparison photos.

Example

The server delivers the command:

```
C:80:DATA COUNT biodata
```

The client uploads the statistical result:

```
POST
/iclock/querydata?SN=1809140005&type=count&cmdid=80&tablename=biodata&cou
nt=1&packcnt=1&packidx=1 HTTP/1.1

Cookie: token=af65a75608cf5b80fbb3b48f0b4df95a

Host: 192.168.213.17:8088

Content-Length: 10

……

biodata=2
```

The client uploads the execution result:

```
ID=80&Return=0&CMD=DATA COUNT
```

9.1.4 Query

Application scenario

The server actively requires the client to upload data that meets the query condition.

Format

The server delivers the command:

```
C:${CmdID}:DATA$(SP)QUERY$(SP)tablename=$(XXX),fielddesc=$(XXX),filter=$(
XXX)
```

The client uploads the data that meets the condition:

```
POST
/iclock/querydata?SN=$(SerialNumber)&type=tabledata&cmdid=$(XXX)&tablenam
e=$(TableName)&count=$(XXX)&packcnt=$(XXX)&packidx=$(XXX) HTTP/1.1
```

```
...
$(DataRecord)
```

Response from the server:

```
$(TableName)=$(XXX)
```

The client uploads the execution result:

```
ID=${CmdID}&Return=${XXX}&CMD=DATA
```

Annotation

Parameter	Description
tablename	It is the name of the table to query.
fielddesc	It is a field. When this field is set to *, all fields of the table are queried; otherwise, the specified field is queried.
filter	It is the query condition. When this field is set to *, the data is not filtered. When this field is set to NewRecord and tablename is transaction, new records are queried. On some devices, when this field is set to starttime=\tendtime=, records in the specified time period are queried.
/iclock/querydata	The URI of the data that is uploaded by the client and meets the condition.
type	The data type. If it is set to tabledata in the QUERY command, it means the data in the table.
count	The number of data entries returned.
packcnt	The total number of packages. When there is a great amount of data, the data needs to be divided into different packages. This value indicates how many packages the data is divided into.
packidx	The package ID, which indicates which of all packages is currently being sent.
$(DataRecord)	The data that meets the condition and needs to be uploaded. The specific format depends on the table name.

return	The returned value is the quantity of this query.

User

Application scenario

The server actively requires the client to upload the specified user. It is generally used to get all the users from the device.

Format

The server delivers the command:

```
C:415:DATA$(SP)QUERY$(SP)tablename=user,fielddesc=$(XXX),filter=$(XXX)
```

The client uploads the data:

```
POST
/iclock/querydata?SN=$(SerialNumber)&type=tabledata&cmdid={CmdID}&tablena
me=user&count=$(XXX)&packcnt=$(XXX)&packidx=$(XXX) HTTP/1.1

...

$(DataRecord)
```

Response from the server:

```
user=$(XXX)
```

The client uploads the successful execution result:

```
ID=${CmdID}&return=${XXX}&CMD=DATA COUNT
```

Annotation

Parameter	Description
tablename	When it is set to user, it means the user table and the data processed is the user data.
$(DataRecord)	When the table name is user, the data format is the user data format. For

	details, see Upload User Information.

Example

The server delivers the command:

```
C:415:DATA QUERY tablename=user,fielddesc=*,filter=*
```

The client uploads the data of three users:

```
POST
/iclock/querydata?SN=3383154200002&type=tabledata&cmdid=415&tablename=use
r&count=3&packcnt=1&packidx=1 HTTP/1.1

…

user uid=1  cardno= pin=1   password=   group=1 starttime=0 endtime=0   name=
privilege=0 disable=0   verify=0

user uid=2  cardno= pin=2   password=1  group=1 starttime=0 endtime=0   name=
privilege=0 disable=0   verify=0

user uid=3  cardno= pin=3   password=   group=1 starttime=0 endtime=0   name=
privilege=0 disable=0   verify=0
```

The server responds that it has received the data of the three user:

```
user=3
```

The client uploads the successful execution result:

```
ID=415&return=3&CMD=DATA QUERY
```

Fingerprint Template

Application scenario

The server actively requires the client to upload the specified fingerprint template. It is generally used to get all the fingerprint templates after getting all the users.

Note

The command is the same as the command to query users, except that you need to replace the table name with templatev10. The `$(DataRecord)` format is the fingerprint template format. For details, see Upload Fingerprint Template.

Access Control Record

Application scenario

The server actively requires the client to upload the specified access control record. It is generally used to get all access control records or the access control record in the specified range after the ACCOUNT command failed.

Format

The server delivers the command:

```
C:415:DATA$(SP)QUERY$(SP)tablename=transaction,fielddesc=$(XXX),filter=$(XXX)
```

The client uploads the data:

```
POST /iclock/querydata?SN=$(SerialNumber)&type=tabledata&cmdid={CmdID}&tablename=transaction&count=$(XXX)&packcnt=$(XXX)&packidx=$(XXX) HTTP/1.1

…

$(DataRecord)
```

Response from the server:

```
transaction=$(XXX)
```

The client uploads the successful execution result:

```
ID=${CmdID}&return=$(XXX)&CMD=DATA
```

Annotation

Parameter	Description
The command format is same to the format of the command used to query users. You need to replace the table name with transaction.	
$(DataRecord)	Its format is same to the format of the access control record. `transaction$(SP)cardno=$(XXX)$(HT)pin=$(XXX)$(HT)verified=$(XXX)$(HT)doorid=$(XXX)$(HT)eventtype=$(XXX)$(HT)inoutstate=$(XXX)$(HT)time_second=$(XXX)$(HT)index=$(XXX)$(HT)sitecode=$(XXX)$(HT)devid=$(XXX)$(HT)maskflag=$(xxx)${HT}temperature=$(xxx)${HT}convt`

	emperature=$(xxx)
transaction	The table name, which means that the data type is access control record.
cardno	Card number
pin	User ID
verified	The verification mode code. For details, see Appendix 3
doorid	Door ID
eventtype	Event type
inoutstate	The in/out status. 0 indicates In, and 1 indicates Out.
time_second	The time stamp of the record, for example, time_second=583512660, in which time_second is calculated according to the algorithm in Appendix 5, And the specific time yyyy-mm-dd-hh-mm-ss is calculated according to the method in Appendix 6.
index	The ID of the access control record.
sitecode	Site code
devid	Device ID
maskflag	Its value is either 0 or 1: 0 Not wearing a mask 1 Wearing a mask
temperature	The value is the temperature data with a decimal point, for example: 36.2.
convtemperature	The value is the temperature data with a decimal point. If the server does not send the IRTempUnitTrans parameter, then the unit of the temperature upload is subject to the IRTempUnit parameter.
${LF} is used to connect multiple records.	
Note: When the filter is set to NewRecord, new records that are not uploaded are queried	

Example

The server delivers the command:

```
C:415:DATA QUERY tablename=user,fielddesc=*,filter=*
```

The client uploads the access control record:

```
POST
/iclock/querydata?SN=3383154200002&type=tabledata&cmdid=415&tablename=tra
nsaction&count=1&packcnt=1&packidx=1 HTTP/1.1

…

transaction cardno=0    pin=0   verified=200    doorid=1    eventtype=100
inoutstate=2    time_second=548003688   index=92    sitecode=0  devid=1
sn=3383154200002
```

The server responds that it has received the data of the three user:

```
transaction=1
```

The client uploads the successful execution result:

```
ID=415&return=1&CMD=DATA
```

Wi-Fi List

Application scenario

The software delivers a command to query the Wi-Fi list. After searching for surrounding Wi-Fis, the deice uploads the SSIDs and other information to the software.

Format

The server delivers the command:

```
C:415:DATA$(SP)QUERY$(SP)tablename=[APList],fielddesc=$(XXX),filter=$(XXX
)
```

The client uploads the data:

```
POST
/iclock/querydata?SN=$(SerialNumber)&type=vmtable&cmdid={CmdID}&tablename
=APList&count=$(XXX)&packcnt=$(XXX)&packidx=$(XXX) HTTP/1.1

…

$(DataRecord)
```

Response from the server:

```
APList=$(XXX)
```

The client uploads the successful execution result:

```
ID=${CmdID}&return=$(XXX)&CMD=DATA
```

Annotation

Parameter	Description
tablename	[APList], which is a Wi-Fi virtual table.
$(DataRecord)	The Wi-Fi data format is as follows. APList$(SP)ecn=$(XXX)$(HT)ssid=$(XXX)$(HT)rssi=$(XXX) $(HT)mac=$(XXX)
APList	A Wi-Fi virtual table.
ecn	Direct congestion flag
ssid	Wi-Fi user name
rssi	The signal
mac	MAC address

Example

The server delivers the command:

```
C:415:DATA QUERY tablename=[APList],fielddesc=*,filter=*
```

The client uploads the access control record:

```
POST
/iclock/querydata?SN=$(SerialNumber)&type=vmtable&cmdid=415&tablename=APL
ist&count=%d&packcnt=%d&packidx=%d HTTP/1.1

Cookie: token=${XXX}

Host: ${ServerIP}:${ServerPort}

Content-Length: ${XXX}

…

APList ecn=%?\tssid=%?\trssi=%?\tmac=%?

APList ecn=%?\tssid=%?\trssi=%?\tmac=%?

APList ecn=%?\tssid=%?\trssi=%?\tmac=%?
```

Response from the server:

```
APList=3
```

The client uploads the successful execution result:

```
ID=415&return=3&CMD=DATA
```

Comparison Photo (supported by visible light devices only)

Application scenario

The server actively requires the client to upload the specified comparison photo. It is generally used to get all the comparison photos from the device.

Format

The server delivers the command:

Does not support hybrid identification protocol, only supports obtaining visible light face comparison photos:

```
C:99:DATA$(SP)QUERY$(SP)tablename=biophoto,fielddesc=$(XXX),filter=$(XXX)
```

After supporting the hybrid identification protocol, you must specify the type of comparison photo:

```
C:99:DATA$(SP)QUERY$(SP)tablename=biophoto,fielddesc=$(XXX),filter=Type=$(XXX)$(HT)$(XXX)
```

Type=${XXX}: The biometric type

Value	Meaning
0	General
1	Fingerprint
2	Face (Near Infrared)
3	Voice Print
4	Iris
5	Retina
6	Palm Print
7	Finger Vein
8	Palm

9	Visible Light Face

The client uploads the data:

```
POST
/iclock/querydata?SN=$(SerialNumber)&type=tabledata&cmdid={CmdID}&tablena
me=biophoto&count=$(XXX)&packcnt=$(XXX)&packidx=$(XXX) HTTP/1.1

…

$(DataRecord)
```

Response from the server:

```
biophoto=$(XXX)
```

The client uploads the successful execution result:

```
ID=${CmdID}&return=$(XXX)&CMD=DATA QUERY
```

Annotation

Parameter	Description
tablename	When it is set to biophoto, it means the comparison photo table and the data processed is the comparison photo data.
$(DataRecord)	When the table name is biophoto, the data format is the format of the comparison photo data. For details, see Upload Comparison Photo.

Example

The server delivers the command:

```
C:99:DATA QUERY tablename=biophoto,fielddesc=*,filter=*
```

The client uploads the data of two comparison photos:

```
POST
/iclock/querydata?SN=1809140005&type=tabledata&cmdid=99&tablename=biophot
o&count=1&packcnt=2&packidx=1   HTTP/1.1

…

biophoto pin=123    filename=123.jpg    type=9  size=95040
content=AAAA.......
```

The server responds that it has received the data of one comparison photo:

```
biophoto=1

POST
/iclock/querydata?SN=1809140005&type=tabledata&cmdid=99&tablename=biophot
o&count=1&packcnt=2&packidx=2

HTTP/1.1

…

biophoto pin=124    filename=124.jpg    type=9  size=95040
content=AAAA.......
```

The server responds that it has received the data of one comparison photo:

```
biophoto=1
```

After the client uploads all the comparison photos, the result is returned.

```
ID=99&Return=2&CMD=DATA QUERY
```

Unified Template

Application scenario

The server actively requires the client to upload the specified unified template data. It is generally used to get the unified template data of the device specified type.

Format

The server delivers the command:

Does not support hybrid identification protocol, only supports obtaining visible light face unified templates:

```
C:99:DATA$(SP)QUERY$(SP)tablename=biodata,fielddesc=$(XXX),filter=$(XXX)
```

After supporting the hybrid identification protocol, you must specify the type of unified templates:

```
C:99:DATA$(SP)QUERY$(SP)tablename=biodata,fielddesc=$(XXX),filter=Type=$(
XXX)$(HT)$(XXX)
```

Type=${XXX}: The biometric type

Value	Meaning
0	General
1	Fingerprint

2	Face (Near Infrared)
3	Voice Print
4	Iris
5	Retina
6	Palm Print
7	Finger Vein
8	Palm
9	Visible Light Face
10	Visible Light Palm

The client uploads the data:

```
POST
/iclock/querydata?SN=$(SerialNumber)&type=tabledata&cmdid={CmdID}&tablena
me=biodata&count=$(XXX)&packcnt=$(XXX)&packidx=$(XXX) HTTP/1.1

...

$(DataRecord)
```

Response from the server:

```
biophoto=$(XXX)
```

The client uploads the successful execution result:

```
ID=${CmdID}&return=$(XXX)&CMD=DATA QUERY
```

Annotation

Parameter	Description
`tablename`	When it is set to biodata, it means the unified template table and the data processed is the unified template data.
`$(DataRecord)`	When the table name is biodata, the data format is the format of the unified template data. For details, see Upload Unified Template.

