Communication protocol

PUSH SDK

Version: 2.2.1

Date: November 2015

Contents

1. Abstract

The Push protocol is a data protocol which is defined based on the Hyper Text Transmission Protocol (HTTP). Established on a TCP/IP connection, the Push protocol is applicable to the data interchange between a server and a ZK attendance machine or a ZK access control machine, and defines the transmission formats of data (including user information, biological recognition templates, and attendance records) and the command format for control equipment. Currently, ZK supports servers such as the WDMS, ZKECO, ZKNET, and ZKBioSecurity3.0, as well as third-party servers such as the ESSL from India.

1.1 Features

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

1.2 Encoding

Most data transmitted via the protocol is consisted of ASCII characters, but individual fields involve coding, for example, the user name. Therefore, the following rules are made for data of this type.

- For Chinese data, the GB2312 encoding is used.
- For other languages, the UTF-8 encoding is used.

Currently, the following data involves this encoding:

- User names in a user information table
- Content of the short messages in a short message table

1.3 Introduction to HTTP Protocol

Since the Push protocol is a data protocol defined based on the HTTP protocol, a brief introduction to the HTTP is given hereby. Skip this part if you are already familiar with it.

The HTTP is a request/response protocol. The format of a request sent by a client to a server is a request method, a URI and 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 contains the protocol version number of the message and a success code or error code. The following is an example.

A request from the client:
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
A response from the server:
HTTP/1.1 200 OK
Server: nginx/0.8.12
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

HTTP communication usually occurs under a TCP/IP connection. The default port is TCP 80, but other ports can also be used. However, the HTTP protocol might also be implemented via other protocols. Only reliable transmission is expected from the HTTP (note: HTTP is usually established on a transport layer protocol), therefore, any protocol providing such guarantee can be used.

2. Definitions

In this document, the format of definition reference is: ${ServerIP}

- ServerIP: The IP address of the server
- ServerPort: A port of the server
- XXX: An unknown value
- Value123……: Value 123…value n
- Required: Mandatory
- Optional: Selectable
- SerialNumber: Serial number
- NUL: Null ()
- SP: A space
- LF: A line break ()
- HT: A tab character ()
- DataRecord: A data record
- CmdRecord: A command record
- CmdID: The ID of a command
- CmdDesc: Command description
- Pin: ID
- Time: Attendance time
- Status: Attendance status
- Verify: Verification mode
- Workcode: A workcode
- Reserved: A reserved field
- OpType: An operation type
- OpWho: An operator
- OpTime: Operation time

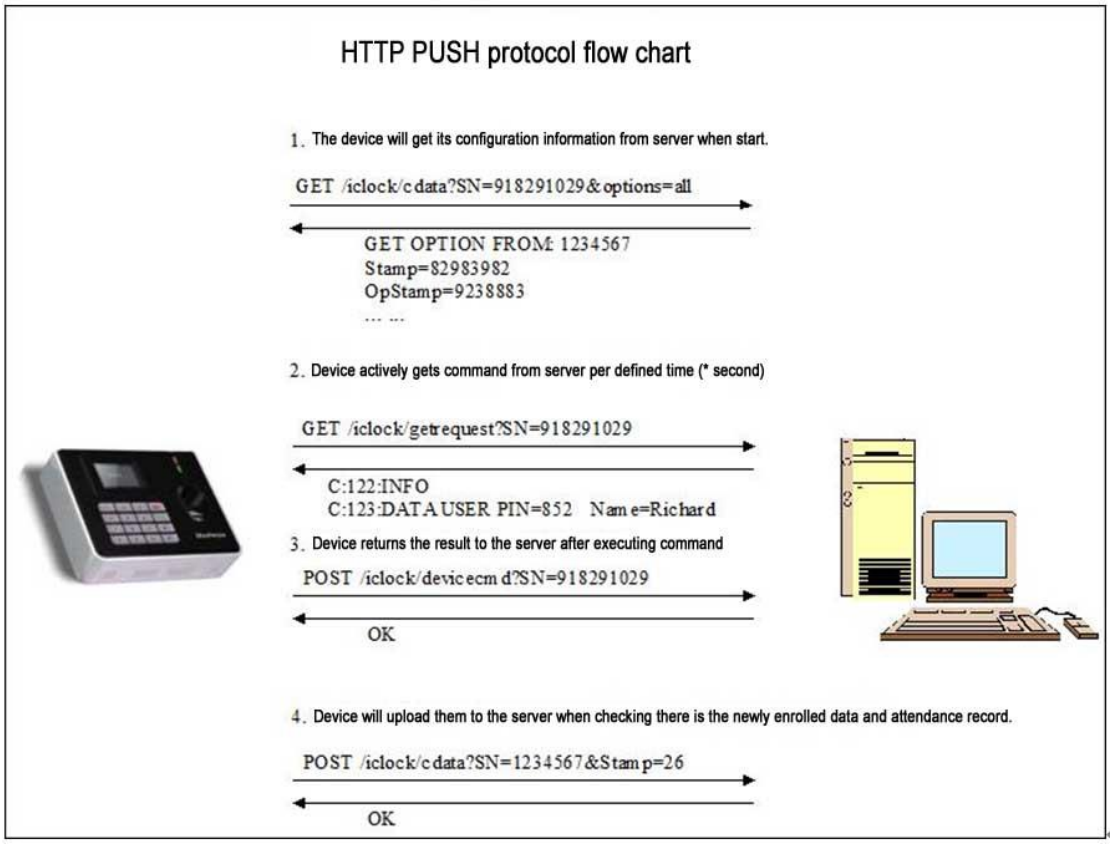
- BinaryData: A binary data flow
- TableName: The name of a data table
- SystemCmd: A system command
- Key: A key
- Value: A value
- FilePath: A file path
- URL: A resource location

3. Functions

The following functions supported by the Push protocol are described from the view of a client.

4. Process

Between a client and a server that both use the Push protocol, a request of "Initialization Information Exchange" must be firstly initiated by the client successfully and then other functions can be used, such as uploading data, obtaining server commands, uploading update information, and replying server commands. These functions are not necessarily in order but dependent to the development of the client application, as shown in the figure below.



5. Initialization Information Exchange

The client initiates a request to and sends corresponding configuration information to the server, and the server replies to the client with corresponding configuration information after receiving the request. Only when the client obtains the corresponding configuration information, the exchange is successful. The configuration information is exchanged in a specified format as shown below:

A request message from the client:

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

Annotation:

HTTP request method: GET method
URI: /iclock/cdata
HTTP protocol version: 1.1
Client configuration information:
SN: ${Required} Client's serial number
options: ${Required} Obtaining server configuration parameters, and only the value "all" is available currently
pushver: ${Optional} version of the Push protocol supported by a newly-developed client software, and is of the 2.2.14 version or higher
language: ${Optional} languages supported by the client, better supported by a newly developed client so that the server knows the language the current equipment uses. See "Appendix 2". pushcommkey: ${Optional} ciphertext information for binding the client and the server, allowing the software to determine whether the equipment is authorized or not. The value differs for different equipment. This parameter needs to be supported by the client only when it is supported by the server.
Host header field: ${Required}
Other header fields: ${Optional}

A normal response from the server

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

GET OPTION FROM: ${SerialNumber}${LF}${XXX}Stamp=${XXX}${LF}ErrorDelay=${XXX}${LF}Del
ay=${XXX}${LF}TransTimes=${XXX}${LF}TransInterval=${XXX}${LF}TransFlag=${XXX}${LF}TimeZ
one=${XXX}${LF}Realtime=${XXX}${LF}Encrypt=${XXX}${LF}ServerVer=${XXX}
```

Annotation:

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.

Server configuration information: The description in the first line must be this: GET OPTION FROM: ${SerialNumber}, with the ${LF} separating configuration information.

${SerialNumber} is the serial number of the request initiated by the client. The configuration information is in key=value pairs, with a ${LF} separating two configurations.

${XXX}Stamp: Timestamps for all kinds of data types, currently supporting the following:

${XXX}	Data type
ATTLOG	Attendance record
OPERLOG	Operation log
ATTPHOTO	Attendance photo

Purpose of timestamp mark design: When the client uploads data, the corresponding timestamp mark is uploaded. The server is responsible for recording this mark. When the equipment reboots, the client initiates a request for initialization of information exchange, and the server sends a series of marks to the client, realizing the function of resuming transmission from breakpoint.

Timestamp mark flaw: As time modification is permitted and the uncertainty of time change is possible, the client may not correctly determine which data has been uploaded to the server and which has not, and this leads to server data loss.

Application of timestamp on server: Currently, the server has only one application of the timestamp mark. When the server needs to reupload all corresponding data, it sets the corresponding timestamp mark to 0. See this function at "Get Command – Control Command – Check Data Update".

Timestamp discard at client side: In the Push design for new framework firmware, no timestamp is used to mark a cut-off point of data uploading. However, for compatibility with old servers, timestamp marks are also sent. Actually, it realizes only the function of data reuploading when the mark is set to 0, so the server does not need to differentiate whether the client has discarded a timestamp or not.

ErrorDelay: Interval time for the client to reconnect to the server after networking connection failure, and the recommended value is 30~300s.

Delay: Interval for the client to connect to the server when the networking is normal (s), that is, the function of requesting "Get Command" by client. The recommended value is 2~60s. When a rapid response is required, a smaller value can be set, but this will increase the pressure on the server.

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: Interval for the client to check and transmit new data (in minute), and no check is performed when it is set to 0. For example, TransInterval=1

TransFlag: Identifying the data to be uploaded by the client automatically to the server, and two formats are supported. Format I: TransFlag=1111000000……, each digit representing a data type. 0 for forbidding automatic uploading of this data type, 1 for allowing automatic uploading of this data type. Data type on each digit

1	Attendance record
2	Operation log

3	Attendance photo
4	Enrolling a new user
5	Changing user information
6	Enrolling a new fingerprint
7	Changing a fingerprint
8	Fingerprint image
9	New enrolled face
10	User picture

Format II: TransFlag=TransData AttLog${HT}OpLog${HT}AttPhoto……

Data types marked by strings

AttLog	Attendance log
OpLog	Operation log
AttPhoto	Attendance photo
EnrollUser	Enrolling a new user
ChgUser	Changing user information
EnrollFP	Enrolling a new fingerprint
ChgFP	Changing a fingerprint
FPImag	Fingerprint image
FACE	New enrolled face
UserPic	User picture

During new client development: Please support both formats simultaneously. When the server sends data in format I with all values set to 0 (TransFlag=0000000000), only uploading attendance photos is supported. During new server development: Only format II needs to be supported.

TimeZone: Specify the time zone where the server is located, primarily for server time synchronization. See the Date header field in [Get Command](#downloadcmd). This value is an integer and designed to support a whole time zone, half time zone and 1/4 time zone. For -12 < Time Zone < 12, it is a whole time zone in the unit of hour. For example, TimeZone=4 means the East 4 zone. For TimeZone > 60 or TimeZone < -60, it can mean a half time zone or 1/4 time zone in the unit of minute. For example, TimeZone=330 means a half of the East 5 time zone.

Realtime: Whether the client transmits new records in real time.1 means that data is transmitted to the server as soon as it is generated, while 0 means data is transmitted at the time defined by the TransTimes and TransInterval.

Encrypt: Whether to transmit data after encryption. It is reserved.

ServerVer: Protocol version and time (format to be determined), which are supported by the server, and it must be set to 2.2.14 or above for a newly-developed server.

Application for client: This parameter allows the client to know whether the server currently connected supports some new features. When the server does not send them, the client assumes by default that the server supports only the initial protocol of the 1.X.X. version. Currently, the following features are differentiated through this parameter:1. When data is uploaded, its timestamp mark is also uploaded, and different versions have different timestamp mark descriptions. For details, see the "Upload Data" command.

Example:

A request from the client:

