安防 PUSH 通讯协议

PUSH SDK

文档版本：V1.1　日期：2018 年 10 月

push 协议版本：V3.1.1

ZKTeco

修改记录

日期	历史版本	描述	修改人	备注
2018/10/10	V1.1	1.通信加密新增2条协议： ①交换公钥协议 ②交换因子协议 2.支持通信加密版本说明： ①门禁PUSH：3.1.1及以上 3.通信加密详细说明（见附录16）	阎广田	
2018/8/29	V1.0	1.添加可见光人脸相关协议	阎广田	
2018/4/16	初版		李仙平	

目录

1. 摘要

Push协议是基于超文本传输协议（HTTP）的基础上定义的数据协议，建立在TCP/IP连接上，主要应用于中控考勤、门禁等设备与服务器的数据交互， 定义了数据（用户信息、生物识别模板、门禁记录等）的传输格式、控制设备的命令格式；目前中控支持的服务器有WDMS、ZKECO、ZKNET、ZKBioSecurity3.0、等，第三方支持的服务器有印度ESSL等

2. 特点

- 新数据主动上传
- 所有行为都由客户端发起，比如【上传数据】、【服务器下发的命令】等

3. 编码

协议中传输的数据大部分都是ASCII字符，但是个别的字段也涉及到编码的问题，比如用户姓名， 所以对该类型数据做如下规定：

- 为中文时，使用GB2312编码
- 为其他语言时，使用UTF-8编码

目前涉及到该编码的数据如下：

- 用户信息表的用户姓名

4. HTTP协议简介

Push协议是基于HTTP协议的基础上定义的数据协议，这里简单介绍下什么是HTTP协议，如果已经熟悉可跳过此部分。

HTTP协议是一种请求/响应型的协议。客户端给服务器发送请求的格式是一个请求方法（request method），URI，协议版本号，然后紧接着一个包含请求修饰符（modifiers），客户端信息，和可能的消息主体的类MIME（MIME-like）消息。服务器对请求端发送响应的格式是以一个状态行（status line），其后跟随一个包含服务器信息、实体元信息和可能的实体主体内容的类MIME（MIME-like）的消息。其中状态行（status line）包含消息的协议版本号和一个成功或错误码。如下例子

```
客户端请求：
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
服务器响应：
```

```
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
```

HTTP通信通常发生在TCP/IP连接上。默认端口是TCP 80，不过其它端口也可以使用。但并不排除HTTP协议会在其它协议之上被实现。HTTP仅仅期望的是一个可靠的传输（译注：HTTP一般建立在传输层协议之上）；所以任何提供这种保证的协议都可以被使用。

5.定义

文档中引用定义使用格式为：${ServerIP}

- ServerIP：服务器IP地址
- ServerPort：服务器端口
- XXX：未知值
- Value1\Value2\Value3……\Valuen：值1\值2\值3……值n
- Required：必须存在
- Optional：可选
- SerialNumber：序列号(可以为字母、数字、字母+数字组合)
- NUL：null（\0）
- SP：空格

- LF：换行符（\n）
- CR：回车（\r）
- HT：制表符（\t）
- DataRecord：数据记录

- CmdRecord：命令记录
- CmdID：命令编号
- CmdDesc：命令描述
- Reserved：预留字段
- BinaryData：二进制数据流
- Base64Data：Base64数据流
- TableName：数据表名
- Key：键
- Value：值
- FilePath：文件路径
- URL：资源位置
- Cond： 条件