Example

The server delivers the command:

```
C:99:DATA QUERY tablename=biodata,fielddesc=*,filter=Type=9
```

The client uploads the data of two comparison photos:

```
POST
/iclock/querydata?SN=1809140005&type=tabledata&cmdid=99&tablename=biodata
&count=1&packcnt=2&packidx=1   HTTP/1.1

……

biodata pin=1 no=0 index=0 valid=1 duress=0 type=1 majorver=10 minorver=0
format=0 tmp=apUBD.......
```

The server responds that it has received the data of one comparison photo:

```
biophoto=1
```

```
POST
/iclock/querydata?SN=1809140005&type=tabledata&cmdid=99&tablename=biodata
&count=1&packcnt=2&packidx=2

HTTP/1.1

…

biodata pin=123 no=0 index=0 valid=1 duress=0 type=1 majorver=10 minorver=0
format=0 tmp=AAAAA.......
```

The server responds that it has received the data of one unified template:

```
biophoto=1
```

After the client uploads all the unified templates, the result is returned.

```
ID=99&Return=2&CMD=DATA QUERY
```

The server specifies the query condition:

Query conditions must include Type, different query conditions are separated by {HT}.

Query visible light face template of User PIN = 1

```
C:99:DATA QUERY tablename=biodata,fielddesc=\*,filter=Type=9\tPin=1
```

Query fingerprint template of the right index finger of User PIN = 1

```
C:99:DATA QUERY tablename=biodata,fielddesc=\*,filter=Type=1\tPin=1\tNo=6
```

9.2 Account

This command is currently used to upload the access control records for software account checking only on the server. Account checking means that the software compares its access control records with those uploaded by the device, to check whether any access control record is missing.

9.2.1 Pull SDK Device

Format

The server delivers the command:

```
C:${CmdID}:ACCOUNT$(SP)transaction$(SP)ContentType=tgz$(SP)MaxIndex=0
```

The client uploads the data:

```
POST
/iclock/querydata?SN=$(SerialNumber)&type=tabledata&cmdid=${CmdID}&tablen
ame=transaction&count=${XXX}&datafmt=3&Stamp=9999

...

name=transaction.tgz

size=${XXX}

datacode=base64

datatype=tgz$(NULL)$(DataRecord)
```

Response from the server:

```
transaction=$(XXX)
```

The client uploads the successful execution result:

```
ID=${CmdID}&Return=${XXX}&CMD=ACCOUNT&transaction&StartTime=${XXX}&EndTim
e=${XXX}&RecordSum=${XXX}
```

Annotation

Parameter	Description
transaction	The table name, which is the access control record.
ContentType	The text format. tgz means a compressed package.
MaxIndex	Access control records of which the ID is greater than this value need

	to be uploaded. When the value is 0, all access control records need to be uploaded.
/iclock/querydata	The URI of the access control record that is uploaded by the client and meets the condition.
type	The type of the data to upload. If it is set to tabledata in the ACCOUNT command, it means the data in the table.
tablename	The table name. Currently, the ACCOUNT command is used only for the access control records, and therefore all the table names in the ACCOUNT command is transaction.
count	The number of data entries uploaded.
datafmt	The format of the data to upload. Currently, it is fixed to 3 in this command, which means a compressed package.
Stamp	The timestamp. Currently, it is fixed to 9999 in this command.
name	The name of the compressed package. transaction.tgz means a compressed package of access control records.
size	The size of the compressed package.
datacode	The data format used when the compressed package is uploaded. base64 means that the compressed package is base64-encoded before upload.
datatype	The data type. tgz means a compressed package.
$(DataRecord)	Multiple access control records need to be packaged into a compressed package and then base64-encoded before upload. For the format details, see Upload Access Control Record.
transaction=${XXX}	${XXX} is the number of access control records uploaded by the client.
return	The number of access control records that meet the condition.
CMD	It is fixed to ACCOUNT in the ACCOUNT command.
StartTime	The start time of the access control record. It is reserved currently and left blank during upload.
EndTime	The end time of the access control record. It is reserved currently and left blank during upload.
RecordSum	The number of access control records that meet the condition.

Example

Upload all access control records in the form of a compressed package:

The server delivers the command:

```
C:291:ACCOUNT transaction ContentType=tgz MaxIndex=0
```

The client uploads the data consisting of 41 access control records:

```
POST
/iclock/querydata?SN=$(SerialNumber)&type=tabledata&cmdid=291&tablename=t
ransaction&count=41&datafmt=3&Stamp=9999

…

name=transaction.tgz

size=636

datacode=base64

datatype=tgz\0 [base64 data stream]
```

Response from the server:

```
transaction=41
```

The client uploads the command execution result:

```
ID=291&Return=41&CMD=ACCOUNT&transaction&StartTime=&EndTime=&RecordSum=41
```

9.2.2 Controller

Format

The server delivers the following three types of commands:

```
C:${CmdID}:ACCOUNT$(SP)transaction$(SP)MaxIndex=${XXX}

C:${CmdID}:ACCOUNT$(SP)transaction$(SP)StartIndex=${XXX}$(HT)EndIndex=${X
XX}

C:${CmdID}:ACCOUNT$(SP)transaction$(SP)StartTime=${XXX}$(HT)EndTime=${XXX
}
```

The client uploads the data:

First, the device uploads the summary:

URL:

```
POST /iclock/cdata?SN=$(SerialNumber)&cmdid=${CmdID}&desc=overview
HTTP/1.1
```

Content:

```
SumCount=${XXX}

FileName=${XXX}

ContentType=${XXX}

FileCount=${XXX}
```

Response from the server:

```
OK
```

Then, the device uploads the access control records as files:

URL:

```
POST
/iclock/file?SN=$(SerialNumber)&cmdid=${CmdID}&fileseq=${XXX}&contentty
pe=tgz&table=transaction&count=$(XXX) HTTP/1.1
```

Content:

The content of transaction.tgz.

The following describes how transaction.tgz is generated:

a. Convert the count number of access control records into those in the push upload format and store them in transaction.txt, for example:

```
cardno=0    pin=0   verified=200    doorid=1    eventtype=100
inoutstate=2    time_second=548003687   index=92

cardno=0    pin=0   verified=200    doorid=1    eventtype=100
inoutstate=2    time_second=548003688   index=93

cardno=0    pin=0   verified=200    doorid=1    eventtype=100
inoutstate=2    time_second=548003689   index=94
```

b. Convert transaction.txt to transaction.tgz.

The server returns:

```
OK
```

The client uploads the execution result:

```
ID=${CmdID}&Return=${XXX}&CMD=ACCOUNT
```

Annotation

Parameter	Description
transaction	The table name, which is the access control record.
MaxIndex	Access control records of which the ID is greater than this value need to be uploaded. When the value is 0, all access control records need to be uploaded.
StartIndex	Need to upload access control records of which the ID is greater than or equal to this value.
EndIndex	Need to upload access control records of which the ID is smaller than or equal to this value.
StartTime	Need to upload access control records of which the record time is greater than or equal to this value. The time format is XXXX-XX-XX XX:XX:XX, for example, 2018-02-22 16:19:20.
EndTime	Need to upload access control records of which the record time is smaller than or equal to this value. The time format is XXXX-XX-XX XX:XX:XX, for example, 2018-02-22 16:19:20.
SumCount	The total number of access control records.
FileName	File name.
ContentType	The text format. tgz means a compressed package.
FileCount	The number of access control record sub-packages that meet the condition.
fileseq	The index of the file sub-packages to upload, which starts from 1.

9.3 Test Host

Application scenario

The software delivers a network test command. After receiving the command, the device test the connectivity based on the IP address and port provided and returns the test result.

Format

The server delivers the command:

```
C:295:Test Host Address=110.80.38.74:8092
```

The client uploads the execution result:

```
ID=${CmdID}&Return=${XXX}&CMD=Test(SP)Host
```

Annotation

Parameter	Description
Address	The IP address and port of the test server

9.4 Control Device

Application scenario

The control commands are used to control device startup, remote door opening, remote door closing, alarm canceling, enabling or disabling of Normal Open, and auxiliary outputs.

Format

The server delivers the command:

```
C:$(CmdID):CONTROL$(SP)DEVICE$(SP)AABBCCDDEE
```

The client uploads the execution result:

```
ID=${CmdID}&Return=${XXX}&CMD=CONTROL$(SP)DEVICE
```

Annotation

AA, BB, CC, and DD are four groups of two-byte strings. Each group of strings is the converted result of a one-byte integer after %02X conversion. Wherein, AA indicates control commands, which are described as follows:

AA	BB	CC	DD	EE
01: Control the output	00: Open all the doors. 01-10: The door ID.	01: Control the lock. 02: Control the auxiliary output.	00: Off. FF: Normal open. 01-FF: The opening duration.	Operator

02: Cancel the alarm	00: Cancel the alarms. 01-10: The door ID.			Operator
03: Restart the device	00: Restart the current device. 01-10: The slave device ID.			Operator
04: Control Normal Open	00: All the doors. 01-10: The door ID.	00: Disable. 01: Enable.		Operator
05: Obtain battery power				Operator
06: Lock/unlock the door	00: Open all the doors. 01-10: The door ID.	00: Unlock. 01: Lock.		Operator
07: Control emergency dual-on or dual-off	01: Emergency dual-on. 02: Emergency dual-off.			Operator
08: The Zigbee command	01: Re-networking. 02: Stop re-networking. 03: Allow access. 04: Stop access.			Operator
09: The Wiegand test command	01: Start the Wiegand test. 02: Stop the Wiegand test.			Operator
10: 485 The lighting command				Operator
11: The network switchover command	01: Switch from wired network to the 4G network. 02: Switch from the wired network to Wi-Fi. 03: Switch from 4G network to wired network. 04: Switch from Wi-Fi to wired network.			Operator
12: The command of registering by identity card.	01: Enable the mode of registering by identity card. 00: Disable the mode of	00: Open all the doors. 01-10: The	Timeout time.	Operator

	registering by identity card.	door ID.		
13: The command of querying the sub-device connection status.				Operator
14: The command of testing the switched network				Operator
15: Cancel the emergency state of elevator control				Operator
16: The command to reset the counter on the entrance control device	01: Reset the out counter 02: Reset the in counter 03: Reset the number of alarms 04: Reset all			Operator
17: Control the Output of elevator control	32 bytes, each bit represents a floor, and the floor expands from left to right		00: Off. FF: Normal open. 01-FF: The opening duration.	Operator
18: Control the elevator's normal opening	32 bytes, each bit represents a floor, and the floor expands from left to right		00: Off. FF: Normal open. 01-FF: The opening duration.	Operator

Floor 1
80000000000000000000000000000000000000000000000000000000000000000000000000000000
Floor 1,3,5,7,9
2A80000000000000000000000000000000000000000000000000000000000000000000000000000000

Example

1) The server delivers the command of opening Door 1 for 5s:

```
C:221:CONTROL DEVICE 01010105
```

2) The server delivers the command of canceling the alarm of Door 1:

```
C:222:CONTROL DEVICE 02010000
```

3) The server delivers the command of restarting the current device:

```
C:223:CONTROL DEVICE 03000000
```

4) The server delivers the Wiegand test command:

```
C:224:CONTROL DEVICE 0900000000
```

After receiving the command, the client uploads the request protocol:

```
POST /iclock/cdata?SN=$(SerialNumber)&table=testdata HTTP/1.1
Cookie: token=${XXX}
Host: ${ServerIP}:${ServerPort}
Content-Length: ${XXX}
…
type=wg\tbitscount=%?\tbits=%?\tsn=%?
```

The server returns:

```
OK
```

5) The server delivers the command of querying the sub-device connection status:

```
C:225:CONTROL DEVICE 0D000000
```

6) The server delivers the command of restarting the current device:

From wired network to 4G network:

```
C:226:CONTROL DEVICE 0B010000
```

From wired network to Wi-Fi:

```
C:227:SET OPTIONS TempWirelessSSID=11,TempWirelessKey=11
```

(TempWirelessSSID: Wi-Fi SSID, the Wi-Fi key)

```
C:228:CONTROL DEVICE 0B020000
```

From 4G network to wired network:

```
C:229:CONTROL DEVICE 0B030000
```

From Wi-Fi to wired network:

```
C:230:CONTROL DEVICE 0B040000
```

7) The server delivers the Wiegand test command:

```
ID=224&Return=0&CMD=CONTROL DEVICE
```

9.4.1 Upgrade

Remote Firmware Upgrade

Application scenario

It is used to remotely upgrade the firmware of the client device from the server software.