```
GET /iclock/cdata?SN=0316144680030&options=all&pushver=2.2.14&language=83&pushcommkey=4a9594af164f2b9779b59e8554b5df26 HTTP/1.1
Host: 58.250.50.81: 8011
User-Agent: iClock Proxy/1.09
Connection: close
```

Accept: */*
A response from the server:
HTTP/1.1 200 OK
Server: nginx/1.6.0
Date: Fri, 03 Jul 2015 06: 53: 01 GMT
Content-Type: text/plain
Content-Length: 190
Connection: close
Pragma: no-cache
Cache-Control: no-store

GET OPTION FROM: 0316144680030
ATTLOGStamp=None
OPERLOGStamp=9999
ATTPHOTOStamp=None
ErrorDelay=30
Delay=10
TransTimes=00: 00;14: 05
TransInterval=1
TransFlag=TransData AttLog OpLog AttPhoto EnrollUser ChgUser EnrollFP ChgFP UserPi
c
TimeZone=8
Realtime=1
Encrypt=None

6. Uploading Update Information

This function multiplexes the Download Command request and adds parameters in its URL to mainly upload the client's firmware version number, number of enrolled users, number of enrolled fingerprints, number of attendance records, IP address of equipment, fingerprint algorithm version, face algorithm version, number of faces required for face enrollment, number of enrolled faces, and marked information about functions supported by the equipment.

A request message from the client:

```
Get /iclock/getrequest?SN=${SerialNumber}&INFO=${Value1},${Value2},${Value3},${Value4},${Value5},${Value6},${Value7},${Value8},${Value9},${Value10}
Host: ${ServerIP}: ${ServerPort}
……
```

Annotation:

```
HTTP request method: GET method
URI: /iclock/getrequest
HTTP protocol version: 1.1
Client configuration information:
SN: ${Required} Client's serial number
${Value1}: Firmware version number
${Value2}: Number of enrolled users
${Value3}: Number of enrolled fingerprints
${Value4}: Number of attendance records
${Value5}: IP address of Equipment
${Value6}: Version of fingerprint algorithm
${Value7}: Version of face algorithm
${Value8}: Number of faces required for face enrollment
${Value9}: Number of enrolled faces
${Value10}: Identifier of functions supported by the equipment in the format of 101 with every digit representing a function, 0—Not supporting this function, 1—Supporting this function.
Description of function on each digit
                    1       Fingerprint function
                    2       Face function
                    3       User photo function
Host header field: ${Required}
Other header fields: ${Optional}
```

For server responses, see Download Command.

Example

```
A request from the client:
GET /iclock/getrequest?SN=0316144680030&INFO=Ver%202.0.12-20150625,0,0,0,192.168.16.2
```

7,10,7,15,0,111 HTTP/1.1
Host: 58.250.50.81: 8011
User-Agent: iClock Proxy/1.09
Connection: close
Accept: */*

A response from the server:
HTTP/1.1 200 OK
Server: nginx/1.6.0
Date: Tue, 30 Jun 2015 01: 24: 26 GMT
Content-Type: text/plain
Content-Length: 2
Connection: close
Pragma: no-cache
Cache-Control: no-store

OK

7. Uploading Data

The data to be uploaded automatically can be set on the server. (For details, see the "TransFlag" parameter in "Initialization Information Exchange".)

7.1 Uploading Mode

Realtime uploading

Interval uploading

Timed uploading

Realtime uploading: If realtime uploading is supported, the interval is uploaded in real time. This is supported by the equipment by default and can be controlled by the server. (For details, see the "Realtime" parameter in "Initializing Information Exchange").
Interval uploading: The server can control specific interval time. (For details, see the "TransInterval" parameter in "Initializing Information Exchange".)
Timed uploading: The server can control specific upload timing. (For details, see the "TransTimes" parameter in "Initializing Information Exchange".)

7.2 Uploading Attendance Record

A request message from the client:

```
POST /iclock/cdata?SN=${SerialNumber}&table=ATTLOG&Stamp=${XXX} HTTP/1.1
Host: ${ServerIP}: ${ServerPort}
Content-Length: ${XXX}
……


 ${DataRecord}
```

Annotation:

```
HTTP request method: POST method
Used URI: /iclock/cdata
HTTP protocol version: 1.1
Client configuration information:
SN: ${Required} Serial number of the client     table=ATTLOG: ${Required} Indicating that the up
loaded data is attendance records.
Stamp: ${Optional} Latest timestamp at which the attendance record is uploaded to the server.
 (For details, see the "Stamp" or "ATTLOGStamp" parameter in "Initializing Information Exchang
e".)
Host header field: ${Required}
Content-Length header field: ${Required}
Other header fields: ${Optional}
Request entity: ${DataRecord}, attendance record data, in the following format:
```

${Pin}${HT}${Time}${HT}${Status}${HT}${Verify}${HT}${Workcode}${HT}${Reserved}${HT}${Reserved}
Note:
${Time}: Verification time, in the format of XXXX-XX-XX XX: XX: XX. For example, 2015-07-29 11: 11: 11, with ${LF} used to connect multiple records.

A normal response message from the server:

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

OK: ${XXX}
```

Annotation:

HTTP status line: Defined with standard HTTP protocol
HTTP response header field:
Content-Length header field: According to the HTTP 1.1, this header field is generally used to specify the data length of the response entity. If the response entity size is uncertain, head fields of Transfer-Encoding: chunked, Content-Length and Transfer-Encoding are supported, all of which are standard definitions of HTTP and are not described in details here.
Response entity: When the server normally receives data and successfully processes data, OK: ${XXX} is replied. ${XXX} represents the number of records successfully processed. When an error occurs, the error description is replied.

Example

A request from the client:

```
POST /iclock/cdata?SN=0316144680030&table=ATTLOG&Stamp=9999 HTTP/1.1
Host: 58.250.50.81: 8011
User-Agent: iClock Proxy/1.09
Connection: close
Accept: */*
Content-Length: 315

1452    2015-07-30 15: 16: 28 0   1   0   0   0
1452    2015-07-30 15: 16: 29 0   1   0   0   0
1452    2015-07-30 15: 16: 30 0   1   0   0   0
1452    2015-07-30 15: 16: 31 0   1   0   0   0
1452    2015-07-30 15: 16: 33 0   1   0   0   0
1452    2015-07-30 15: 16: 34 0   1   0   0   0
1452    2015-07-30 15: 16: 35 0   1   0   0   0
8965    2015-07-30 15: 16: 36 0   1   0   0   0
8965    2015-07-30 15: 16: 37 0   1   0   0   0
```

A response from the server:

```
HTTP/1.1 200 OK
Server: nginx/1.6.0
Date: Thu, 30 Jul 2015 07: 25: 38 GMT
Content-Type: text/plain
Content-Length: 4
```

```
Connection: close
Pragma: no-cache
Cache-Control: no-store

OK: 9
```

7.3 Uploading Attendance Photo

The configuration ServerVer parameter sent by the server for initialization information exchange is greater than or equal to version 2.2.14.

A request message from the client:

```
POST /iclock/cdata?SN=${SerialNumber}&table=ATTPHOTO&Stamp=${XXX} HTTP/1.1
Host: ${ServerIP}: ${ServerPort}
Content-Length: ${XXX}
……

${DataRecord}
```

Annotation:

```
HTTP request method: POST method
URI: /iclock/fdata or /iclock/cdata
HTTP protocol version: 1.1
Client configuration information:
SN: ${Required} Serial number of the client     table=ATTPHOTO: ${Required}
Stamp: ${Optional} Latest timestamp at which the attendance photo is uploaded to the server.
 (For details, see the "ATTPHOTOStamp" parameter in "Initializing Information Exchange".) Host
header field: ${Required}
Content-Length header field: ${Required}
Other header fields: ${Optional}
Request entity: ${DataRecord}, attendance photo data, in the following format:
        PIN=${XXX}${LF}SN=${SerialNumber}${LF}size=${XXX}${LF}CMD=uploadphoto${NUL}${Bi
naryData}
Note:
        PIN=${XXX}: Filename of the attendance photo, with only the jpg format supporte
d currently.
        SN=${XXX}: Serial number of the client.
        size=${XXX}: Original size of the attendance photo
        ${BinaryData}: Binary dataflow of the original photo. Transmission of multiple recor
ds is not supported.
```

A normal response message from the server:

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

```
OK
```

Annotation:

```
HTTP status line: Defined with standard HTTP protocol
HTTP response header field:
Content-Length header field: According to the HTTP 1.1, this header field is usually used to spe
cify the data length of the response entity. If the response entity size is uncertain, head fields of
 Transfer-Encoding: chunked, Content-Length and Transfer-Encoding are supported, all of whic
h are standard definitions of HTTP and are not described in details here.
Response entity: When the server normally receives data and successfully processes dat
a, OK is replied. When an error occurs, the error description is replied.
```

Example:

```
A request from the client:
POST /iclock/cdata?SN=0316144680030&table=ATTPHOTO&Stamp=9999 HTTP/1.1
Host: 58.250.50.81: 8011
User-Agent: iClock Proxy/1.09
Connection: close
Accept: */*
Content-Length: 1684

PIN=20150731103012-123.jpg  SN=0316144680030    size=9512   CMD=uploadphoto${NUL}
${BinaryData}

A response from the server:
HTTP/1.1 200 OK
Server: nginx/1.6.0
Date: Thu, 30 Jul 2015 07: 25: 38 GMT
Content-Type: text/plain
Content-Length: 2
Connection: close
Pragma: no-cache
Cache-Control: no-store

OK
```

7.4 Uploading Operation Record

The configuration ServerVer parameter sent by the server for initialization information exchange is greater than or equivalent to version 2.2.14.