Method 1:

Application scenario

To remotely upgrade the client firmware, the client needs to be compatible with the controller and the new architecture PULL SDK Device. The files to upgrade must be converted by the server to the specified format and then delivered to the client.

Format

The server delivers the command:

```
C:${CmdID}:UPGRADE$(SP)checksum=${XXX},url=$(URL),size=${XXX}
```

The client downloads the upgrade package from the URL carried in the command:

```
GET /iclock/file?SN=$(SerialNumber)&url=$(URL) HTTP/1.1
```

...

The client uploads the execution result:

```
ID=${CmdID}&Return=${XXX}&CMD=UPGRADE
```

Annotation

Parameter	Description
Checksum	The MD5 checksum.
url	The address to download the upgrade file named emfw.cfg.
size	The size of the original file.
Note: In this method, the firmware upgrade file is base64-encoded by the server before delivery. After receiving the file, the client needs to convert it into a binary file and name it as emfw.cfg.	

Example

The server delivers the firmware upgrade command:

```
C:384:UPGRADE
checksum=a5bf4dcd6020f408589224274aab132d,url=http*//localhost*8088\firew
are\F20\admin\emfw.cfg,size=2312
```

The client sends a request to download the upgrade package:

```
GET
/iclock/file?SN=3383154200002&url=http://192.168.213.17:8088/fireware/F20
/admin/emfw.cfg HTTP/1.1
Cookie: token=af65a75608cf5b80fbb3b48f0b4df95a
Host: 192.168.213.17:8088
…
```

The client uploads the successful execution result:

```
ID=384&Return=0&CMD=UPGRADE
```

Method 2:

Application scenario

The client directly obtains the upgrade file to upgrade the firmware remotely, without converting the file format.

Format

The server delivers the command:

```
C:${CmdID}:UPGRADE$(SP)type=1,checksum=${XXX},size=${XXX},url=$(URL)
```

The client sends a request to download the upgrade package:

```
GET /iclock/file?SN=$(SerialNumber)&url=$(URL) HTTP/1.1
Cookie: token=af65a75608cf5b80fbb3b48f0b4df95a
Host: 192.168.213.17:8088
…
```

The client uploads the execution result:

```
ID=${CmdID}&Return=${XXX}&CMD=UPGRADE
```

Annotation

Parameter	Description
type	1 means obtaining the upgrade file from the URL. Currently, only 1 is supported.
Checksum	The MD5 checksum.
url	The address to download the upgrade file named emfw.cfg.
size	The size of the upgrade package.
Note: In this method, the client directly obtains the firmware upgrade file, without the need of converting its format.	

Example

The server delivers the command:

```
C:123:UPGRADE
type=1,checksum=oqoier9883kjankdefi894eu,size=6558,url=http://192.168.0.13:89/data/emfw.cfg
```

The client sends a request to download the upgrade package:

```
GET /iclock/file?SN=3383154200002&url=http://192.168.0.13:89/data/emfw.cfg HTTP/1.1
Cookie: token=af65a75608cf5b80fbb3b48f0b4df95a
Host: 192.168.0.13:89
...
```

The client uploads the successful execution result:

```
ID=384&Return=0&CMD=UPGRADE
```

Method 3:

Application scenario

Remotely upgrade the client's firmware, and obtain files in a subcontracted pull mode, without format conversion, and the client directly obtains the files. The subcontracting pull process refers to the Range protocol of HTTP, and the process is as follows:

1) The device side specifies RANGE in the HTTP Header: xx-xx specifies the byte range of the upgrade package to be pulled.
2) The software side parses the RANGE specified in the HTTP Header, and returns the upgrade package data in the specified range to the device side.
3) The device side pulls 1M of data each time by default, and the maximum one-time use of 1M memory, which solves the problem of insufficient memory caused by the device pulling large data packets at one time.

Format

The server delivers the command:

```
C:${CmdID}:UPGRADE$(SP)checksum=${XXX},size=${XXX},url=$(URL),supportsubcontracting=${XXX}
```

The client sends a request to download the upgrade package:

```
GET $(URL) HTTP/1.1
Cookie: token=af65a75608cf5b80fbb3b48f0b4df95a
Host: 192.168.213.17:8088
……
```

The client uploads the execution result:

```
ID=${CmdID}&Return=${XXX}&CMD=UPGRADE
```

Annotation

Parameter	Description
Checksum	The MD5 checksum.
url	Represents the download resource address of the upgrade file, which can support both absolute path and relative path (if it is a relative path, the device will automatically splice into an absolute path according to the address of the currently connected cloud server).
size	The size of the upgrade package.
supportsubcontracting	Represents whether the software supports the upgrade package subcontracting protocol (0: not supported, 1: supported)

> **Note:** In this method, the client directly obtains the firmware upgrade file, without the need of converting its format.

Example

The server delivers the command:

```
C:123:UPGRADE
checksum=oqoier9883kjankdefi894eu,size=6558,url=http://192.168.0.13:89/da
ta/emfw.cfg
```

The client sends a request to download the upgrade package:

```
GET http://192.168.0.13:89/data/emfw.cfg
Cookie: token=af65a75608cf5b80fbb3b48f0b4df95a
Host: 192.168.0.13:89
...
```

The client uploads the successful execution result:

```
ID=123&Return=0&CMD=UPGRADE
```

9.5 Configuration Class

These commands are used to configure the client or obtain the client configurations.

9.5.1 Set Options

Application scenario

It is used to set the client configuration parameters, such as the IP address, gateway, and lock driver time length.

Format

The server delivers the command:

```
C:$(CmdID):SET$(SP)OPTIONS$(SP)parameter-value list
```

The client uploads the execution result:

```
ID=$(CmdID)&Return=0&CMD=SET OPTIONS
```

Note

The parameter list is in the form of parameter name=parameter value. Multiple parameters are separated by commas, but the last parameter is not followed by a comma.

Example

Modify the IP address, gateway and mask:

```
C:403:SET OPTIONS
IPAddress=192.168.213.221,GATEIPAddress=192.168.213.1,NetMask=255.255.255
.0
```

Set the server IP address and port for the device:

```
C:399:SET OPTIONS WebServerIP=192.168.213.221,WebServerPort=8088
```

Synchronize the time to the client:

```
C:401:SET OPTIONS DateTime=583080894
```

Annotation:

Parameter	Description
DateTime	The value means the current time of the server and is in seconds converted by using the method in Appendix 5.

Some devices stop time synchronization after receiving this command, while some devices immediately trigger the following request after executing this command. Compatibility should be considered.

Request:

```
GET /iclock/rtdata?SN=3383154200002&type=time HTTP/1.1
Cookie: token=af65a75608cf5b80fbb3b48f0b4df95a
Host: 192.168.213.17:8088
…
```

Response:

```
HTTP/1.1 200 OK
Server: Apache-Coyote/1.1
Content-Type: text/plain;charset=UTF-8
Content-Length: 33
Date: Wed, 11 Jan 2017 01:30:03 GMT
DateTime=583050990,ServerTZ=+0800
```

Annotation:

Parameter	Description
DateTime	The value means a Greenwich mean time and is in seconds converted by using the method in Appendix 5.
ServerTZ	The time zone of the server

Modify the communication password:

```
C:402:SET OPTIONS ComPwd=1
```

Set access control parameters:

```
C:405:SET OPTIONS
Door1Drivertime=5,Door1KeepOpenTimeZone=0,Door1ValidTZ=1,Door1SupperPas
sWord=,Door1Intertime=0,Door1VerifyType=0,Door1SensorType=1,Door1Detect
ortime=15,Door1CloseAndLock=0,Door1ForcePassWord=,Reader1IOState=0,Wieg
andIDIn=1,WiegandID=1
```

Door1Drivertime: Lock driver time length of Door 1

Door1KeepOpenTimeZone: Normal Open time period of Door 1

Door1ValidTZ: Activation validity period of Door 1

Door1SupperPassWord: Supper password

Door1Intertime: Punch interval (not applicable to the new-architecture device)

Door1VerifyType: Verification mode of Door 1

Door1SensorType: Door sensor type of Door 1. 0: none. 1: Normal open. 2: Normal close.

Door1Detectortime: Door sensor delay time of Door 1

Door1CloseAndLock: Support lock at door closing or not

Door1ForcePassWord: Duress password

Reader1IOState: In/out status of Reader 1

WiegandIDIn:

WiegandID:

Access control new verification parameters:

If the device supports the new verification method, the verification methods of Door1VerifyType and

DoorVerifyType will not follow the rules described in Appendix 3, but the following new rules:

Supported verification methods, a total of 16 digits, currently only supports up to 7 digits, of which the 0th digit is the logical digit and is issued as a string.

```
typedef enum _VERIFY_TYPE_E
{
  VF_TYPE_LOGIC = 0, //0: logical digit, 0: or; 1: and
  VF_TYPE_FACE = 1, //1: face
  VF_TYPE_PALM_PRINT = 2, //2: palmprint
  VF_TYPE_PALM_VEIN = 3, //3: palm vein
  VF_TYPE_FP = 4, //4: fingerprint
  VF_TYPE_VEIN = 5, //5: finger vein
  VF_TYPE_VP = 6, //6: vocal print
  VF_TYPE_IRIS = 7, //7: iris
  VF_TYPE_RETINA = 8, //8: retina
  VF_TYPE_PW = 9, //9: password
  VF_TYPE_PIN = 10, //10: User ID
  VF_TYPE_RF = 11, //11: card
  VF_TYPE_HEALTH_QR_CODE = 12, //12: QR code/Health code
}VERIFY_TYPE_E;
```

Such as: card & password & face: 0000101000000011

Card | password | face: 0000101000000010

Card | password | face | QR code/Health code: 0001101000000010

QR code encryption parameters:

```
C:402:SET OPTIONS QRCodeDecryptType=XXX,QRCodeDecryptKey=XXX
```

QRCodeDecryptType: QR code decryption method, currently supports three methods, value 1, 2, 3.

1. Use scheme one, use today's date as the key and use AES256 algorithm to encrypt (the key is fixed)
2. Use scheme two, the system randomly generates a key and uses AES256 algorithm for encryption (the key is not fixed).
3. Use scheme three, the system randomly generates a key and uses RSA1024 algorithm for encryption

(public and private keys are not fixed).

QRCodeDecryptKey: QR code key

Elevator Control Parameters:

After the voice module function is turned on, the following parameters can be set:

VoiceModuleType: Voice module control mode, default: 0.

0: No verification required; 1: Verification required.

VoiceModuleCancelTime: The output time when the voice module cancels the button. Default: 2(s).

VoiceModuleCancelCount: The output times when the voice module cancels the button. Default: 1.

Video Intercom Contact List Parameters:

IsSupportSIPServer: Whether supports SIP server 0: Not supported; 1: Supported

ServerIPAddress: SIP server address

SIPUser: SIP user name

SIPAuthName: SIP authentication name

SIPKey: SIP password

SIPTransferProtocol: SIP transport protocol 0: UDP; 1: TCP; 2: TLS

SIPIsVerifyPeer: SIP verify TLS certificate 0: Do not verify; 1: Verify

IntercomCallMode: Call mode 1: Direct Dial mode; 2: Directory mode

IntercomContactID: Contact ID for intercom calls in direct calling mode

IntercomGroupSearchFunOn: Categorized search (enabled or not)

IntercomGroupDesc: Categorization label name

IntercomContactHintDesc: Description of contact list search prompt information

The client uploads the successful execution result:

```
ID=405&Return=0&CMD=SET OPTIONS
```

9.5.2 Get Options

Application scenario

It is used to obtain the device configurations.

Format

The server delivers the command:

```
C:$(CmdID):GET$(SP)OPTIONS$(SP)parameter name list
```

The client uploads the parameters to get:

```
POST
/iclock/querydata?SN=$(SerialNumber)&type=options&cmdid=$(CmdID)&tablenam
e=options&count=1&packcnt=1&packidx=1 HTTP/1.1

…

Parameter-value list
```

The client uploads the execution result:

```
ID=$(CmdID)&Return=0&CMD=GET OPTIONS
```

Annotation

Parameter	Description
/iclock/querydata	The URI of the data that is uploaded by the client and meets the condition.
type	"options" means the type of the data to be uploaded by the client is the parameter.
tablename	The table name. "options" means a parameter configuration table.
count	The number of data entries returned.
packcnt	The total number of packages. When there is a great amount of data, the data needs to be divided into different packages. This value indicates how many packages the data is divided into.
packidx	The package ID, which indicates which of all packages is currently being sent.
Note: Multiple parameters are separated by comma but not the last parameter.	