A request message from the client:

```
POST /iclock/cdata?SN=${SerialNumber}&table=OPERLOG&Stamp=${XXX} HTTP/1.1
Host: ${ServerIP}: ${ServerPort}
Content-Length: ${XXX}
……
```

${DataRecord}

Annotation:

HTTP request method: POST method
URI: /iclock/cdata
HTTP protocol version: 1.1
Client configuration information:
SN: ${Required} Serial number of the client
table=OPERLOG: ${Required}
Stamp: ${Optional} Latest timestamp at which the attendance record is uploaded to the server. (For details, see the "OPERLOGStamp" parameter in "Initializing Information Exchange".) Host header field: ${Required}
Content-Length header field: ${Required}
Other header fields: ${Optional}
Request entity: ${DataRecord}, operation record data, in the following format:
OPLOG${SP}${OpType}${HT}${OpWho}${HT}${OpTime}${HT}${Value1}${HT}${Value2}${HT}${Value3}${HT}${Reserved}
${OpType}: Operation code. See Appendix 3.
${Value1}, ${Value2}, ${Value3}, ${Reserved}: Operand 1, 2, 3 and 4. See Appendix 4. Note: ${LF} is used to connect multiple records.

A normal response message from the server:

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

OK: ${XXX}

Annotation:

HTTP status line: Defined with standard HTTP protocol
HTTP response header field:
Content-Length header field: According to the HTTP 1.1, this header field is generally used to specify the data length of the response entity. If the response entity size is uncertain, head fields of Transfer-Encoding: chunked, Content-Length and Transfer-Encoding are supported, all of which are standard definitions of HTTP and are not described in details here. Response entity: When the server normally receives data and successfully processes data, OK: ${XXX} is replied. ${XXX} represents the number of records successfully processed. When an error occurs, the error description is replied.

Example

A request from the client:
POST /iclock/cdata?SN=0316144680030&table=OPERLOG&Stamp=9999 HTTP/1.1
Host: 58.250.50.81: 8011
User-Agent: iClock Proxy/1.09
Connection: close
Accept: */*
Content-Length: 166

```
OPLOG 4 14 2015-07-30 10: 22: 34 0   0   0   0

A response from the server:
HTTP/1.1 200 OK
Server: nginx/1.6.0
Date: Thu, 30 Jul 2015 07: 25: 38 GMT
Content-Type: text/plain
Content-Length: 3
Connection: close
Pragma: no-cache
Cache-Control: no-store

OK: 1
```

7.5 Uploading User Information

The configuration ServerVer parameter sent by the server for initialization information exchange is greater than or equals to version 2.2.14.

A request message from the client:

```
POST /iclock/cdata?SN=${SerialNumber}&table=OPERLOG&Stamp=${XXX} HTTP/1.1
Host: ${ServerIP}: ${ServerPort}
Content-Length: ${XXX}
……

${DataRecord}
```

Annotation:

```
HTTP request method: POST method
URI: /iclock/cdata
HTTP protocol version: 1.1
Client configuration information:
SN: ${Required} Serial number of the client
table=OPERLOG: ${Required}
Stamp: ${Optional} Latest timestamp at which user information is uploaded to the server. (For details, see the "OPERLOGStamp" parameter in "Initializing Information Exchange".)
Host header field: ${Required}
Content-Length header field: ${Required}
Other header fields: ${Optional}
Request entity: ${DataRecord}, fingerprint template data, in the following format:
USER${SP}PIN=${XXX}${HT}Name=${XXX}${HT}Pri=${XXX}${HT}Passwd=${XXX}${HT}Card=${XXX}${HT}Grp=${XXX}${HT}TZ=${XXX}
Note: Name=${XXX}: User name. When the equipment is in Chinese, the GB2312 code is used. When the equipment is in another language, the UTF-8 code is used.
Card=${XXX}: Card number, supporting only two formats.
a. hexadecimal data, in the format of [%02x%02x%02x%02x], representing the first, second, third and fourth digit from left to right. For example, if the card number is 123456789, this is Card=[15CD5B07]
```

b. string data. If the card number is 123456789, this is: Card=123456789 TZ=${XXX}: Information on number of the time period used by the user, in the format of XXXXXXXXXXXXXXX. Digits 1-4 describe whether the group time period is used, digits 5-8 description use personal time period 1, digits 9-12 description use personal time period 2, and digits 13-16 description use personal time period 3.
For example: 0000000000000000 represents use of the group time period.
0001000200000000 represents using personal time period, with personal time period 1 using the time information of time period numbered 2.
0001000200010000 represents using personal time period, with personal time period 1 using the time information of time period numbered 2 and personal time period 2 using the time information of time period numbered 1.
${LF} is used to connect multiple records.

A normal response message from the server:

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

OK: ${XXX}
```

Annotation:

HTTP status line: Defined with standard HTTP protocol
HTTP response header field:
Content-Length header field: According to the HTTP 1.1, this header field is generally used to specify the data length of the response entity. If the response entity size is uncertain, head fields of Transfer-Encoding: chunked, Content-Length and Transfer-Encoding are supported, all of which are standard definitions of HTTP and are not described in details here.
Response entity: When the server normally receives data and successfully processes data, OK: ${XXX} is replied. ${XXX} represents the number of records successfully processed. In case of an error, an error description is replied.

Example:

```
POST /iclock/cdata?SN=0316144680030&table=OPERLOG&Stamp=9999 HTTP/1.1
Host:  58.250.50.81: 8011
User-Agent:  iClock Proxy/1.09
Connection:  close
Accept:  */*
Content-Length:  166

USER PIN=36234  Name=36234  Pri=0   Passwd= Card=133440 Grp=1   TZ=0001000000000000
USER PIN=36235  Name=36235  Pri=0   Passwd= Card=133441 Grp=1   TZ=0001000000000000

A response from the server:
HTTP/1.1 200 OK
Server:  nginx/1.6.0
Date:  Thu, 30 Jul 2015 07: 25: 38 GMT
Content-Type:  text/plain
Content-Length:  4
Connection:  close
```

```
Pragma:  no-cache
Cache-Control:  no-store

OK: 2
```

7.6 Uploading Fingerprint Template

The configuration ServerVer parameter sent by the server for initialization information exchange is greater than or equal to version 2.2.14.

A request message from the client:

```
POST /iclock/cdata?SN=${SerialNumber}&table=OPERLOG&Stamp=${XXX} HTTP/1.1
Host: ${ServerIP}: ${ServerPort}
Content-Length: ${XXX}
……

${DataRecord}
```

Annotation:

```
HTTP request method: POST method
URI: /iclock/cdata
HTTP protocol version: 1.1
Client configuration information:
SN: ${Required} Serial number of the client
table=OPERLOG: ${Required}
Stamp: ${Optional} Latest timestamp at which the fingerprint template is uploaded to the serv
er. (For details, see the "OPERLOGStamp" parameter in "Initializing Information Exchange".)
Host header field: ${Required}
Content-Length header field: ${Required}
Other header fields: ${Optional}
Request entity: ${DataRecord}, fingerprint template data, in the following format:
FP${SP}PIN=${XXX}${HT}FID=${XXX}${HT}Size=${XXX}${HT}Valid=${XXX}${HT}TMP=${XXX}
Note:
Size=${XXX}: Length after base64 coding of the fingerprint template
TMP=${XXX}: When the fingerprint template is transmitted, base64 coding needs to be conduc
ted for the original binary fingerprint template. ${LF} is used to connect multiple records.
```

A normal response message from the server:

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

OK: ${XXX}
```

Annotation:

```
HTTP status line: Defined with standard HTTP protocol
```

HTTP response header field:
Content-Length header field: Based on HTTP 1.1, this header field is usually used to specify the data length of the response entity. If the response entity size is uncertain, head fields of Transfe r-Encoding: chunked, Content-Length and Transfer-Encoding are supported, all of which are st andard definitions of HTTP and are not described in details here.
Response entity: When the server normally receives data and successfully processes data, OK: ${XXX} is replied. ${XXX} represents the number of records successfully processed. In case of a n error, an error description is replied.

Example

A request from the client:
POST /iclock/cdata?SN=0316144680030&table=OPERLOG&Stamp=9999 HTTP/1.1
Host: 58.250.50.81: 8011
User-Agent: iClock Proxy/1.09
Connection: close
Accept: */*
Content-Length: 4950

FP PIN=2 FID=0 Size=1124 Valid=1 TMP=SghTUzIxAAADS00ECAUHCc7QAAAnSnkBAAAAg
/YUfEsyAIEPHgH6ALFHRQBBAPkP8wBAS2UPEwBTACYPe0tYAHkljACuAHdIeQBtAGwEUAB1S20
DhAB+AK8EXUuPAOoPJABwANVENQDCANsPZQDbSx8PbwDeACwPz0vjAJ8PdwArAPFELAD5
AMwPvQASSvMKMgAwAQkPSUE2DkcXQ0uCQ1B4AJT7GZuC3GyNySrvjoKT7X77SkYkB9L6MQh
MCV5G1PUR+T0FfPiGTqMABQHp+XgBhclzg397xf0iD5CkQAXvErv3q4PQZ940xfmXzBb5bcher2
e7PQkLAXyf8gJ78nP7iwFIQmrXcwKn31LfiwoBIDQBAjrbrAdLOBFwwv8ExVYStv7ABQBOE7+JCE
t/GIBs/4SqDwPoJ4yHwMDEBcHDicB/DQCrLkbAwgnFcnwGAHn2g4ilCQCsNoPCnm4RS75CjJ0r
gwFqwS4HADFDZmwEAwJcRjrABwA1jWTCtMX+whMAw4+JwYnC+//Aw3pCwsK0wRlAxFKJB
8PAsfzCw3WLwaENAz9besHBi4eqBwOPW4PBwsEoygBxKnvCwcLAwqx4WFIB02WGwl86boCL
w4b/ZMFx3ADXJ4FSi3X/wqt2whjD/wkAc261eMKKigQBG249lwUD0nP9QxEAdbZwwYt5wcHC
wME6wcMuBQCeeANl1gCQMWjC/v+DwL90UbQCAIF7cMLNAJo2CMHA+/xTzQCCy2jA/pPAB
MXTh2ZXEQEfnkCn/8OLwWVdSQMB18tDiwsBF8s9RZFcFUvqza3C/3gAwMOMw1lxZ8laxeXM7
cFyxMDCpLvAdCDD/n4HAKMcHsMH/gsAcNreO/z/tfxC/wsAdBvk/bf9/T5KFAAM4KfPxMHFjP/C
B8DDi8FzCgBs4p+EwhHBGwDr4qQEUcA2wsPBwMF4B8DCEML+wAQAei/wH1wB6/Gma8AFxl
2lwIPAg1gYxej46Ut8n8PBwqrBbgYJEHsO4vk8/cMkBRCIF1NpxhCGe0jBERDTMGxri4/CrGb/FBA
aMqiLccDAwMTFAMLAi8PB/8HAA9WLOgj/DxDPPqtA/8GJtcRCDBDEj55Bi/7JyZMEEdFRQ7T4
FP PIN=2 FID=1 Size=2120 Valid=1 TMP=T3dTUzIxAAAGNDsECAUHCc7QAAAeNWkBAAAA
htlDsjQjAKlPYgDiAHo7NgA1AHAPZgBSNKQPdwBWAOAPVjRYABMPsACiADM76wB8ALkO2QB
4NFgPVQCCAP0O2TSTALoOYABeAEo7TgCdAFMPwQGmNNgNwgCiAH0PnDSpAEQPCgFuAEI5
eQCtAEcPlwCpNMoNEgGwAPkNvTSyADoP+ABxAEY/KgC1ANUPVQC/NMMP7gC8AAwNHTTA
AM0O9AAEAEc5JgDCANQO7QDKNN4OzgDOAH4OHjTQANUODgAdAMc6ZgDgAEsPNgDmN
MMOKQDiAKMOEjTlAOUOzgAhAM86QQDmAGUPRQDoNFIPaQDwAAIP6DT3AD8OBgEzALk6u
AD3AMQOCAAHNTQOPwAHAaEPDTUGAUlOtQDMAUY7oQAMAU8PEAAJNbQO8gAZAXgPxD
QfATsOCgHaAcQ63AApATYOZAAqNToMmAAvAY0M/TQuAcANyQD1AcY60gA2AT8OaAA/NcE
LOgA7ASYPlDQ8AWQLJQCEAfQ7hQBEAd8OKwBCNb8ORABMASoPdDROAeUOnACUAco5Lv0j
b0N/vw8xbVZpOl8G9+JPA0Zbg6sF2hafgw2db/Kq3hanMJLdJ1p/HQLu48rxWbIqlj8Lee9U5wA
SmAL+8DIEdPjjI1MEKQj1H1OD0qPInf2aGYFzANzaklW6/UoNkQruW6gFTYfW/PMF5zDAjd2O3P
osZbvLGlVugNb5YXuLNMx1+QARCTT6k7VQf7qL1IpZi2c0ufv9+NHwGXoLNOR66PcwC7wAWjs
UDD0UBfargLTBFRd1CKH4be/q65T3cYRJ+3fqD7LgB4KAmYEciic2SSdlEZmBvBlfM/iXpQJeAHuJ
r7rAC04HdXxwdkBTDH5Bglr9MH1HtmT/BYXRkmT/c7TgcsH23gAPAw2o9lWBgqaEzH27xWgGL
RFyFNOMc7QwfgGD9f4l/i+/dH0pivH+2AaHtkgGpfi9efR7vsJofaXaLA5M5Eu+oAJxBhkGHYN3sb
SN2IMdCCwHZ7FUej1fyfXU8mW17ByJhEmaQBKYMGKTkXwN/XAXnzZXgG6IPQ3n/JE4hAmKg
Z4bpGpP24wPjfRRa9Hpz8AElEwBAungoQU0ZwUAwAwAaAcidMBEwDgKAH0IlctEQv4SAHzOE
zP0/0NAW8L9xgCfPx3/CgDDC+L/U8tKDQBnDAk4RvgMw/3AFgCNyRcwH1VYVDyfB8XQChnBV

f8SADfU/cbK/v7/O8H8BcL4y0gDAOQkLToLBgMb/TwwQQnFuCAZS1X+CwAm4vdKyf/AKwUA
X+2AdiEBUS4A/yeD/VdyZWADADMztMAeNFc1APwwMDvA+WdZW//AxQTFND9ZZAgApVcw
OlXG9AYAelgg/ur/EzRUWRD+/v31wDEHTcE4GgADm+lV9Pz9wP0wKTpDxnhoEAAHaOL9O/vK
/v7+/jQHxAVuZ5X9wgUAsaw0TCYBCHLkR/4FISx1Oh8AAX7wBkcozy8wRTbC/j7CQBgCARCAVs
HBABy0X40EAOuAg3YANEWDaZ3CCMVYjwP6RjYEAD5KXI4wAUSPVlYFxUqJePxPHQAEkRL++G
P8+/4j/v+O/8fK/cFW/ykOxd+QDv5Xc8DCwJULBmufSTj//cDqBQZ/nlNBBwBQZEz4y/8yBADDp
vHA+CQBl6tDUlE+wccHwfsUAJyrhVVdyv3E///9/wX7+2MCAQevRsHDAQqbQf9SBgB4dUZD9A
sAv7U3QQXAxUQEALq2QGjdAB+Jx/7///7+0vz4HcH+/8H9/wX++TcBCL9AwQXF7cZ0/sLBCAD
9AkD79Xb/IAACygUoxsvB/f/9+/04/fjK/v3AwP//Bf3E9MH9/v3+HcUP0PfD/MD+//86/fvP/SErS//
AOz74MAAS3El9BMUl510xBgD/40AH/Hk/AWfmUP5VhClANN3nQMDBV8AAOt5j/0AKAIA0UEV
0/P0GAOv6g8RAMRE9CmLAI8EQtDhCcwkQuAzyQPvJ+/8FEJwPlf5wMxHWEzf9wgT/wjcQFRhQ
wQbV0xxyY/0DEMMihcEFJNwvNMEDEBY5RvUEEJVBVvmTBRY5Tf1RChBCpwDG9MHAwMBKAA
==
FP PIN=3 FID=0 Size=1592 Valid=1 TMP=TetTUzIxAAAEqKsECAUHCc7QAAAcqWkBAAAAh
FUooagrAKAPQADpAGKnzgAuAKkPtQA1qIsPnQBGAO0PTqhKAGIPlgCrADanNwByAElPngB6q
DQOfgCLAGENJKiNAD0PiABTAKimdACXAB0OVgCmqKwOigCnAFgOtqi0AKsPiAByAJimyQC3A
KsPjQDEqKwPngDBAFIPrqjDAKlPaQAPABGnrwDQAIwPAQDQqJsP6wDUAF4PXKjgAJ8PwgAhAI
Om2QDrAIwPTgDrqJIPQwD8AFYPdqj/AJQP9gDBAYmnugAGAXwPrwANqZMP6QAJAUYOWag
TAYsP+AD2AWqmVAA0AX8PUQA8qYAP6ABDAbIPKdPn+18ZTgTm3GePXwtHgjd7mih7jAplJY
ui+i6h1lRzlwfYrpLTDpojnwVODhLjqxleOdv2Afg5iMsIJlU6C2uTeQgwABW4PHhhcX2DhRNxqC
gS0PMpAGTwKaDYAkEWqQB88vGuWAIdE0oIJ5kGuRof7PcJDhDuZKcUGLX4xXs/aVLRqOyV9v3
26BRpuqzzQhBmApMXuSc/DZL1wezA/p1XgAbS8Jr0nANWq9IlCgnK9TsCQaxMC9kDZQxXAjeI
z/Xm8gv4DABRoH8C8fiq+fcX2VDY+5L5dfoD79dIz/XX7LfvZvOHr94H6QrL9xbvuTys+2sgRAHH
pSVRDQCpBCtrlmTEpgGQBxw4/55qYK0BcQgP/zHBAIWgEScTAF0Jxvz7V/5HwP5rZ5QMBDYlHv
5KwGKRBAQSCydZBQDDzifE7wUAtQ0nYNMAQLgBwP3+QV6hUl30CgAlJ+D8O/3EVf4wFwAqK
iL9+Vf/MThTZcAEwY+gAWwsj5LDQAYElSxndMINAG0tI/1j/8L/XQzFbDYhwJ9pwnsFxdI2j1MJA
HQzHoNU+64BRkprwsS2DwTmS/38//wwkMPGxgwAnEorwjv/YGjAwYQOAEmKZ8ckksLBwsDA
zwCxwSj/wVjCcdQAU8VWxMPCwcIHw/tqwMDC/UsMxTlqfif9/MD9/qoMBPRuT8LDxMBYwWiu
AZdwMFLAwwBl3UHDw8DDB8Uzcu6RwsIGADmzQ8UgCQBleCfDiWEfqA6Fyf////T++VX+/f7/
wP4F/8RowItvBgBaQDTGIAwAe4ipwjvlzW2hfhUAFo8DQfpX//39/S7/7nwIqByPQ8LCiAV+xKABl
489jMK2BAQdkDSDFQCIV6v7IMfHw8TCwgb/xtTAhQUAdZvnw/poCgAVoD2SuWQDqLe5IHSF
CsURv5jCacHB/sPGAKZrFsAGAEzE4cF+rwFsyRPAhjsGBKbLKcLAiAnFa8q2wv/Cw1MGxcjSv8HC
Rg0AWyEXftZHSQQA3O3KWQ2ojvADaEf/xgBDVRLABRAnAMxHArh3AAl+wATVvwKoQAUQKQ
gTAPzGrhFuCQZpwMgQ/7qHdsDCQ/4FwRW4LhYJwf/COsJUaMNDQBAQOuYJdFfAZmb+YAjV
/C7YM2QJEQMtk/77Vf//UgUQWfH9UKcR9TdwLcCeUm2vEZA7fVJdyRDg7nvA/2jC/gY/Drg7XA
PAbcI7/cb6QwALQwAADKcPCQ==

A response from the server:
HTTP/1.1 200 OK
Server: nginx/1.6.0
Date: Thu, 30 Jul 2015 07: 25: 38 GMT
Content-Type: text/plain
Content-Length: 4
Connection: close
Pragma: no-cache
Cache-Control: no-store

OK: 3

7.7 Uploading Face Template

The configuration ServerVer parameter sent by the server for initialization information exchange is larger than or equal to version 2.2.14.