Example

```
C:408:GET OPTIONS
~SerialNumber,FirmVer,~DeviceName,LockCount,ReaderCount,AuxInCount,AuxOut
Count,MachineType,~IsOnlyRFMachine,~MaxUserCount,~MaxAttLogCount,~MaxFing
erCount,~MaxUserFingerCount,MThreshold,NetMask,GATEIPAddress,~ZKFPVersion
,SimpleEventType,VerifyStyles,EventTypes,ComPwd
```

Annotations :

Parameter	Description
~SerialNumber	The SN
FirmVer	The version of the firmware.
DeviceName	The name of the device.
LockCount	The number of doors.
ReaderCount	The number of readers.
AuxInCount	The number of auxiliary inputs.
AuxOutCount	The number of auxiliary outputs.
MachineType	The machine model. PULL SDK Device 101.
~IsOnlyRFMachine	Specify whether only the enable/disable parameter of the RF card is supported.
~MaxUserCount	The maximum number of users or cards.
~MaxAttLogCount	The maximum number of attendance records.
~MaxUserFingerCount	The maximum number of fingers of a single user.
MThreshold	The N match threshold.
IPAddress	The IP address of the wired network.
NetMask	The subnet mask of the wired network.
GATEIPAddress	The gateway address of the wired network.
~ZKFPVersion	The version of the finger algorithm.
SimpleEventType	Event merging.
VerifyStyles	The supported verification mode. It contains 32 bits in total. Each of the first 16 bits corresponds to a different combination of verification modes. For details, see Appendix 3

	Description of Verification Mode Code: 0 Verification mode is not supported. 1 Verification mode is supported. 200 An access control event.
EventTypes	The supported event type, which contains 32 bytes (0 to 31) in total. Each byte contains 8 bits, represented by 2 hexadecimal numbers, a total of 256 bites (0 to 255). For the function represented by each bit, see Appendix 2. Description of Real-time Events: 0 Event is not supported. 1 Event is supported. Take BF0FE03D300001000000000070000000000000000000000077002001000000 as an example. The 0th byte is BF, which is 11111101 in binary mode (the lower bit is placed first), the 6th bit is 0 and other bits are 1. It means that in the function controlled by the first eight bits, only the linkage event represented by the 6th bit is not supported, and other events are supported.
ComPwd	The communication password.

```
C:409:GET OPTIONS
IclockSvrFun,OverallAntiFunOn,~REXInputFunOn,~CardFormatFunOn,~SupAuthrizeFunOn,~ReaderCFGFunOn,~ReaderLinkageFunOn,~RelayStateFunOn,~Ext485ReaderFunOn,~TimeAPBFunOn,~CtlAllRelayFunOn,~LossCardFunOn,DisableUserFunOn,DeleteAndFunOn,LogIDFunOn,DateFmtFunOn,DelAllLossCardFunOn,DelayOpenDoorFunOn,UserOpenDoorDelayFunOn,MultiCardInterTimeFunOn,DSTFunOn,OutRelaySetFunOn,MachineTZFunOn,AutoServerFunOn,PC485AsInbio485,MasterInbio485,RS232BaudRate,AutoServerMode,IPCLinkFunOn,IPCLinkServerIP
```

Annotations:

Parameter	Description
IclockSvrFun	Whether to enable ADMS. 0 Disable. 1 Enable.
OverallAntiFunOn	Specify whether to enable the regional APB function. It is not

	used.
~REXInputFunOn	Specify whether to enable the exit door locking function.
~CardFormatFunOn	Specify whether the Wiegand reader supports the user-defined card format.
~SupAuthrizeFunOn	Specify whether to enable the super user function.
~ReaderCFGFunOn	Specify whether to enable the reader configuration function. It is not used.
~ReaderLinkageFunOn	Specify whether to enable the reader linkage function.
~RelayStateFunOn	Specify whether to support the relay status function.
~Ext485ReaderFunOn	Specify whether to enable the external reader 485 function.
~TimeAPBFunOn	Specify whether to enable the APB time function.
~ CtlAllRelayFunOn	Specify whether to enable the All Relay function.
~ LossCardFunOn	Specify whether to enable the function of deleting all blacklists.
DisableUserFunOn	Specify whether to enable the blacklist function.
DeleteAndFunOn	Specify whether to enable the AND deletion function.
LogIDFunOn	Specify whether to enable the ID logging function.
DateFmtFunOn	Specify whether to enable the date format function.
DelAllLossCardFunOn	Specify whether to enable the function of deleting all blacklists.
DelayOpenDoorFunOn	Specify whether to enable the delay open door function. This function is not supported.
UserOpenDoorDelayFunOn	Specify whether to enable the function of delaying the door open time for a user. This function is not supported.
MultiCardInterTimeFunOn	Specify whether to enable the function of setting the interval for multiple cards. This function is not supported.
DSTFunOn	Specify whether to enable the DLST function. This function is not supported.
OutRelaySetFunOn	Specify whether to enable the configuration output function. This function is not supported.
MachineTZFunOn	Specify whether to enable the function of setting a time zone

	for the machine. This function is not supported.
AutoServerFunOn	Specify whether to enable the remote identification function.
PC485AsInbio485	485 reader/communication switchover.
MasterInbio485	485 reader/communication switchover.
RS232BaudRate	RS232/485 baud rate.
AutoServerMode	Remote identification mode.
IPCLinkFunOn	Specify whether to enable the soft linkage with IPC. This function is not supported.
IPCLinkServerIP	The IPC address. This function is not supported.

```
C:410:GET OPTIONS FingerFunOn,FvFunOn,FaceFunOn,~MaxFace7Count,~MaxFvCount
```

Annotations :

Parameter	Description
FingerFunOn	Specify whether to enable the finger function.
FvFunOn	Specify whether to enable the finger vein function.
FaceFunOn	Specify whether to enable the face function.
MaxFace7Count	The maximum number of face templates.
MaxFvCount	The IPC address. This function is not supported.

9.5.3 Set Adscreen_Options (only supported by the channel controller)

Application scenario

It is used to set the parameters required for the AD screen device to display AD pictures and videos for the screen device to download pictures and videos for use.

Format

The server delivers the command:

```
C:$(CmdID):SET$(SP)ADSCREEN_OPTIONS$(SP)parameter-value list
```

The client first uploads the execution result -5000, and prompts the software to wait for the screen device to return the download result:

```
ID=$(CmdID)&Return=5000&CMD=SET ADSCREEN_OPTIONS
```

After the screen device returns the download result, the client uploads a command response again, the CmdID remains unchanged, and the update software command return value:

```
ID=$(CmdID)&Return=&{XXX}&CMD=SET ADSCREEN_OPTIONS
```

Annotation

Parameter list format: InOut=x,Area=x,Type=x,Range=x,URL=xxx,InPlayInterval/OutPlayInterval=x

Parameter	Description
InPlayInterval	The switching interval of In
OutPlayInterval	The switching interval of Out
Range	The size of the display area occupying the screen
InOut	0-In, 1-Out
Area	Area number, 1\2\3…
Type	0-Video, 1-Picture/icon, 2-Text, 3-Firmware upgrade
URL	The url address of the file to be pulled

Description of the return value of the screen pull result:

-1	Failed to send command
≥0	Success
-4	Failed to unzip
-18	The device has no space
-618	URL format error

9.6 Remote Registration

9.6.1 ENROLL_INFO

Application scenario

Initiated by the server, the general data is registered on the client.

Command Format

The server delivers the command:

```
C:$(CmdID):ENROLL_INFO TYPE=%?\tPIN=%?\tRETRY=%?\tMODE=%?
```

The client uploads the registration data:

```
POST /iclock/cdata?SN=${SerialNumber}&type=EnrollInfoData HTTP/1.1
Host: ${ServerIP}:${ServerPort}
Content-Length: ${XXX}
……
TYPE=${XXX}${HT}PIN=${XXX}${HT}TMP=${XXX}
```

The client uploads the execution result:

```
ID=${XXX}&Return=${XXX}&CMD=ENROLL_INFO
```

Annotation

Parameter	Description
TYPE	Registration type: 0: MF card number **Note:** If the TYPE field is not delivered, the device defaults to processing with TYPE=0.
PIN	Registered User ID.
RETRY	If the registration fails, the number of times to retry.
MODE	Saving and uploading methods: 0 : The device does not save registration data but requires uploading of registration data. 1 : The device saves registration data and also uploads registration data. **Note:** If the MODE field is not delivered, the device defaults to processing with MODE=0.
TMP	Registered data.

9.6.2 ENROLL_BIO

Application scenario

Initiated by the server, the bio template is registered on the client.

Command Format

The server delivers the command:

```
C:$(CmdID):ENROLL_BIO
TYPE=%?\tNO=%?\tPIN=%?\tRETRY=%?\tOVERWRITE=%?\tMODE=%?
```

The client uploads the registration data:

```
POST /iclock/cdata?SN=${SerialNumber}&type=BioData HTTP/1.1
Host: ${ServerIP}:${ServerPort}
Content-Length: ${XXX}
.....
TYPE=${XXX}${HT}PIN=${XXX}${HT}NO=${XXX}${HT}INDEX=${XXX}${HT}MAJOR_VER=${XXX}${HT}MINOR_VER=${XXX}${HT}TMP=${XXX}${HT}BIOPHOTO=${XXX}
.....
TYPE=${XXX}${HT}PIN=${XXX}${HT}NO=${XXX}${HT}INDEX=${XXX}${HT}MAJOR_VER=${XXX}${HT}MINOR_VER=${XXX}${HT}TMP=${XXX}${HT}BIOPHOTO=${XXX}
```

The client uploads the execution result:

```
ID=${XXX}&Return=${XXX}&CMD=ENROLL_BIO
```

Annotation

<table>
<tr><th>Parameter</th><th>Description</th></tr>
<tr><td>TYPE</td><td>Registration biometric type:
<table>
<tr><th>Value</th><th>Meaning</th></tr>
<tr><td>0</td><td>General</td></tr>
<tr><td>1</td><td>Fingerprint</td></tr>
<tr><td>2</td><td>Face (Near Infrared)</td></tr>
</table>
</td></tr>
</table>

	3 Voice Print 4 Iris 5 Retina 6 Palm Print 7 Finger Vein 8 Palm 9 Visible Light Face
NO	Biological template specific individual number: **Biological Template** / **Description** Fingerprint: The number is 0 to 9. The corresponding fingers are: Left hand : little finger/ ring finger/ middle finger/ index finger/ thumb; Right hand : thumb/ index finger/ middle finger/ ring finger/ little thumb Finger vein: Same as fingerprint. Face & Visible Light Face: The number is 0. Iris: 0 : Left Eye; 1 : Right Eye Palm vein: 0 : Left Hand; 1 : Right Hand; If not clearly defined, the default value is 0.
PIN	Registered User ID.
RETRY	If the registration fails, the number of times to retry.
OVERWRITE	Whether to overwrite: 0 : If the corresponding user already exists, it will not be overwritten and an error will be returned. 1 : The corresponding user exists and covers.

INDEX	Registration bio template index (if there is only one template, this field can be left blank).
MAJOR_VER	Registration bio template major version number.
MINOR_VER	Registration bio template minor version number.
TMP	Registered template data (based on BASE64 encoding format).
MODE	Saving and uploading methods: 0 : The device does not save registration data but requires uploading of registration data. 1 : The device saves registration data and also uploads registration data. **Note:** If the MODE field is not delivered, the device defaults to processing with MODE=0.
BIOPHOTO (Optional)	Original JPEG format photo of registration bio template (BASE64 encoding)
Note: In case of multiple bio templates, each template data is separated by ' r n'.	