A request message from the client:

```
POST /iclock/cdata?SN=${SerialNumber}&table=OPERLOG&Stamp=${XXX} HTTP/1.1
Host: ${ServerIP}: ${ServerPort}
Content-Length: ${XXX}
……

${DataRecord}
```

Annotation:

```
HTTP request method: POST method
URI: /iclock/cdata
HTTP protocol version: 1.1
Client configuration information:
SN: ${Required} Serial number of the client
table=OPERLOG: ${Required}
Stamp: ${Optional} Latest timestamp at which the face template is uploaded to the server. (For details, see the "OPERLOGStamp" parameter in "Initializing Information Exchange".)
Host header field: ${Required}
Content-Length header field: ${Required}
Other header fields: ${Optional}
Request entity: ${DataRecord}, face template data, in the following format:
FACE${SP}PIN=${XXX}${HT}FID=${XXX}${HT}SIZE=${XXX}${HT}VALID=${XXX}${HT}TMP=${XXX}
Note:
SIZE=${XXX}: Length after base64 coding of the face template
TMP=${XXX}: When the face template is transmitted, sixteen bytes (of random content) need to be added as the prefix of the original binary face template before base64 coding is conducted.
${LF} is used to connect multiple records.
```

A normal response message from the server:

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

OK: ${XXX}
```

Annotation:

```
HTTP status line: Defined with standard HTTP protocol
HTTP response header field:
Content-Length header field: Based to the HTTP 1.1, this header field is generally used to specify the data length of the response entity. If the response entity size is uncertain, head fields of Transfer-Encoding: chunked, Content-Length and Transfer-Encoding are supported, all of which are standard definitions of HTTP and are not described in details here.
```

Response entity: When the server normally receives data and successfully processes data, OK is sent: ${XXX} is replied. ${XXX} represents the number of records successfully processed. When an error occurs, the error description is replied.

Example:

A request from the client:
POST /iclock/cdata?SN=0316144680030&table=OPERLOG&Stamp=9999 HTTP/1.1
Host: 58.250.50.81: 8011
User-Agent: iClock Proxy/1.09
Connection: close
Accept: */*
Content-Length: 1684

FACE PIN=306 FID=2 SIZE=1648 VALID=1 TMP=AAAAAAAAAAAAAAAAAAAAAFpLRmlYAT
FLFLToAUQBQ1Mg+fgXuia23BDrNtwSfgJ8g74H3YHmXlkFpgetB5eH5yXuBvMLoa6wSx9HNgK
7RP80v1i+LLY8nCn7PXmD7w15Bp8N1wm/A78PowejZx9jJyWnBZ88K5wVfDDcNTjifGlvox9iD8
sf1g37B70Fk4WRl5RrKq8uD2MngRexMxk5cbDiH+c3xj+CV8Zf1idaDfWbkB8Rnwt/AV8Du0SvAd
dBywHMQ9MVysfFkNENfZD9FJ9jrnGeBD5Kcwp7CVySfJzOE+wZxjWFVY6fgreXHBd6B4ov4BXB
X+GuIZ4pazBTINiG8kf4h/DHxGaFxYe+yh+O1slCDsPcweuB/SHnA+UnqwG3AvnA88DZg/vhma
aV4dsWzwerBn1jLcN8wu/ErlTiR+YHVsc1wy2wdaC5uEmxKbwZ+AGeB5fFt6WLVa8kq/gvqqv8L
vwsBAACAgIHAQABAAEGAUxyAQkcIP9SAAohGhchAQAAEAEAAQYFASBPAgYGB1YKAQEMBg
AACg8HFQ0DDAoEAg8CCBQDAxtLDwM40CUFEBNVSQYDDRNJJg8CAAcJFBMMAQQiYA8MA
whZHAcEehMCACYEAAwACQsJBQAFBAYxAAknpwQGJ6EiBAUPIxwFBwQPOgQCBAARGz8WC
AIAGUQWBggLhyMIBJQlBAIiAgUEAgMPAwUACQAFUAABE/8EBwkQEgABCRECBwQFFEYNCAc
CFiJ3YwUACTpKLgwBDn8yDApjFAIBEwAAAwACBgEAAAECAUMBAwBCAAACAQIBAAEABAMB
AQUEAQMCBR0x/z4CAhUhSk8GAAcIDhUBNgsLDQMBAQAAAAYDAAAAAAEEAQAHBwEGQWE
IAAJe/zQBAAYY/30GAAABAAICAgAABAMAAAEFFAgAAAoEAAEUAgABAQEABwEECQMWGAIB
DSIHASH/HQQIClQeBgYMCgsIAwEBDSB0IBEJAjLbQgkIB245BQYKCAAEAAEDAwEDDgcKAQIBA
hwABB+4DAQOoiAFCAgXFwMBByNDCgYBABUQGSYDAwItYhMIATG9KQwLV0oCBBkEAQgBD
ggHAQADBwJDAxAS6AIGFxYXBgMiRxQNCgMXvCMOBQIXGrxABgkGJQ4YBQEQHxUJBUgNCg
ggAwIBAgMKAAAAAgMCJwADAzUAAAEAAwAAAB4BAAEAAAAAAAAAA1j/JgQGAxf/fAMBAgk
LAAEEAgIAAAAEAAEAAwEAAAABAAQAAAAAAAABAAAAARgQAQABAjP/KQMAAAIDAQEBBAI
EBAoCAAoUAgMBRAYCACoBAQAAAAsDAgAAAgEJAAMDHgABEiwNAQQnRA8DAwc5lDlCAQ
AIETAbAAEGM00eAQApYCIKA/81CwU5DAIGAAEEBQQABQACKAAGG2MEAxkpFQMBCTkKAAU
CFSYJAQABDByEGwIBAjUrlQUDO/8eCwSMRAQACAQABQEABgAAAAMAAA8AAAMuAAAAAw
gCBQMFCQcCBBNECgMAAAQvZhABAAhZ/DIBACL/XQgDJiEHAgwFAQEAAAcDAAAAAQEDAA
ACAAAAAwEAAgEFCQEBBAEHIAIAAQELFRQEAwICS/8rBQEDRVQUBAYODwQAAAgAAAANAA
AAAAAA
A response from the server:
HTTP/1.1 200 OK
Server: nginx/1.6.0
Date: Thu, 30 Jul 2015 07: 25: 38 GMT
Content-Type: text/plain
Content-Length: 4
Connection: close
Pragma: no-cache
Cache-Control: no-store

OK: 1

7.8 Uploading User Photo

The configuration ServerVer parameter sent by the server for initialization information exchange is greater than or equal to version 2.2.14.

A request message from the client:

```
POST /iclock/cdata?SN=${SerialNumber}&table=OPERLOG&Stamp=${XXX} HTTP/1.1
Host: ${ServerIP}: ${ServerPort}
Content-Length: ${XXX}
……

${DataRecord}
```

Annotation:

```
HTTP request method: POST method
URI: /iclock/cdata
HTTP protocol version: 1.1
Client configuration information:
SN: ${Required} Serial number of the client
table=OPERLOG: ${Required}
Stamp: ${Optional} Latest timestamp at which the user photo is uploaded to the server. (For de
tails, see the "OPERLOGStamp" parameter in "Initializing Information Exchange".)
Host header field: ${Required}
Content-Length header field: ${Required}
Other header fields: ${Optional}
Request entity: ${DataRecord}, fingerprint template data, in the following format:
USERPIC${SP}PIN=${XXX}${HT}FileName=${XXX}${HT}Size=${XXX}${HT}Content=${XXX}
Note:
FileName=${XXX}: Filename of the user photo, with only the jpg format supported currently.
Content=${XXX}: When the user photo is transmitted, base64 coding needs to be conducte
d for the original binary user photo.
Size=${XXX}: Length of the user photo after base64 coding.
${LF} is used to connect multiple records.
```

A normal response message from the server:

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

OK: ${XXX}
```

Annotation:

```
HTTP status line: Defined with standard HTTP protocol
HTTP response header field:
Content-Length header field: Based on the HTTP 1.1, this header field is usually used to specify
the data length of the response entity. If the response entity size is uncertain, head fields of Tra
nsfer-Encoding: chunked, Content-Length and Transfer-Encoding are supported, all of which a
re standard definitions of HTTP and are not described in details here.
```

Response entity: When the server normally receives data and successfully processes data, OK is sent: ${XXX} is replied. ${XXX} represents the number of records successfully processed. When an error occurs, the error description is replied.

Example:

```
POST /iclock/cdata?SN=0316144680030&table=OPERLOG&Stamp=9999 HTTP/1.1
Host: 58.250.50.81: 8011
User-Agent: iClock Proxy/1.09
Connection: close
Accept: */*
Content-Length: 1684

USERPIC PIN=123 FileName=123.jpg    Size=10 Content=AAAAAAAAAA……


A response from the server:
HTTP/1.1 200 OK
Server: nginx/1.6.0
Date: Thu, 30 Jul 2015 07: 25: 38 GMT
Content-Type: text/plain
Content-Length: 4
Connection: close
Pragma: no-cache
Cache-Control: no-store

OK: 1
```

8. Get Command

If the server needs to operate the equipment, the server generates a command format, waits till the equipment initiates a request, and then sends a command to the equipment. For the result of command execution, see Reply Command.

A request message from the client:

```
Get /iclock/getrequest?SN=${SerialNumber}
Host: ${ServerIP}: ${ServerPort}
……
```

Annotation:

```
HTTP request method: GET method
URI: /iclock/getrequest
HTTP protocol version: 1.1
Client configuration information:
SN: ${Required} Serial number of the client
Host head field: ${Required}
Other header fields: ${Optional}
```

A normal response message from the server:

```
When no commands are sent, the reply is as follows:
HTTP/1.1 200 OK
Date: ${XXX}
Content-Length: 2
……

OK

When a command is sent, the reply is as follows:
HTTP/1.1 200 OK Date: ${XXX} Content-Length: ${XXX} ……
```

Annotation:

```
HTTP status line: Defined with standard HTTP protocol
HTTP response header field:
Date header field: ${Required} This header field is used for synchronization with the server tim
e, in GMT format. For example, Date: Fri, 03 Jul 2015 06: 53: 01 GMT
Content-Length header field: Based on HTTP 1.1, this header field is usually used to specify the
data length of the response entity. If the response entity size is uncertain, head fields of Transfe
r-Encoding: chunked, Content-Length and Transfer-Encoding are supported, all of which are st
andard definitions of HTTP and are not described in details here.
Response entity: ${CmdRecord}, issued command record, in the following data format:
C: ${CmdID}: ${CmdDesc}${SP}${XXX}
Note：
${CmdID}: This command ID is generated by the server randomly, supporting numbers
and letters and with a length not over 16 digits. The client needs to reply to the comm
and with this command ID. For details, see the "Reply Command" function as follows.
```

${CmdDesc}: Command description falls into data commands and control commands. The data command is unified as the "DATA" description and detailed in the following "Data Command" function, and all kinds of control commands are different descriptions.
${LF} is used to connect multiple records.

8.1 DATA Command

When ${CmdDesc} in a command issued by the server is "DATA", this command is deemed as a data command. The client data can be added, deleted, modified, or queried, but different service data supports different operations. For details, see the following.

8.1.1 UPDATE Subcommand

Adding or modifying data: Whether adding or modifying depends on whether corresponding data exists on the client, and this operation has nothing to do with the server. The following shows the command format:

C: ${CmdID}: DATA${SP}UPDATE${SP}${TableName}${SP}${DataRecord}
Note:
UPDATE: This description is used to represent the operation of adding or modifying data.
${TableName}: Different names of service data tables, for example, the user information USERINFO. The following describes specific supported data.
${DataRecord}: Service data records in the form of key=value. Different service data has different key descriptions. The following describes the specifics.

8.1.1.1 User Information

The command format is:

C: ${CmdID}: DATA${SP}UPDATE${SP}USERINFO${SP}PIN=${XXX}${HT}Name=${XXX}${HT}Passwd=${XXX}${HT}Card=${XXX}${HT}Grp=${XXX}${HT}TZ=${XXX}${HT}Pri=${XXX}
Note:
PIN=${XXX}: User ID
Name=${XXX}: User name. When the equipment is in Chinese, the GB2312 code is used. When the equipment is in another language, the UTF-8 code is used.
Passwd=${XXX}: Password
Card=${XXX}: Card number, supporting two formats.
a. hexadecimal data, in the format of [%02x%02x%02x%02x], representing the first, second, third or fourth digit from left to right. For example, if the card number is 123456789, this is: Card=[15CD5B07]
b. string data. If the card number is 123456789, this is: Card=123456789
Grp=${XXX}: Group to which the user belongs, group 1 by default.
TZ=${XXX}: Information on number of the time period used by the user, in the format of XXXXXXXXXXXXXXXX. Digit 1-4 describe whether the group time period is used, digit 5-8 describe using personal time period 1, digit 9-12 describe using personal time period 2, and digit 13-16 d

escribe using personal time period 3.
For example: 0000000000000000 represents use of the group time period.
0001000200000000 represents use of personal time period, with personal time period 1 using t he time information of number 2 time period.
0001000200010000 represents using personal time period, with personal time period 1 using t he time information of number 2 time period and personal time period 2 using the time infor mation of number 1 time period.
Pri=${XXX}: User privilege value, with the meaning described as below

0	Normal user
2	Registrar
6	Administrator
10	User-defined
14	Super administrator

${LF} is used to connect multiple records.

For how the result of command execution is replied, see the Reply Command function. For the Return value, see Appendix 1. The format of returned content is:

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

8.1.1.2 Fingerprint Template

The command format is:

```
C: ${CmdID}: DATA${SP}UPDATE${SP}FINGERTMP${SP}PIN=${XXX}${HT}FID=${XXX}${HT}Size=
${XXX}${HT}Valid=${XXX}${HT}TMP=${XXX}
```

Note:
PIN=${XXX}: User ID
FID=${XXX}: Finger number, valued from 0 – 9.
Size=${XXX}: Length of binary data of the finger template after base64 coding
Valid=${XXX}: to describe the template validity and duress mark, with the following valu es and meanings:
Value and description

0	Invalid template
1	Normal template
3	Duress template

TMP=${XXX}: When the fingerprint template is transmitted, base64 coding needs to be conduc ted for the original binary fingerprint template.
${LF} is used to connect multiple records.

For how the result of command execution is replied, see the Reply Command function. For the Return value, see Appendix 1. The format of returned content is:

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

8.1.1.3 Face Template

The command format is:

```
C: ${CmdID}: DATA${SP}UPDATE${SP}FACE${SP}PIN=${XXX}${HT}FID=${XXX}${HT}Valid=${XXX}
${HT}Size=${XXX}${HT}TMP=${XXX}
```

Note:
PIN=${XXX}: User ID
FID=${XXX}: Face template number, valued from 0.
Size=${XXX}: Length of binary data of the face template after base64 coding
Valid=${XXX}: Face template validity mark, with the following values and meanings: Value and description

0	Invalid template
1	Normal template

TMP=${XXX}: When the face template is transmitted, base64 coding needs to be conducted for the original binary face template.
${LF} is used to connect multiple records.

For how the result of command execution is replied, see the Reply Command function. For the Return value, see Appendix 1. The format of returned content is:

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

8.1.1.4 User Photo

The command format is:

```
C: ${CmdID}: DATA${SP}UPDATE${SP}USERPIC${SP}PIN=${XXX}${HT}Size=${XXX}${HT}Content=${XXX}
```

Note:
PIN=${XXX}: User ID
Size=${XXX}: Length of binary data of the user photo after base64 coding
Content=${XXX}: When the user photo is transmitted, base64 coding needs to be conducted for the original binary user photo.
${LF} is used to connect multiple records.

For how the result of command execution is replied, see the Reply Command function. For the Return value, see Appendix 1. The format of returned content is:

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

8.1.1.5 Short Message

The command format is:

```
C: ${CmdID}: DATA${SP}UPDATE${SP}SMS${SP}MSG=${XXX}${HT}TAG=${XXX}${HT}UID=${XXX}${HT}MIN=${XXX}${HT}StartTime=${XXX}
```

Note:
MSG=${XXX}: Content of the short message, supporting up to 320 bytes. When the equipment is in Chinese, the GB2312 code is used. When the equipment is in another language, the UTF-8 code is used.
TAG=${XXX}: Type of the short message, with the following values and meanings:
Value and description

253	Public short message
254	User short message
255	Reserved short message

UID=${XXX}: Number of the short message, supporting only integer.
MIN=${XXX}: Valid duration of the short message, in minute.
StartTime=${XXX}: Starting time for the short message to take effect, in the format of XXXX-XX-

XX XX: XX: XX. For example, 2015-07-29 00: 00: 00
${LF} is used to connect multiple records.

For how the result of command execution is replied, see the Reply Command function. For the Return value, see Appendix 1. The format of returned content is:

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

8.1.1.6 Personal Short Message User List

The command format is:

C: ${CmdID}: DATA${SP}UPDATE${SP}USER_SMS${SP}PIN=${XXX}${HT}UID=${XXX}
Note:
PIN=${XXX}: User ID
UID=${XXX}: Number of the short message, supporting only integer.
${LF} is used to connect multiple records.

For how the result of command execution is replied, see the Reply Command function. For the Return value, see Appendix 1. The format of returned content is:

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

8.1.2DELETE Subcommand

To delete data. The command format is:

C: ${CmdID}: DATA${SP}DELETE${SP}${TableName}${SP}${DataRecord}
Note:
DELETE: This description is used to represent the operation of deleting data.
${TableName}: Different service data table names. For example, the user information is USERINFO, and the following describes specific supported data.
${DataRecord}: Condition for deleting data. Different service data supports different conditions. The following describes the specifics.

8.1.2.1 User Information

The command format is:

C: ${CmdID}: DATA${SP}DELETE${SP}USERINFO${SP}PIN=${XXX}
Note:
PIN=${XXX}: User ID
To delete specified user information, including fingerprint template, face template and user photo.

For how the result of command execution is replied, see the Reply Command function. For the Return value, see Appendix 1. The format of returned content is:

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

8.1.2.2 Fingerprint Template

The command format is:

C: ${CmdID}: DATA${SP}DELETE${SP}FINGERTMP${SP}PIN=${XXX}${HT}FID=${XXX}
Note:
PIN=${XXX}: User ID
FID=${XXX}: Finger number, valued from 0-9.
To delete specified fingerprint template. When only PIN information is transmitted, all fingerprints of the user are deleted.