Example

The software issues a visible light face with a registered User ID of 1, and retry at most 3 times during registration. If the user has a face, it will cover:

```
C:395:ENROLL_BIO TYPE=9  NO=0  PIN=1 RETRY=3  OVERWRITE=1 MODE=1
```

The client uploads the registered visible light facial data:

```
POST /iclock/cdata?SN=${SerialNumber}&type=BioData HTTP/1.1
Host: ${ServerIP}:${ServerPort}
Content-Length: ${XXX}
TYPE=9 PIN=1 NO=0 INDEX=0 MAJOR_VER=58 MINOR_VER=1 TMP=XXXXXXXXXXXXXX
BIOPHOTO=XXXXXXXXXXXXXX...
```

The client uploads the successful execution result:

```
ID=395&return=0&CMD=ENROLL_BIO
```

10 Remote Identification

The device sends the data that passes identification to the server. After identification, the server returns the result, the data that passes identification, and the control command, as shown below:

Client Request

```
POST /iclock/cdata?SN=${SerialNumber}&AuthType=device HTTP/1.1
Host: ${ServerIP}:${ServerPort}
Content-Length: ${XXX}
…
time=${XXX}${HT}pin=${XXX}${HT}cardno=${XXX}${HT}addrtype=${XXX}${HT}eventaddr=${XXX}${HT}event=${XXX}${HT}inoutstatus=${XXX}${HT}verifytype=${XXX}
```

Annotation

```
HTTP request method: POST
URI: /iclock/cdata
HTTP version: 1.1
```

Client configurations:

Annotation	Required/Optional	Description
SN	${Required}	AuthType=device means globally remote APB.
Host header	${Required}	
Content-Length header	${Required}	
Other headers	${Optional}	
Request entity	Event record	
time	Time, in the format of YYYY-MM-DD HH:MM:SS	
pin	User ID (it is 0 is used when it is left blank).	
cardno	Card number	

<table>
<tr><td>addrtype</td><td>Event point type, that is, by whom the event is generated.<table><tr><th>Value</th><th>Event Generator</th></tr><tr><td>1</td><td>Device</td></tr><tr><td>2</td><td>Door</td></tr><tr><td>3</td><td>Reader</td></tr><tr><td>4</td><td>Auxiliary input</td></tr><tr><td>5</td><td>Auxiliary output</td></tr></table></td></tr>
<tr><td>eventaddr</td><td>The event address. If the event is generated by the device, this value can be the device address. If the event is generated by the reader, this value can be the reader address. Currently, the door is the only option.</td></tr>
<tr><td>event</td><td>Event type</td></tr>
<tr><td>inoutstatus</td><td>IN or OUT status, that is, entering or exiting the door.</td></tr>
<tr><td>verifytype</td><td>Verification type</td></tr>
</table>

Server Response

When the user information is available, the response is as follows:

```
HTTP/1.1 200 OK
Date: ${XXX}
Content-Length: ${XXX}
…
AUTH=${XXX}${CR}${LF}time=${XXX}${HT}pin=${XXX}${HT}cardno=${XXX}${HT}addrtype=${XXX}${HT}eventaddr=${XXX}${HT}event=${XXX}${HT}inoutstatus=${XXX}${HT}verifytype=${XXX}${CR}${LF}CONTROL$(SP)DEVICE$(SP)AABBCCDDEE${CR}${LF}TIPS=${XXX}
```

Annotation

<table>
<tr><th>Parameters</th><th>Description</th></tr>
<tr><td>AUTH</td><td>Whether the verification is successful or not.<table><tr><th>Value</th><th>Description</th></tr><tr><td>SUCCESS</td><td>The verification is successful</td></tr></table></td></tr>
</table>

	FAILED: The verification failed TIMEOUT: The verification times out
EVENT	The requested original event record
CONTROL DEVICE	The controller command after successful verification. For more details, check Control Device.
TIPS	Verification prompt information (UTF-8 encoding). This field is optional.

11 Appendixes

11.1 Appendix 1 - Description of Return Values of Commands

Return value	Description
>=0	Successful
-1	Failed to send the command
-2	No response to the command
-3	Insufficient cache
-4	Failed to decompress
-5	Incorrect length of read data
-6	The length after decompression does not meet the required one
-7	Repeated command
-8	Unauthorized connection
-9	Incorrect data. CRC failed
-10	Incorrect data. Failed to resolve data API
-11	Incorrect data parameter
-12	Command execution error
-13	Incorrect command. The command does not exit
-14	Incorrect communication password
-15	Failed to write the file
-16	Failed to read the file
-17	The file does not exist
-18	The device has insufficient space
-19	Checksum error
-20	The length of the received data is inconsistent with the specified data length
-21	No platform parameter is set on the device
-22	The received firmware platform for upgrade is not same to the local platform
-23	The firmware version for upgrade is earlier than that in the device

-24	Incorrect flag of the upgrade file
-25	The name of the received file for firmware upgrade is not emfw.cfg
-26	The length of the received fingerprint template is 0
-27	The PIN of the received fingerprint template is incorrect. Failed to find the user.
-28	Door opening command is executed during the open time period.
-30	The integrated template algorithm's version is inconsistent
[-400,-100] is the PULL SDK internal return value	
-100	The table structure does not exist
-101	The conditional field in the table structure does not exist
-102	The total field count is inconsistent
-103	The field sequence is inconsistent
-104	Incorrect real-time event data
-105	Incorrect data for resolution
-106	The size of the delivered data exceeds 4 MB
-107	Failed to get the table structure
-108	Invalid OPTIONS
-201	LoadLibrary failed
-202	Failed to call the interface
-203	Communication initialization failed
-206	Failed to start the serial port agent program. A general cause is that the serial port does not exist or has been occupied
-301	Failed to get the TCP/IP version
[*,-600] is a returned value exclusive to push communication (Return value already mentioned above cannot be defined repeatedly)	
-601	Reserved. For firmware internal use only
-603	Internal error: invalid handle
-604	Insufficient memory
-609	Internal error: invalid handle
-612	The device failed to save the parameter

-614	Failed to remotely cancel the alarm
-615	Remote restart failed
-616	Remotely enabling or canceling normal open failed
-618	Incorrect URL format
-619	Firmware internal error: blank input parameter
-620	The format of the data delivered by the server is incorrect
-621	The table structure does not support this field
-628	Main failed to load the fingerprint to the memory
-629	Incorrect table name
-630	Failed to execute the shell command
-631	Incorrect base64 length
-632	Incorrect file name
-701	Reserved. For firmware internal use only
-702	Reserved. For firmware internal use only
-703	Reserved. For firmware internal use only
-704	Reserved. For firmware internal use only
-705	Reserved. For firmware internal use only
-706	Reserved. For firmware internal use only
-707	Reserved. For firmware internal use only
-708	The firmware does not support this command
-709	Failed to upload options
-720	Failed to upgrade the firmware
-721	The device downloads the file
-722	Failed to upload the data to the software
-723	Reserved. For firmware internal use only
-724	Reserved. For firmware internal use only
-725	It times out when the device waits for the result
-726	The server returns an HTTP error
-727	Reserved. For firmware internal use only

-728	Failed to upload the number of table data entries
-729	Reserved. For firmware internal use only
-730	No response from the server
-731	An error occurs when the server receives the data
-732	Reserved. For firmware internal use only
-801	Incorrect command format. The correct format is cmd:SN:command

11.2 Appendix 2 - Description of Real-time Events

Event Code	Event Type
0~19 & 200~253	Normal Events
20~99	Error Events
100~199	Alarm Events

NOTE: The red font is the event code transplanted from the BEST protocol, which is currently only used on controller devices that support the BEST protocol.

The blue font is the special event code for elevator control device.

Event code	Description		Remarks
0	The door opens normally after punch	The door opens normally after verification	The event codes 0, 14, and 17 are merged and have the same meaning.
1	Punch in the normal open time period	Verify in the normal open time period	The event codes 1 and 16 are merged and have the same meaning.
2	The first user opens the door by punch	The first user opens the door	The event codes 2, 18, and 19 are

			merged and have the same meaning.
3	Multiple users open the door by punch	Multiple users open the door	The event codes 3, 15, and 203 are merged and have the same meaning.
4	Open the door by using the emergency password	Open the door by using the emergency password	
5	Open the door in the normal open time period	Open the door in the normal open time period	
6	Trigger linkage	Trigger linkage	The value of the verification mode in the linkage event record is the specific event type
7	Cancel the alarm	Cancel the alarm	
8	Open the door remotely	Open the door remotely	
9	Close the door remotely	Close the door remotely	
10	Disable the normal open time period of the day	Disable the normal open time period of the day	
11	Enable the normal open time period of the day	Enable the normal open time period of the day	
12	Enable auxiliary output remotely	Enable auxiliary output remotely	
13	Disable auxiliary output remotely	Disable auxiliary output remotely	
14	The door is normally opened by fingerprint	The door is normally opened after verification	The event codes 0, 14, and 17 are merged and have the same

			meaning.
15	Multiple users open the door by fingerprint	Multiple users open the door	The event codes 3, 15, and 203 are merged and have the same meaning.
16	Check the fingerprint in the normal open time period	Perform verification in the normal open time period	The event codes 1 and 16 are merged and have the same meaning.
17	Open the door by card and fingerprint	Open the door by verification	The event codes 0, 14, and 17 are merged and have the same meaning.
18	The first user opens the door by fingerprint	The first user opens the door	The event codes 2, 18, and 19 are merged and have the same meaning.
19	The first user opens the door by card and fingerprint	The first user opens the door	The event codes 2, 18, and 19 are merged and have the same meaning.
20	The punch interval is too short	The operation interval is too short	The event codes 20, 31, and 50 are merged and have the same

			meaning.
21	Punch during the non-valid time period	Open the door by verification during the non-valid time period	The event codes 21, 35, and 49 are merged and have the same meaning.
22	Invalid time period	Invalid time period	
23	Unauthorized access	Unauthorized access	
24	APB	APB	
25	Interlock	Interlock	
26	Multiple users perform verification by punch	Multiple users wait for verification	The event codes 26, 32, and 51 are merged and have the same meaning
27	The card is not registered	The user is not registered	The event codes 27, 30, and 34 are merged and have the same meaning
28	Door opening times out	Door opening times out	
29	The card privilege expires	The user privilege expires	The event codes 29, 33, and 53 are merged and have the same meaning
30	The password is incorrect	The user is not registered	The event codes 27, 30, and 34 are merged and have the same

			meaning
31	The fingerprint check interval is too short	The operation interval is too short	The event codes 20, 31, and 50 are merged and have the same meaning
32	Multiple users perform verification by fingerprint	Multiple users wait for verification	The event codes 26, 32, and 51 are merged and have the same meaning
33	The fingerprint expires	The user privilege expires	The event codes 29, 33, and 53 are merged and have the same meaning
34	The fingerprint is not registered	The user is not registered	The event codes 27, 30, and 34 are merged and have the same meaning
35	Check the fingerprint during the non-valid time period	Open the door by verification during the non-valid time period	The event codes 21, 35, and 49 are merged and have the same meaning
36	Press the exit button during the non-valid time period	Press the exit button during the non-valid time period	
37	The door cannot be closed during the normal open time period	The door cannot be closed during the normal open time period	
38	The card has been reported as lost	The card has been reported as lost	

39	Blacklist	Blacklist	
40	Multi-user verification by fingerprint failed	Multi-user verification failed	The event codes 40, 48, and 52 are merged and have the same meaning
41	Incorrect verification mode	Incorrect verification mode	
42	Incorrect Wiegand format	Incorrect Wiegand format	
43	The user is focused	The user is focused	
44	Remote identification failed	Remote identification failed	
45	Remote identification times out	Remote identification times out	
47	Failed to send the command	Failed to send the command	
48	Multi-user verification by punch failed	Multi-user verification failed	The event codes 40, 48, and 52 are merged and have the same meaning
49	Open the door by password during the non-valid time period	Open the door by verification during the non-valid time period	The event codes 21, 35, and 49 are merged and have the same meaning
50	The password verification interval is too short	The operation interval is too short	The event codes 20, 31, and 50 are merged and have the same meaning
51	Multiple users perform verification by password	Multiple users wait for verification	The event codes 26, 32, and 51 are merged

			and have the same meaning
52	Multi-user verification by password failed	Multi-user verification failed	The event codes 40, 48, and 52 are merged and have the same meaning
53	The password expires	The user privilege expires	The event codes 29, 33, and 53 are merged and have the same meaning
54	Battery voltage is too low	Battery voltage is too low	
55	Replace the battery immediately	Replace the battery immediately	
56	Unauthorized operation	Unauthorized operation	
57	Backup power supply	Backup power supply	
58	Normal open alarm	Normal open alarm	
59	Unauthorized admin	Unauthorized admin	
60	The door is locked inside	The door is locked inside	
61	Repeat verification	Repeat verification	
62	Prohibited user	Prohibited user	
63	Door locked	Door locked	
64	The exit button is not operated within the time period	The exit button is not operated within the time period	
65	Auxiliary input is not operated within the time period	Auxiliary input is not operated within the time period	
66	Reader upgrade failed	Reader upgrade failed	
67	Remote comparison is successful (device is not authorized)	Remote comparison is successful (device is not authorized)	
68	High bogy temperature -Access Denied	High bogy temperature -Access Denied	
69	Without mask -Access Denied	Without mask -Access Denied	

70	The face comparison server communication is abnormal	The face comparison server communication is abnormal	
71	Face server responds abnormally	Face server responds abnormally	
72	Call unanswered	Call unanswered	
73	Invalid QR code	Invalid QR code	
74	QR code has expired	QR code has expired	
75	Combined verification timeout	Combined verification timeout	
76	Cannot get fingerprints in TITAN mode	Cannot get fingerprints in TITAN mode	
77	Internet cable not plugged in	Internet cable not plugged in	
78	Device hotspot disconnected abnormally	Device hotspot disconnected abnormally	
79	Mobile network disconnected abnormally	Mobile network disconnected abnormally	
80	The elevator control floor selection button is triggered without any permission	The elevator control floor selection button is triggered without any permission	
100	Tamper alarm	Tamper alarm	
101	Open the door with the duress password	Alarm for opening the door with the duress password	The event codes 101 and 103 are merged and have the same meaning
102	The door is opened accidentally	The door is opened accidentally	
103	Open the door with the duress fingerprint	Alarm for opening the door with the duress fingerprint	The event codes 101 and 103 are merged and have the same meaning
104	Alarm for punch with an invalid card	Alarm for punch with an invalid card	
105	Network disconnected	Failed to connect to the server	
106	Battery power failure	Battery power failure	
107	Mains power failure	Mains power failure	