For how the result of command execution is replied, see the Reply Command function. For the Return value, see Appendix 1. The format of returned content is:

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

8.1.2.3 Face Template

The command format is:

C: ${CmdID}: DATA${SP}DELETE${SP}FACE${SP}PIN=${XXX}
Note:
PIN=${XXX}:User ID
To delete specified face template of the user

For how the result of command execution is replied, see the Reply Command function. For the Return value, see Appendix 1. The format of returned content is:

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

8.1.2.4 User Photo

The command format is:

C: ${CmdID}: DATA${SP}DELETE${SP}USERPIC${SP}PIN=${XXX}
Note:
PIN=${XXX}: User ID
To delete specified user photo of the user

For how the result of command execution is replied, see the Reply Command function. For the Return value, see Appendix 1. The format of returned content is:

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

8.1.2.5 Short Message

The command format is:

```
C:${CmdID}:DATA${SP}DELETE${SP}SMS${SP}UID=${XXX}
Note:
UID=${XXX}: short message number, supporting only integers.
${LF} is used to connect multiple records.
```

For how the result of command execution is replied, see the Reply Command function. For the Return value, see Appendix 1. The format of returned content is:

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

8.1.3 QUERY Subcommand

To query data, the command format is:

```
C: ${CmdID}: DATA${SP}QUERY${SP}${TableName}${SP}${DataRecord}
Note:
QUERY: This description is used to represent the operation of querying data.
${TableName}: Different service data table names. For example, the user information is USERIN
FO, and the following describes specific supported data.
${DataRecord}: Condition for querying data. Different service data supports different condition
s. The following describes the specifics.
```

8.1.3.1 Attendance Record

The command format is:

```
C: ${CmdID}: DATA${SP}QUERY${SP}ATTLOG${SP}StartTime=${XXX}${HT}EndTime=${XXX}
Note:
StartTime=${XXX}: Query starting time, in the format of XXXX-XX-XX XX: XX: XX. For example, 2
015-07-29 00: 00: 00
EndTime=${XXX}: Query ending time, in the format of XXXX-XX-XX XX: XX: XX. For example, 20
15-07-29 23: 59: 59
```

For how the result of command execution is replied, see the Reply Command function. For the Return value, see Appendix 1. The format of returned content is:

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

To query the attendance record within specified time period. For how to upload, see "Uploading Attendance Record".

8.1.3.2 Attendance Photo

The command format is:

```
C: ${CmdID}: DATA${SP}QUERY${SP}ATTPHOTO${SP}StartTime=${XXX}${HT}EndTime=${XXX}
Note:
StartTime=${XXX}: Query starting time, in the format of XXXX-XX-XX XX: XX: XX. For example, 2
015-07-29 00: 00: 00
EndTime=${XXX}: Query ending time, in the format of XXXX-XX-XX XX: XX: XX. For example, 20
15-07-29 23: 59: 59
```

For how the result of command execution is replied, see the Reply Command function. For the Return value, see Appendix 1. The format of returned content is:

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

To query the attendance photo within specified time period. For how to upload, see "Uploading Attendance Photo".

8.1.3.3 User Information

The command format is:

```
C: ${CmdID}: DATA${SP}QUERY${SP}USERINFO${SP}PIN=${XXX}
Note:
PIN=${XXX}: User ID
```

For how the result of command execution is replied, see the Reply Command function. For the Return value, see Appendix 1. The format of returned content is:

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

To query the basic information of specified user. For how to upload, see "Uploading User Information".

8.1.3.4 Fingerprint Template

The command format is:

```
C: ${CmdID}: DATA${SP}QUERY${SP}FINGERTMP${SP}PIN=${XXX}${HT}FingerID=${XXX}
Note:
PIN=${XXX}: User ID
FingerID=${XXX}: Finger number, valued from 0 – 9.
```

For how the result of command execution is replied, see the Reply Command function. For the Return value, see Appendix 1. The format of returned content is:

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

To query the fingerprint template information of the user. When only the PIN information is transmitted, the information about all fingerprint templates of the user is queried. For how to upload, see "Uploading Fingerprint Template".

8.2 CLEAR Command

8.2.1 Clearing Attendance Record

To clear the client attendance record, the command format is:

```
C: ${CmdID}: CLEAR${SP}LOG
Note:
CLEAR${SP}LOG is used to describe this command.
```

For how the result of command execution is replied, see the Reply Command function. For the Return value, see Appendix 1. The format of returned content is:

```
ID=${XXX}&Return=${XXX}&CMD=CLEAR_LOG
Note:
CMD=CLEAR_LOG: CLEAR_LOG is used to describe this command.
```

8.2.2 Clearing Attendance Photo

To clear the client attendance photo, the command format is:

```
C: ${CmdID}: CLEAR${SP}PHOTO
Note:
CLEAR${SP}PHOTO is used to describe this command.
```

For how the result of command execution is replied, see the Reply Command function. For the Return value, see Appendix 1. The format of returned content is:

```
ID=${XXX}&Return=${XXX}&CMD=CLEAR_PHOTO
Note:
CMD=CLEAR_PHOTO: CLEAR_PHOTO is used to describe this command.
```

8.2.3 Clearing All Data

To clear all client data, the command format is:

```
C: ${CmdID}: CLEAR${SP}DATA
Note:
CLEAR${SP}DATA is used to describe this command.
```

For how the result of command execution is replied, see the Reply Command function. For the Return value, see Appendix 1. The format of returned content is:

```
ID=${XXX}&Return=${XXX}&CMD=CLEAR_DATA
```

Note:
CMD=CLEAR_DATA: CLEAR_DATA is used to describe this command.

8.3 Check Command

8.3.1 Checking Data Update

The client is required to read configuration information from the server and re-upload corresponding data to the server based on the timestamp. For details, see "Initializing Information Exchange". Currently, only the server resetting the timestamp to 0 is supported. For example, set parameter Stamp to 0. After reading configuration parameters, the client conducts Uploading Attendance Record again, and the command format is:

C: ${CmdID}: CHECK
Note:
CHECK is used to describe this command.

For how the result of command execution is replied, see the Reply Command function. For the Return value, see Appendix 1. The format of returned content is:

ID=${XXX}&Return=${XXX}&CMD=CHECK

8.3.2 Checking and Transmitting New Data

The client immediately checks whether new data exists and transmits the new data to the server. The command format is:

C: ${CmdID}: LOG
Note: LOG is used to describe this command.

For how the result of command execution is replied, see the Reply Command function. For the Return value, see Appendix 1. The format of returned content is:

ID=${XXX}&Return=${XXX}&CMD=LOG

8.3.3 Automatically Verifying Attendance Data

The server issues the verification for attendance records within a time period, start and end time of uploading by the attendance equipment, as well as total number of records. The verification is achieved by the server, and the command format is:

C: ${CmdID}: VERIFY${SP}SUM${SP}ATTLOG${SP}StartTime=${XXX}${HT}EndTime=${XXX}
Note:
VERIFY${SP}SUM is used to describe this command
StartTime=${XXX}: Starting time of issuing by the server, in the format of XXXX-XX-XX XX:

XX: XX. For example, 2015-07-29 00: 00: 00
EndTime=${XXX}: Ending time of issuing by the server, in the format of XXXX-XX-XX XX: XX: XX. For example, 2015-07-29 00: 00: 00

For how the result of command execution is replied, see the Reply Command function. For the Return value, see Appendix 1. The format of returned content is:

ID=${XXX}&Return=${XXX}&CMD=VERIFY${SP}SUM&StartTime=${XXX}&EndTime=${XXX}&AttlogSum=${XXX}
Note:
AttlogSum=${XXX}: Total number of attendance records within the period from starting to ending time

8.4 Configuring Option Command

8.4.1 Option for Setting the Client

To set the client configuration information, the command format is:

C: ${CmdID}: SET${SP}OPTION${SP}${Key}=${Value}
Note:
SET${SP}OPTION is used to describe this command.
The configuration information is set in the form of key-value, and this command supports only the configuration of single configuration information.

For how the result of command execution is replied, see the Reply Command function. For the Return value, see Appendix 1. The format of returned content is:

ID=${XXX}&Return=${XXX}&CMD=SET${SP}OPTION

8.4.2 Option for Refreshing the Client

The client reloads the configuration information. The command format is:

C: ${CmdID}: RELOAD${SP}OPTIONS
Note:
RELOAD${SP}OPTIONS is used to describe this command.

For how the result of command execution is replied, see the Reply Command function. For the Return value, see Appendix 1. The format of returned content is:

ID=${XXX}&Return=${XXX}&CMD=RELOAD${SP}OPTIONS

8.4.3 Sending Client Information to the Server

The server gets information such as client configuration. The command format is:

C: ${CmdID}: INFO
Note:
INFO is used to describe this command.

For how the result of command execution is replied, see the Reply Command function. For the Return value, see Appendix 1. The format of returned content is:

ID=${XXX}&Return=${XXX}&CMD=INFO${LF}${Key}=${Value}${LF}${Key}=${Value}${LF}${Key}=${Value}${LF}${Key}=${Value}……
Note:
CMD=INFO is followed by specific customer configuration information, in the form of key-value.

8.5 File Command

8.5.1 Getting File in the Client

The client sends a server-specified file to the server. The command format is:

C: ${CmdID}: GetFile${SP}${FilePath}
Note:
GetFile is used to describe this command.
${FilePath}: File in the client system

For how the result of command execution is replied, see the Reply Command function. For the Return value, see Appendix 1. The format of returned content is:

ID=${XXX}${LF}SN=${SerialNumber}${LF}FILENAME=${XXX}${LF}CMD=GetFile${LF}Return=${XXX}${LF}Content=${BinaryData}
Note:
Return=${XXX}: Size of returned file.
Content=${BinaryData}: Binary data flow of the transmitted file

8.5.2 Sending File to the Client

The equipment is required to download a file from the server and saves the file in a specified folder. (After being downloaded, a .tgz file is automatically decompressed to the specified directory of FilePath or /mnt/mtdblock if no directory is specified. For a file in another format, the file save path and filename need to be specified.) This file must be provided by the server by HTTP, as well as the URL for obtaining this file. If the URL starts with "http://", the equipment deems the URL as a complete URL address, otherwise, the equipment appends the server's /iclock/ address to specified URL. The command format is:

C: ${CmdID}: PutFile${SP}${URL}${HT}${FilePath}
Note:
GetFile is used to describe this command.

${URL}: Address of the file to be downloaded from the server
${FilePath}: Destination path for the file to be saved on the client
Example 1: PutFile file/fw/X938/main.tgz main.tgz or PutFile file/fw/X938/main.tgz requires t he equipment to download http: //server/iclock/file/fw/X938/main.tgz, and decompress main. tgz into the folder of /mnt/mtdblock.
Example 2: PutFile file/fw/X938/main.tgz /mnt/ requires the equipment to download http: //s erver/iclock/file/fw/X938/main.tgz, and decompress main.tgz into the folder of /mnt/.
Example 3: PutFile file/fw/X938/ssruser.dat /mnt/mtdblock/ssruser.dat requires the equipment to download http: //server/iclock/file/fw/X938/ssruser.dat, and remain the file to be /mnt/mtd block/ssruser.dat.