108	Failed to connect to the main controller	Failed to connect to the main controller	
109	Reader disassembly alarm	Reader disassembly alarm	
110	Reader offline alarm	Read head offline alarm	
111	POE power down	POE power down	
112	Extension board offline alarm	Extension board offline alarm	
113	Illegal security level	Illegal security level	
114	Line detection, fire alarm input disconnected	Line detection, fire alarm input disconnected	
115	Line detection, fire alarm input short circuit	Line detection, fire alarm input short circuit	
116	Line detection, auxiliary input disconnect	Line detection, auxiliary input disconnect	
117	Line detection, auxiliary input short circuit	Line detection, auxiliary input short circuit	
118	Line detection, the exit button is off	Line detection, the exit button is off	
119	Line detection, short circuit of exit button	Line detection, short circuit of exit button	
120	Line detection, door sensor disconnected	Line detection, door sensor disconnected	
121	Line detection, door sensor short circuit	Line detection, door sensor short circuit	
132	Cancel the emergency state of the elevator control	Cancel the emergency state of the elevator control	
133	It is not possible to cancel the elevator control emergency state.	It is not possible to cancel the elevator control emergency state.	
134	The elevator control is in a restricted state	The elevator control is in a restricted state	
150	The elevator control manual button is short-circuited	The elevator control manual button is short-circuited	
151	The elevator control manual button is disconnected	The elevator control manual button is disconnected	
152	The elevator control emergency button is short-circuited	The elevator control emergency button is short-circuited	
153	The elevator control emergency	The elevator control emergency	

	button is disconnected	button is disconnected	
154	The elevator control fire button is short-circuited	The elevator control fire button is short-circuited	
155	The elevator control fire button is disconnected	The elevator control fire button is disconnected	
200	The door is open	The door is open	
201	The door is closed	The door is closed	
202	Open the door by pressing the exit button	Open the door by pressing the exit button	
203	Multiple users open the door by card and fingerprint	Multiple users open the door	The event codes 3, 15, and 203 are merged and have the same meaning
204	The door normal open time period ends	The door normal open time period ends	
205	Remotely enable normal open for the door	Remotely enable normal open for the door	
206	Start the device	Start the device	
207	Open the door by password	Open the door by password	
208	The super user opens the door	The super user opens the door	
209	Press the exit button (locked)	Press the exit button (locked)	
210	Start firefighting door opening	Start firefighting door opening	
211	The super user closes the door	The super user closes the door	
212	Enable the elevator control function	Enable the elevator control function	
213	Disable the elevator control function	Disable the elevator control function	
214	Successfully connected to the server	Successfully connected to the server	
215	The first card opens the door by password	The first card opens the door by password	
216	Open the door by password during the normal open time period	Open the door by password during the normal open time period	
217	Successfully connected to the main controller	Successfully connected to the main controller	

218	Pass by identity card	Pass by identity card	
219	Verification within the time period when the exit button is normally open	Verification within the time period when the exit button is normally open	
220	Disconnected from the auxiliary input point	Disconnected from the auxiliary input point	
221	The auxiliary input point is short-circuited	The auxiliary input point is short-circuited	
222	Remote identification is successful	Remote identification is successful	
223	Remote identification	Remote identification	
224	Ring the doorbell	Ring the doorbell	
225	The auxiliary input point is normal	The auxiliary input point is normal	
226	Trigger the auxiliary input point	Trigger the auxiliary input point	
227	The door is in dual-on mode	The door is in dual-on mode	
228	The door is in dual-off mode	The door is in dual-off mode	
229	Regularly enable normal open for auxiliary output	Regularly enable normal open for auxiliary output	
230	Regularly disable normal open for auxiliary output	Regularly disable normal open for auxiliary output	
231	IPC firmware linkage event	IPC firmware linkage event	The value of the verification mode in the linkage event record is the specific event type
232	Passed the verification	Passed the verification	
233	Remote lock	Remote lock	
234	Remote unlock	Remote unlock	
235	Reader upgrade successfully	Reader upgrade successfully	
236	Reader disassembly alarm release	Reader disassembly alarm release	
237	Reader online	Reader online	
239	Device call	Device call	
240	Call ended	Call ended	

241	Linked capture event	Linked capture event	
242	Linked recording event	Linked recording event	
243	Fire alarm input disconnected	Fire alarm input disconnected	
244	Fire alarm input short circuit	Fire alarm input short circuit	
245	Connect to hotspot	Connect to hotspot	
246	Mobile network connection is successful	Mobile network connection is successful	
247	Expansion board online	Expansion board online	
248	Plug in the internet cable	Plug in the internet cable	
249	The elevator control direct button triggers	The elevator control direct button triggers	
250	The elevator control floor selection button is triggered with the permissions	The elevator control floor selection button is triggered with the permissions	
251	The elevator control cancels the button via voice	The elevator control cancels the button via voice	
252	The elevator control triggers the button via voice	The elevator control triggers the button via voice	
254	Definition of the extended event number	Definition of the extended event number	0-255 are insufficient and therefore 254 is used to define the number bit of the extended event. This function is newly supported by the firmware
255	Door status	Door status	

11.3 Appendix 3 - Description of Verification Mode Code

Let, the value of VerifyStyles be FB1E1F00 (11111011 00011110 00011111 00000000):

Then,

Binary Value	1	1	1	1	1	0	1	1	0	0	0	1	1	1	1	0	…
Position	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	…

Here, the position 15 corresponds to the binary value 0, which indicates that the face verification mode is not supported (the value of the face verification mode is 15).

Verification mode	Description
0	Vein, face, fingerprint, card, or password (automatically identified)
1	Fingerprint only
2	User ID
3	Password only
4	Card only
5	Fingerprint or password
6	Fingerprint or card
7	Card or password
8	User ID and fingerprint
9	Fingerprint and password
10	Card and fingerprint
11	Card and password
12	Fingerprint, password, and card
13	User ID, fingerprint, and password
14	User ID and fingerprint, or card and fingerprint
15	Face
16	Face and fingerprint
17	Face and password
18	Face and card
19	Face, fingerprint, and card

20	Face, fingerprint, and password
21	Finger vein
22	Finger vein and password
23	Finger vein and card
24	Finger vein, password, and card
25	Palm
26	Palm and card
27	Palm and face
28	Palm and fingerprint
29	Palm, fingerprint and face
200	Other

11.4 Appendix 4 - Language Number

Language Number	Meaning
83	Simplified Chinese
69	English
97	Spanish
70	French
66	Arabic
80	Portuguese
82	Russian
71	German
65	Farsi
76	Thai
73	Indonesian
74	Japanese
75	Korean
86	Vietnamese
116	Turkish

72	Hebrew
90	Czech
68	Dutch
105	Italian
89	Slovak
103	Greek
112	Polish
84	Traditional Chinese

11.5 Appendix 5 - Algorithm to Convert Date to Seconds

ZKTeco's algorithm to convert year, month, day, hour, minute, and second into values in seconds:

```
unsigned int OldEncodeTime(int year, int mon, int day, int hour, int min,
int sec)
{
    unsigned int tt;
    tt = ((year - 2000) * 12 * 31 + ((mon - 1) * 31) + day - 1) * (24 * 60
    * 60) + (hour * 60 + min) * 60 + sec;
    return tt;
}
```

11.6 Appendix 6 - Algorithm to Convert Seconds to Date

ZKTeco's algorithm to convert seconds to year, month, day, hour, minute, and second:

```
void OldDecodeTime(unsigned int tt, int *year, int *mon, int *day, int *hour,
int *min, int *sec)
{
    *sec = tt % 60;
    tt /= 60;
    *min = tt % 60;
    tt /= 60;
    *hour = tt % 24;
    tt /= 24;
    *day = tt % 31 + 1;
    tt /= 31;
```

```
    *mon = tt % 12 + 1;
    tt /= 12;
    *year = tt + 2000;
}
```

11.7 Appendix 7 - Device Types and Corresponding Models

Device type value	Model
1	C3-200
2	C3-400
3	C4-400
4	C3-100
5	C4-200 and INBIO280
6	C4 (four-door to two-door)
7	C3 (four-door to two-door)
8	C3-160 and INBIO160
9	C3-260 and INBIO260
10	C3-460 and INBIO460
23	C5-100
24	C5-200
25	C5-400
26	INBIO5-100
27	INBIO5-200
28	INBIO5-400
40	BIOIR9000
101	PULL SDK Device
102	Finger vein
103	iFace7
301	Wireless lock

11.8 Appendix 8 - APB Values

APB value	Description
0	No APB
1	APB between Door 1 and Door 2
2	APB between Door 3 and Door 4
3	APB between Door 1 and Door 2 and APB between Door 3 and Door 4
4	APB between Door 1 or 2 and Door 3 or 4
5	APB between Door 1 and Door 2 or 3
6	APB between Door 1 and Door 2 or 3 or 4
16	APB between readers 1
32	APB between readers 2
48	APB between each two of readers 1 and 2 respectively
64	APB between readers 3
80	APB between each two of readers 1 and 3 respectively
96	APB between each two of readers 2 and 3 respectively
112	APB between each two of readers 1, 2, and 3 respectively
128	APB between readers 4
144	APB between each two of readers 1 and 4 respectively
160	APB between each two of readers 2 and 4 respectively
176	APB between each two of readers 1, 2, and 4 respectively
196	APB between each two of readers 3 and 4 respectively
208	APB between each two of readers 1, 3, and 4 respectively
224	APB between each two of readers 2, 3, and 4 respectively
240	APB between each two of readers 1, 2, 3 and 4 respectively

11.9 Appendix 9 - Interlock Value

Interlock value	Description
0	None
1	Interlock between Door 1 and Door 2
2	Interlock between Door 3 and Door 4
3	Interlock among Door 1, Door 2, and Door 3
4	Interlock between Door 1 and Door 2, or between Door 3 and Door 4
5	Interlock among Door 1, Door 2, Door 3, and Door 4

11.10 Appendix 10 - Table of Access Control Parameters

Parameter name	Description
DoorAutoMatch	Automatic match of Wiegand format. 0: User-defined. 1: Auto
DoorDrivertime	Lock driver time length
DoorKeepOpenTimeZone	Normal open time period of the door. 0: Disable normal open
DoorCancelKeepOpenDelay	Cancel the normal open time (Linux year) * 10000 + (Linux month) * 100 + day
DoorValidTZ	Valid time period of the door
DoorSupperPassWord	Emergency password, up to eight bits
DoorIntertime	Punch interval in seconds. Default: 2s
DoorVerifyType	Door opening mode. See Appendix 3 Verification Mode Code
DoorSensorType	Door sensor type. 0: None. 1: Normal open. 2: Normal close
DoorDetectortime	Door sensor delay in seconds. 0: No alarm for the delay. >0: Alarm for delay
DoorCloseAndLock	Lock at door closing. 1: Enable. 0: Disable. Default: 1
DoorReaderType	Reader type
DoorForcePassWord	Duress password, up to 6 bits
DoorInTimeAPB	APB duration in seconds
DoorREXInput	Exit button status. 1: Do not lock (default). 0: Lock
DoorREXTimeOut	Exit button delay in seconds. The delay after the button switch event occurs when the door is locked
WiegandIDIn	Wiegand input type

WiegandID	Wiegand output type
DoorDelayOpenTime	Door open delay. The delay time after verification
ExtDoorDelayDrivertime	Extended pass time in seconds for the disabled
DoorMultiCardInterTime	Multi-user door opening interval in seconds
DoorFirstCardOpenDoor	First-card normal open. 0: Disable. 1: First-card normal open. 2: First-card activation
DoorMultiCardOpenDoor	Multi-card door opening. 0: Disable. 1: Enable
DoorDisable	Enable or disable the door. 1: Enable. 0: Disable.
DoorInOutAntiPassback	In/out APB for a single door. 0: No APB. 1: In APB. 2: Out APB. 3: In and out APB
DoorMaskFlag	Lock the door. 1: Lock

11.11 Appendix 11 - Table of Reader Parameters

Parameter name	Description
IdentityCardVerifyMode	0: Normal 1: Identity card

11.12 Appendix 12 - Table of Device Parameters

Parameter name	Description
DSTOn	DLST function
CurTimeMode	Current mode - 1: DLST. 2: Not DLST
DLSTMode	DLST mode
IPAddress	IP address
GATEIPAddress	Gateway
NetMask	Subnet mask
BackupTime	Regularly back up event records. Accuracy: hour
DelRecord	The number of records to delete after the number of event records exceeds the allowed maximum number.
MachineTZ	Time zone

AntiPassback	APB (see Appendix 8)
InterLock	Interlock (see Appendix 9)
NtpFunOn	NTP function on and off (0-off, 1-on)

11.13 Appendix 13 - Protocol Version Rule

- **Released version of the protocol:**

3.0.1

3.1.1

3.1.2

Encryption protocol version: 3.1.1 and above

- **Device end:**

The device pushes the current PUSH version to the server according to the following protocol:

```
GET /iclock/cdata?SN=${SerialNumber}&options=all&pushver=${XXX}
```

Based on the request, the server returns the version of the published protocol that is used by the server for development to the device.

```
PushProtVer=xxx
```

If this parameter is not returned, the default device protocol version of the server is 3.0.1.

The device compares its current PUSH version with the protocol version returned by the server and uses the lower version for interaction.

- **Server-side:**

The server sends the following request to get the version of PUSH used by the client. If no pushver field is returned, then the server uses PUSH version 3.0.1 by default.

```
GET /iclock/cdata?SN=${SerialNumber}&options=all&pushver=${XXX}
```

The server needs to return the version of the publish protocol used by the software:

```
PushProtVer=xxx
```

The server compares the software's protocol version with the protocol version uploaded by the device and uses the lower version for interaction.

11.14 Appendix 14 - Parameter CmdFormat

The following describes the meaning of the values of the parameter CmdFormat (its default value is not 0).

When the value is 0, the format is:

```
C:XXX:COMMAND……
```

When the value is 1, the format is:

```
DataType=1,SN=%s, Priority=%d, CmdID=%s,CmdDesc=%s
```

Note

DataType: The data format type, which is 1 by default

SN: The command is distributed to the specified SN.

Priority: The command priority, which is 0 by default

CmdID: The command ID

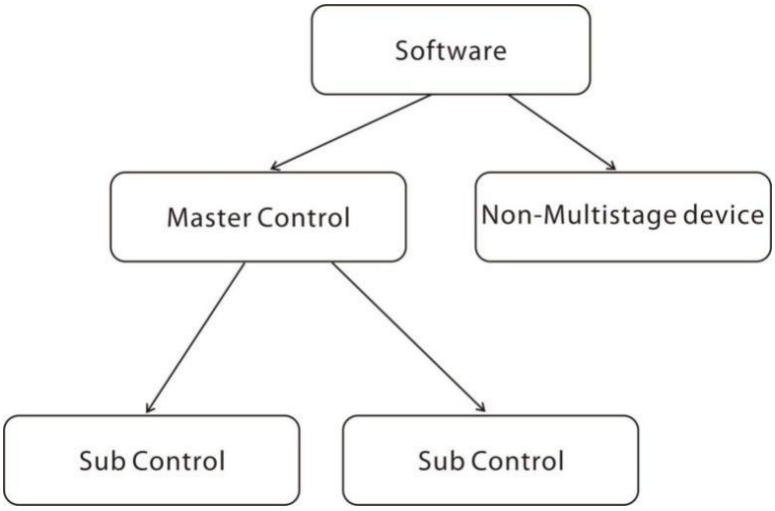
For example

```
DataType=1, SN=123456789, CmdID=295, CmdDesc=C:295:DATA UPDATE user
CardNo=    Pin=1  Password=234 Group=0 StartTime=0 EndTime=0   Name=
Privilege=0
DataType=1,SN=DDG7012017010600049, CmdID=295, CmdDesc=C:295:DATA
UPDATE DoorParameters ID=1Name=DoorAutoMatch Value=1 DevID=1
ID=1Name=DoorDrivertime Value=5 DevID=1
ID=1Name=DoorKeepOpenTimeZone Value=0 DevID=1
ID=1Name=DoorValidTZ       Value=1 DevID=1
ID=1Name=DoorSupperPassWord Value=  DevID=1
```

As for CmdFormat=1, the following prefix must be added to all C:XXX:COMMAND commands:

```
DataType=1,SN=%s,Priority=%d,CmdID=%s,CmdDesc=
```

11.15 Appendix 15 - Multi-level Control Design sketch of multi-level control



Description of multi-level control parameters -

- **MultiStageControlFunOn -** 1: Enable and 0: Disable.
  It enables or disables the multi-level control function. The default value is 0.
- **MasterControlOn -** 1: Enable the main controller and 0: Disable the main controller.
  It enables or disables the main controller. The default value is 0.
- **SubControlOn -** 1: Enable the sub-controller and 0: Disable the sub-controller.
  It enables or disables the sub-controller. The default value is 0.

Application scenario configurations -

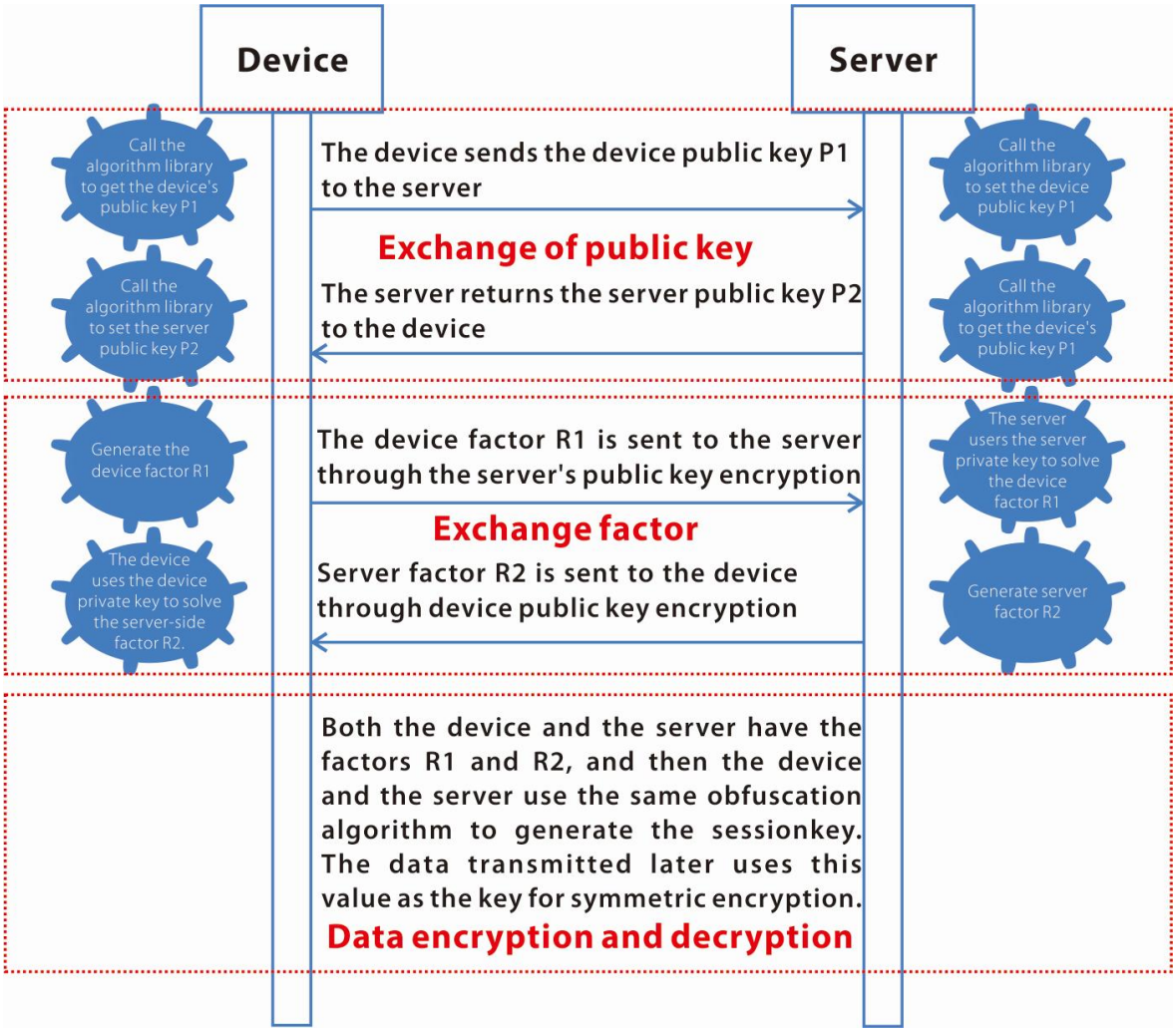
- **Non-multi-level device** (see Appendix 7).
  - When MultiStageControlFunOn, SubControlOn, and MasterControlOn do not exist or their values equal to the following values.
  - MultiStageControlFunOn=1 or MultiStageControlFunOn=0
  - SubControlOn=0
  - MasterControlOn=0
- **Sub-controller**
  - MultiStageControlFunOn=1
  - SubControlOn=1
  - MasterControlOn=0

- **Main controller**

  - MultiStageControlFunOn=1
  - MasterControlOn=1
  - SubControlOn=0

11.16 Appendix 16 - Data Encryption Key Exchange & Compatibility Scheme

Data Encryption Key Exchange Scheme



- **Algorithm:**

  The encryption algorithm libraries are encapsulated in a centralized manner. The device uses static algorithm libraries.

- **Scheme:**

  (a) The asymmetric encrypted public-private key is initialized when the device and server reconnect.

  (b) The device and the server exchange the public keys:

  - The device sends the device public key P1 to the server.
  - The server returns the server public key P2 to the device.
  - The public keys are exchanged. Both the device and the server own the public keys P1 and P2.

  (c) The device and the server exchange the factors:

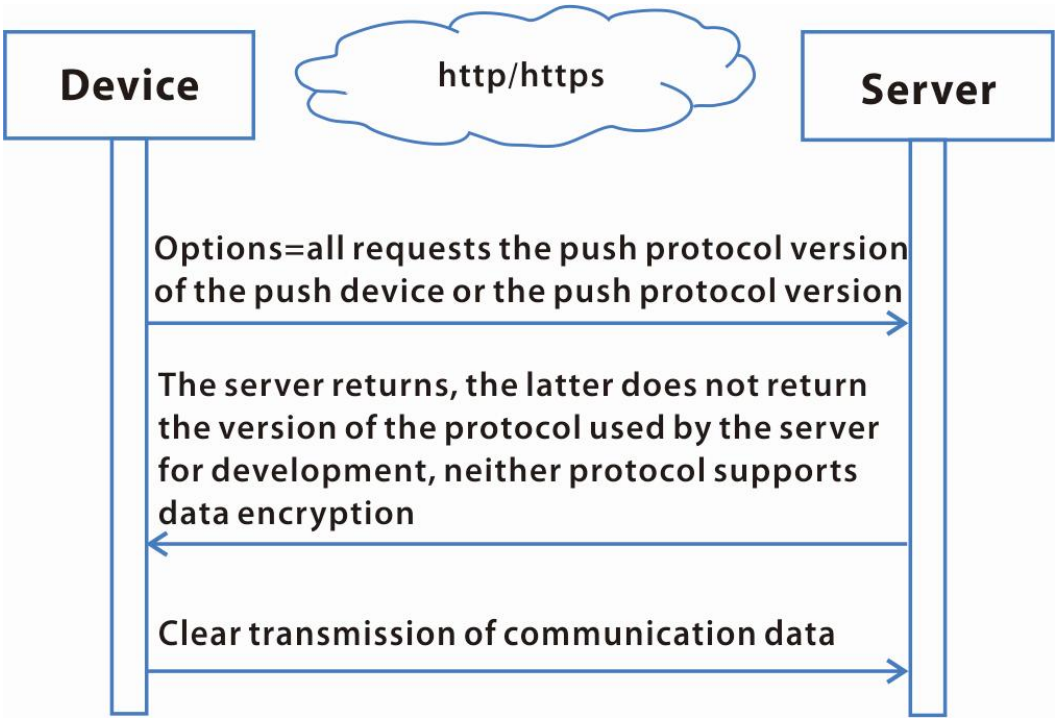
  - The device generates a factor R1 and encrypts it and then sends it to the server by using the server's public key.
  - The server decrypts the device factor R1 by using the server's private key.
  - The server generates a factor R2 and encrypts it and then sends it to the device by using the device's public key.
  - The server decrypts the server factor R2 by using the device's private key.
  - The factors are exchanged. Both the device and the server own the factors R1 and R2.

  (d) Both the device and the server own the factors R1 and R2. Then, the device and the server use the same algorithm to generate a sessionKey. All the data transmitted later uses this value as the private key for symmetric encryption.