For how the result of command execution is replied, see the Reply Command function. For the Return value, see Appendix 1. The format of returned content is:

ID=${XXX}${LF}Return=${XXX}${LF}CMD=PutFile
Note:
Return=${XXX}: Size of the returned file

8.6 Remote Enrollment Command

8.6.1 Enrolling User Fingerprint

The fingerprint enrollment is initiated by the server and conducted on the client. The command format is:

C: ${CmdID}: ENROLL_FP${SP}PIN=${XXX}${HT}FID=${XXX}${HT}RETRY=${XXX}${HT}OVERWRITE =${XXX}
Note:
ENROLL_FP is used to describe this command.
PIN=${XXX}: Enrolled user ID
FID=${XXX}: Enrolled fingerprint number
RETRY=${XXX}: Number of retries required if enrollment fails
OVERWRITE=${XXX}: Whether to overwrite the fingerprint. 0 means the fingerprint of cor responding user exists and will not be overwritten and error information is returned. 1 means the fingerprint of corresponding user exists and will be overwritten.

For how the result of command execution is replied, see the Reply Command function. For the Return value, see Appendix 1. The format of returned content is:

ID=${XXX}&Return=${XXX}&CMD=ENROLL_FP

8.7 Control Command

8.7.1 Rebooting the Client

To reboot the client, the command format is:

```
C: ${CmdID}: REBOOT
Note: REBOOT is used to describe this command.
```

For how the result of command execution is replied, see the Reply Command function. For the Return value, see Appendix 1. The format of returned content is:

```
ID=${XXX}&Return=${XXX}&CMD=REBOOT
```

8.7.2 Outputting the Door Unlocking Signal

The access equipment outputs the door unlocking signal. The command format is:

```
C: ${CmdID}: AC_UNLOCK
Note: AC_UNLOCK is used to describe this command.
```

For how the result of command execution is replied, see the Reply Command function. For the Return value, see Appendix 1. The format of returned content is:

```
ID=${XXX}&Return=${XXX}&CMD=AC_UNLOCK
```

8.7.3 Canceling the Alarm Signal Output

The access equipment cancels the alarm signal output. The command format is:

```
C: ${CmdID}: AC_UNALARM
Note: AC_UNALARM is used to describe this command.
```

For how the result of command execution is replied, see the Reply Command function. For the Return value, see Appendix 1. The format of returned content is:

```
ID=${XXX}&Return=${XXX}&CMD=AC_UNALARM
```

8.8 Other Commands

8.8.1 Executing the System Command

The server issues operating system commands which are supported by the client which send execution results to the server. The command format is:

```
C: ${CmdID}: SHELL${SP}${SystemCmd}
Note:
SHELL is used to describe this command
${SystemCmd}: Operating system command. For example, when the client is linux system, 1s is
 supported.
```

For how the result of command execution is replied, see the Reply Command function. For the Return value, see Appendix 1. The format of returned content is:

ID=${XXX}${LF}SN=${SerialNumber}${LF}Return=${XXX}${LF}CMD=Shell${LF}FILENAME=shello
ut.txt${LF}Content=${XXX}
Note:
Return=${XXX}: The value is the returned value for the system command.
Content=${XXX}: The value is the output content of the system command.

9. Command Reply

After Getting Command Issued by the Server, the client needs to reply corresponding command.

A request message from the client:

```
POST /iclock/devicecmd?SN=${SerialNumber}
Host: ${ServerIP}: ${ServerPort}
Content-Length: ${XXX}
……
 ${CmdRecord}
```

Annotation:

HTTP request method: GET method
URI: /iclock/devicecmd
HTTP protocol version: 1.1
Client configuration information:
SN: ${Required} Serial number of the client
Host head field: ${Required}
Content-Length header field: ${Required}
Other header fields: ${Optional}
Response entity: ${CmdRecord}, record of replied commands. The reply content all contains th
e ID\Return\CMD information, with the following meanings:
ID: Number of the command issued by the client
Return: Returned result after the client executes the command
CMD: Description of the command issued by the server
A small number of replies contain other information. For specific reply content format, see the description of each command.
${LF} is used to connect multiple command reply records.

A normal response message from the server:

```
HTTP/1.1 200 OK
Date: ${XXX}
Content-Length: 2
……

OK
```

Annotation:

HTTP status line: Defined with standard HTTP protocol
HTTP response header field:
Date header field: ${Required} This header field is used for synchronization with the server tim e, in GMT format. For example, Date: Fri, 03 Jul 2015 06: 53: 01 GMT
Content-Length header field: Based on HTTP 1.1, this header field is usually used to specify the data length of the response entity. If the response entity size is uncertain, head fields of Transfe r-Encoding: chunked, Content-Length and Transfer-Encoding are supported, all of which are st andard definitions of HTTP and are not described in details here.

Example:

A request from the client:
POST /iclock/devicecmd?SN=0316144680030 HTTP/1.1
Host: 58.250.50.81: 8011
User-Agent: iClock Proxy/1.09
Connection: close
Accept: */*
Content-Length: 143

ID=info8487&Return=0&CMD=DATA
ID=info8488&Return=0&CMD=DATA
ID=info8489&Return=0&CMD=DATA
ID=info7464&Return=0&CMD=DATA
ID=fp7464&Return=0&CMD=DATA

A response from the server:
HTTP/1.1 200 OK
Server: nginx/1.6.0
Date: Tue, 30 Jun 2015 01: 24: 48 GMT
Content-Type: text/plain
Content-Length: 2
Connection: close
Pragma: no-cache
Cache-Control: no-store

OK

10. Remote Attendance

When attendance is required for a user on a business trip and no information about this user is stored in the attendance machine, the user can check on attendance remotely. Current application scenario: The user uses the attendance machine keypad to directly enter ID and press OK, and then the attendance machine requests the server to issue all information about this user (basic information and fingerprint information). After that, the user checks on attendance. After being downloaded, the user information is stored in the attendance machine for a period of time. The saving time is set via a parameter. After this period of time, the user information will be deleted.

A request message from the client:

```
GET /iclock/cdata?SN=${SerialNumber}&table=RemoteAtt&PIN=${XXX}
Host: ${ServerIP}: ${ServerPort}
……
```

Annotation:

```
HTTP request method: GET method
URI: /iclock/cdata
HTTP protocol version: 1.1
Client configuration information:
   SN: ${Required} Serial number of the client
   table=RemoteAtt: Acquiring user information for remote attendance
   PIN=${XXX}: ID information to be required
   Host head field: ${Required}
   Other header fields: ${Optional}
```

A normal response message from the server:

```
When user information exists, the reply information is:
HTTP/1.1 200 OK
Date: ${XXX}
Content-Length: ${XXX}
……

DATA${SP}UPDATE${SP}USERINFO${SP}PIN=${XXX}${HT}Name=${XXX}${HT}Passwd=${XXX}${H
T}Card=${XXX}${HT}Grp=${XXX}${HT}TZ=${XXX}${HT}Pri=${XXX}
DATA${SP}UPDATE${SP}FINGERTMP${SP}PIN=${XXX}${HT}FID=${XXX}${HT}Size=${XXX}${HT}Val
id=${XXX}${HT}TMP=${XXX}
```

Annotation: ${LF} is used to connect multiple data records of the response entity. For specific data format, see Issuing User Information and Issuing Fingerprint Template.

11. Appendix 1

Error Code	Description
0	Successful
2	Enrolling User Print, the fingerprint of corresponding user already exists.
4	Enrolling User Print, enrollment failure, which is usually due to poor quality or not the same fingerprint enrolled for three times.
5	Enrolling User Print, the enrolled fingerprint already exists in the database
6	Enrolling User Print, enrollment is cancelled.
7	Enrolling User Print, the equipment is busy and enrollment cannot be conducted.
-1	The parameter is incorrect.
-2	The transmitted user photo data does not match the given size.
-3	Reading or writing is incorrect.
-9	The transmitted template data does not match the given size.
-10	The user specified by PIN does not exist in the equipment.
-11	The fingerprint template format is illegal.
-12	The fingerprint template is illegal.
-1001	Limited capacity
-1002	Not supported by the equipment
-1003	Command execution timeout
-1004	The data and equipment configuration are inconsistent.
-1005	The equipment is busy.
-1006	The data is too long.
-1007	Memory error

12. Appendix 2

Language number	Meaning
83	Simplified Chinese
69	English
97	Spanish
70	French
66	Arabic
80	Portuguese
82	Russian
71	German
65	Persian
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

13. Appendix 3

Operation code	Meaning
0	Startup
1	Shutdown
2	Authentication fails
3	Alarm
4	Access menu
5	Change settings
6	Enroll fingerprint
7	Enroll password
8	Enroll HID card
9	Delete user
10	Delete fingerprint
11	Delete password
12	Delete RF card
13	Clear data
14	Create MF card
15	Enroll MF card
16	Register MF card
17	Delete MF card registration
18	Clear MF card content
19	Move enrolled data into the card
20	Copy data in the card to the machine
21	Set time
22	Delivery configuration
23	Delete entry and exit records
24	Clear administrator privilege

25	Modify access group settings
26	Modify user access settings
27	Modify access time period
28	Modify unlocking combination settings
29	Unlock
30	Enroll a new user
31	Change fingerprint attribute
32	Duress alarm

14. Appendix 4

Operation code	Operation object 1	Operation object 2	Operation object 3	Operation object 4
2	If 1:1 authentication is used, this is user ID.			
3	Alarm	For alarm causes, see Appendix 5.		
5	Sequence number of modified setting item	Value after modification		
6	User ID	Sequence number of the fingerprint	Length of the fingerprint template	
9	User ID			
10	User ID			
11	User ID			
12	User ID			

15. Appendix 5

Alarm reason	Meaning
50	Door Close Detected
51	Door Open Detected
53	Out Door Button
54	Door Broken Accidentally
55	Machine Been Broken
58	Try Invalid Verification
65535	Alarm Cancelled

Attendance Push Protocol Document

company: ZKTeco Inc

web: www.zkteco.com

author: xsen

mail: xsen@zkteco.com

date: 2015-10-09