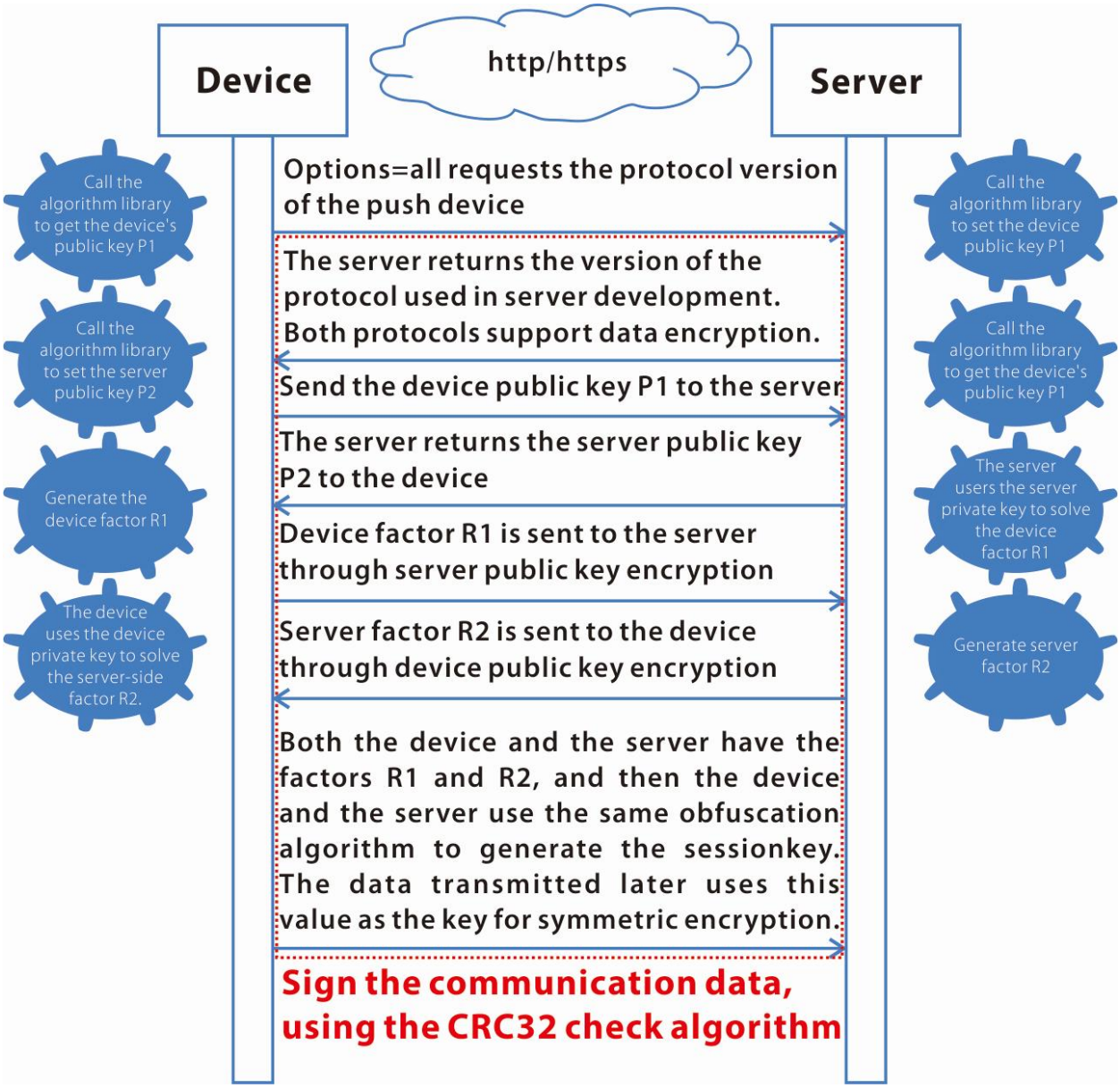
Compatibility Scheme

The device and the server are compatible based on the protocol version, as follows:

- **Case 1**



- **Case 2**



Note:

- The device determines whether to use HTTPS or HTTP based on the set server address.
- In the first request protocol header of the existing device, pushver field is added to the current communication protocol version number of the device, and PushProtVer is added to the data returned by the software to indicate which protocol version the software was developed on. The two protocol versions are compared, and the device and the server use the lower version for communication.

  Case 1: When protocol versions of both the server and the device are not supported, the data communication is transmitted in plain text.

  Case 2: If the protocol versions of both the server and the device are supported, the data is

encrypted for transmission.

The interaction sequence is as follows:

- The server and the device exchange their public keys P1 and P2 based on the new protocol.
- The server and the device exchange their factors R1 and R2 based on the new protocol.
- CRC32 check is performed for the communication data signature. When both the device and the server own the factors R1 and R2, the device and the server use the same algorithm to generate a sessionKey. All the data transmitted later uses this value as the private key for symmetric encryption.

11.17 Appendix 17 - Error Codes of Error Logs

Error code	Description
00000000	Successful
D01E0001	Face detection failed
D01E0002	Face shielded
D01E0003	Insufficient clarity
D01E0004	Two large face angle
D01E0005	Life detection failed
D01E0006	Failed to extract the template

According to the Error generator + Module + Type + Error value definition

Function (Bit Position)	Description
Error generator (1st bit)	D: Error code returned from the device S: Error code returned from the software
Module (2nd ~ 3rd bits)	**Device-end:** 01: PUSH communication module 02: Template processing module 03: Hardware interaction module 04: PULL communication module 05: Standalone communication module 06: Data relay module 07: License service module **Software-end:**

	To be confirmed
Type (4th bit)	E: ERROR
Error value (5th ~ 8th bits)	Integer data

11.18 Appendix 18 - Biometric Type Index Definition

Index	Type
0	Common
1	Fingerprint
2	Near-infrared face
3	Voiceprint
4	Iris
5	Retina
6	Palmprint
7	Finger vein
8	Palm vein
9	Visible light face
10	Visible light palm

Parameter	Description	Description
type	Biometric type Type 1-8 belongs to near-infrared; Type 9-10 belongs to visible light.	0-Common 1-Fingerprint 2-Near-infrared face 3-Voiceprint 4-Iris 5-Retina 6-Palmprint 7-Finger vein 8-Palm vein 9-Visible light face 10- Visible light palm

MultiBioPhotoSupport	Supports biometric photos	The type is defined bit by bit. Different types are separated by colons, 0 means not supported, 1 means supported. Such as: 0: 1: 1: 0: 0: 0: 0: 0: 0: 0: 0, indicating support for near-infrared fingerprint photo and face photo.
MultiBioDataSupport	Supports bio-templates	The type is defined bit by bit. Different types are separated by colons, 0 means not supported, 1 means supported. Such as: 0: 1: 1: 0: 0: 0: 0: 0: 0: 0: 0, indicating support for near-infrared fingerprint template and face template.
MultiBioVersion	Supported algorithms	The type is defined bit by bit. Different types are separated by colons, 0 means not supported, non-0 means supported version number. Such as: 0: 10: 0: 7: 0: 0: 0: 0: 0: 0: 0: 0: 0, indicating support for fingerprint algorithm10.0 and near-infrared face algorithm7.0.
MaxMultiBioDataCount	Supports maximum number of bio-templates.	The type is defined bit by bit. Different types are separated by colons, 0 means not supported, non-0 means supported maximum capacity. Such as: 0: 10000: 3000: 0: 0: 0: 0: 0: 0: 0: 0, indicating support for the maximum number of fingerprint templates is 10000 and the maximum number of near-infrared face templates is 3000.
MaxMultiBioPhotoCount	Supports maximum number of biometric photos.	The type is defined bit by bit. Different types are separated by colons, 0 means not supported, non-0 means supported maximum capacity. Such as: 0: 10000: 3000: 0: 0: 0: 0: 0: 0: 0: 0, indicating support for the maximum number of fingerprint photos is 10000 and the maximum number of near-infrared face photos is 3000.
MultiBioDataCount	The current capacity of bio-templates	The type is defined bit by bit. Different types are separated by colons. Such as: 0: 10000: 3000: 0: 0: 0: 0: 0: 0: 0: 0, indicating the current number of fingerprint templates is 10000 and the current number of near-infrared face templates is 3000.

MultiBioPhotoCount	The current capacity of biometric photos	The type is defined bit by bit. Different types are separated by colons. Such as: 0: 10000: 3000: 0: 0: 0: 0: 0: 0: 0: 0, indicating the current number of fingerprint photos is 10000 and the current number of near-infrared face photos is 3000.

11.19 Appendix 19 - Description of Extended Real-time Events

Normal events: The range is from 4000 to 4999.

Error code	Description
4000	Fire control normally open ends
4001	Normal communication of the personnel
4002	Configured successfully (elevator control)
4003	Successful loss report (elevator control)
4004	Successfully cancel the loss report (elevator control)
4005	Voice light on the door closing button (elevator control)
4006	Voice light on the door open button (elevator control)
4007	Green Code (health code)
4008	Mains power recovery event: the normal push event that the main power supply resumes
4009	Video intercom call
4010	Video intercom answering
4011	Video intercom call timeout
4012	Video intercom rejection
4013	Video intercom hangs up normally

Error events: The range is from 5000 to 5999.

Error code	Description
5000	The clutch is abnormal
5001	The fire control board is abnormal
5002	The fire control board is losing electricity

5003	The power supply is abnormal
5004	The signal synchronization is abnormal
5005	The motor current is too large
5006	The mainboard current is too large
5007	The mainboard current is overloaded
5008	Receives continuous opening signal
5009	The voice loading is abnormal
5010	The slave machine is offline
5011	The slave machine communication interface is disconnected
5012	The belt is broken
5013	The turnstile-opening solenoid is abnormal
5014	The drop-arm solenoid is abnormal
5015	The turnstile arm cannot be lifted
5016	The turnstile arm cannot be dropped
5017	Configuration failed (elevator control)
5018	Failed to report loss (elevator control)
5019	Failed to cancel the loss report (elevator control)
5020	Card data exception (elevator control)
5021	QR code is invalid
5022	The acquisition of the health code of Chongqing is abnormal

Alarm events: The range is from 6000 to 6999.

Error code	Description
6000	The fire control start is normally opened
6001	Yellow Code (health code)
6002	Red Code (health code)
6003	Unknown health status
6004	Abnormal status of health code of Chongqing
6005	Push event when the device record capacity reaches 90%

6006	Short circuit of reader (RS485)
6007	Short circuit of reader (Wiegand)
6008	Using emergency power supply, switch from AC power supply to DC power supply
6009	Using emergency power supply, switch from DC power supply to AC power supply
6010	Using emergency DC power supply, the power battery enters a low voltage state

11.20 Appendix 20 - Description of Channel Controller Upload Parameters

Parameter	Description
~DeviceName	Device name
MAC	Device physical address
~MaxAttLogCount	Maximum capacity of device records
UserCount	Number of current users
~MaxUserCount	Maximum capacity of device users
PhotoFunOn	Whether to support user photos
~MaxUserPhotoCount	Device user photo capacity
FingerFunOn	Device fingerprint function switch
FPVersion	Fingerprint algorithm version
~MaxFingerCount	Maximum capacity of device fingerprints
FPCount	Current number of fingerprints
FaceFunOn	Facial function switch
FaceVersion	Face algorithm version
~MaxFaceCount	Maximum capacity of device faces
FaceCount	Current number of faces
FvFunOn	Finger vein function switch
FvVersion	Finger vein algorithm version
~MaxFvCount	Maximum capacity of device finger veins
FvCount	Current number of finger veins
PvFunOn	Palm vein function switch

PvVersion	Palm vein algorithm version
~MaxPvCount	Maximum capacity of device palm veins
PvCount	Current number of palm veins
Language	Language
IPAddress	Device IP address
~Platform	Platform name
~OEMVendor	Manufacturer's name
FWVersion	Firmware version number
PushVersion	Push version number
RegDeviceType	Used to determine whether the device is allowed to access by the software
PassMode	Passing mode setting
MotorOpenSpeed	Gate opening speed
MotorCloseSpeed	Gate closing speed
PassWaySwipeMode	Swipe method
PassWayVolume	Volume level setting
PassWayUnmanned	Open duration time
PassWayCloseDelay	Gate closing delay time
PassWayCounter	Passing counter
PassWayWorkMode	Working mode
PassWayType	Passageway type
FireFightingMode	Fire mode
MotorOpenSlowStroke	Gate opening deceleration distance
MotorOpenSlowSpeed	Gate opening deceleration speed
MotorCloseSlowStroke	Gate closing deceleration distance
MotorCloseSlowSpeed	Gate closing deceleration speed
MemoryOpenGate	Memory function
VOLUME	Device volume
PassWayPreventClamp	Anti-pinch area setting
PassWayTailSet	Anti-tailgate setting

PassWayClutchAlarm	Clutch brake unlock
PassWayPassingTime	Passing time
PassWayAlarmOnOff	Alarm tone switch
PassWayInOutSwap	Entrance & exit exchange
PassWayExportSet	Intensity adjustment
PassWayClosePlace	Closing position
PassWayShowGateInfo	Display gate information
TZPassModeOn	Time period mode enable
PassWayReverseInto	False direction entry
PassWayClutchAngle	Clutch start angle
PassWayClampPro	Anti-pinch action setting
PassUnmannedRange	Open duration range
PassCloseDelayRange	Gate closing delay range
PassCloseDelay	Gate delay closing
VoiceToAlarm	Voice switching
IsSupportADScreen	Whether to support advertising screen
ADScreenFunOn	Advertising screen function switch
InPlayInterval	Entrance picture playback switching time
OutPlayInterval	Exit picture playback switching time

ZKTeco Industrial Park, No. 32, Industrial Road,

Tangxia Town, Dongguan, China.

Phone : +86 769 - 82109991

Fax : +86 755 - 89602394

www.zkteco.com