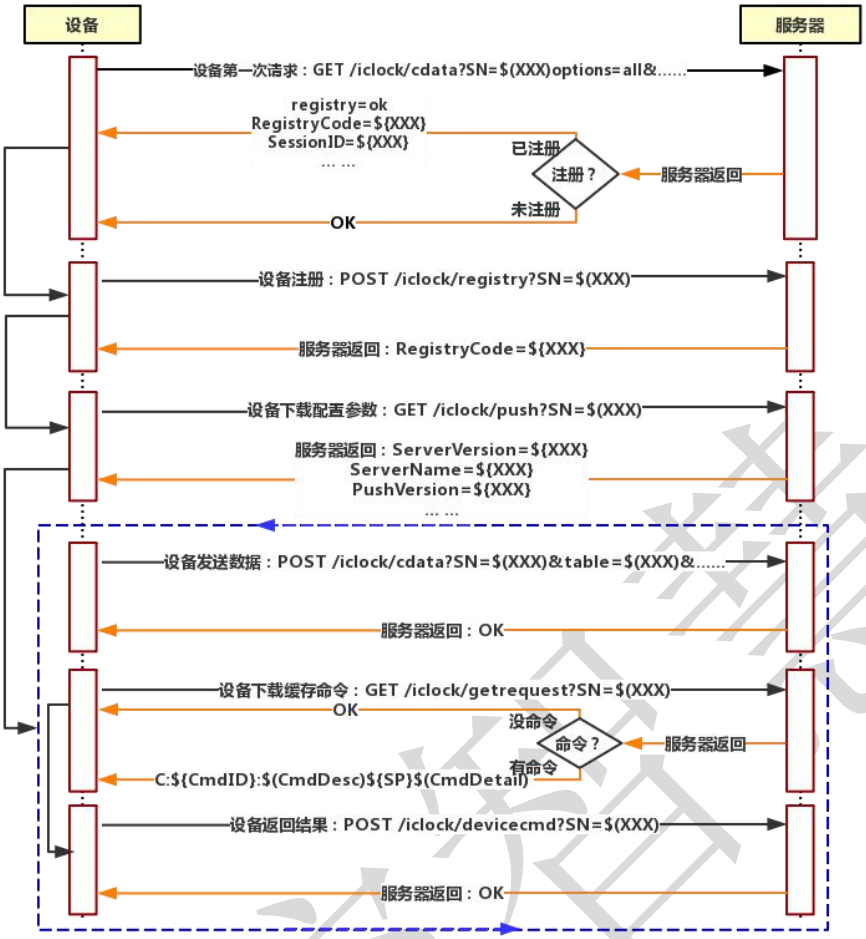
6. 功能

从客户端的角度来描述Push协议支持的功能

- 初始化信息交互
- 上传
- 下载
- 服务器命令详细说明
- 附录

7. 流程

使用Push协议的客户端和服务器，必须由客户端先发起“初始化信息交互”请求成功之后，才能使用其他功能，比如上传数据、获取服务器命令、上传更新信息、回复服务器命令等，其中这些功能并没有先后顺序，取决于客户端应用程序的开发，如下图



7.1 初始化信息交互

客户端发起请求，服务器接收到该请求后，根据是否已经注册有两种流程。如果是已经注册的设备发起请求，服务器返回注册码和服务器配置参数，完成连接过程；如果是还未注册的设备，则需要客户端再发起注册请求同时将设备参数发给服务器，注册成功后，服务器返回注册码，之后客户端再发起请求下载服务器配置参数，客户端获取到相应的配置信息后，才能算交互成功。具体如下

客户端请求连接

应用场景：

设备还未与软件建立连接时，需要发起连接请求以建立连接

客户端请求连接消息格式：(兼容老协议)：

```
GET /iclock/cdata?SN=$(SerialNumber)&pushver=${XXX}&options=all&${XXX}=${XXX} HTTP/1.1
```

```
Host: ${ServerIP}:${ServerPort}
```

```
……
```

注释：

```
HTTP请求方法使用：GET方法
URI使用：/iclock/cdata
HTTP协议版本使用：1.1
客户端配置信息：
SN：${Required}表示客户端的序列号 （后续相同内容不再说明）
pushver：${Optional}表示设备当前最新的push协议版本，详见【附录13】
options：${Required}表示获取服务器配置参数，目前值只有all
Host头域：${Required} （后续相同内容不再说明）
其他头域：${Optional}
如果设备已经注册，下一步需要发起下载配置参数请求；如果设备未注册，下一步需要发起注册请求，注册成功后再发起下载配置参数请求。（现阶段客户端每一次建立连接都需要进行注册）
${XXX}：当设备中设置该参数时才会推送，比如DeviceType。
```

设备未注册时服务器正常响应格式：

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

设备已注册时服务器正常响应格式：

```
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}

注释：

registry： OK表示设备已注册
RegistryCode:服务器生成随机数，最长32个字节。

ServerVersion： 服务器版本号

ServerName: 服务器名称

PushProtVer：服务端依据哪个协议版本开发的，详见【附录13】。

ErrorDelay: 联网失败后客户端重新联接服务器的间隔时间（秒），建议设置30~300秒，如果不配置客户端默认值为30秒

RequestDelay: 客户端获取命令请求间隔时间（秒），如果不配置客户端默认值为30秒

TransTimes: 定时检查传送新数据的时间点，如果不配置默认为"12:30;14:30”

TransInterval: 检查是否有新数据需要传送的间隔时间（分钟）,如果不配置客户端默认为2分钟

TransTables: 需要检查新数据并上传的数据，默认为"User Transaction",需要自动上传用户和门禁记录

Realtime: 客户端是否实时传送新记录。为1表示有新数据就传送到服务器,为0表示按照 TransTimes 和 TransInterval 规定的时间传送,如果不配置客户端默认值为1

SessionID: PUSH 通信会话 ID，今后设备的请求都需要此字段来计算新的 Token

TimeoutSec: 设置网络超时时间，如果不配置客户端默认 10秒

如果设备已经注册，软件会且必须返回 registry 、 RegistryCode ； 如果设备未注册，不返回 registry 、 RegistryCode 。

HTTP状态行：使用标准的HTTP协议定义

HTTP响应头域：

Date头域:${Required}使用该头域来同步服务器时间,并且时间格式使用GMT格式,如Date: Fri, 03 Jul 2015 06:53:01 GMT

Content-Length头域：根据HTTP 1.1协议，一般使用该头域指定响应实体的数据长度，如果是在不确定响应实体的大小时，也支持Transfer-Encoding: chunked，Content-Length及Transfer-Encoding头域均是HTTP协议的标准定义，这里就不再详述。

示例：

客户端请求连接：
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: zh-CN

设备未注册时服务器正常响应：

HTTP/1.1 200 OK

Server: Apache-Coyote/1.1

```
Content-Length: 2
Date: Mon, 09 Jan 2017 02:20:19 GMT
OK
```

7.2 交换公钥（支持通信加密的场合）

应用场景：

```
设备推送设备公钥，接收服务器返回的服务器公钥。
```

客户端请求消息：

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

注释：

```
HTTP请求方法使用：POST方法
URI使用：/iclock/exchange
HTTP协议版本使用：1.1
Host头域：${Required}
其他头域：${Optional}
PublicKey：调用加密库返回的设备公钥。
```

服务器响应：

```
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}
```

注释：

```
PublicKey：服务器返回的服务器公钥。
```

示例：

客户端请求消息：

```
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-Language: zh-CN
Content-Type: application/push;charset=UTF-8
Content-Language: zh-CN
Content-Length: 3580
PublicKey=DMCtR13RwiGI4M9TRn/3xEmkddz2lqoZR7zUrOMhOc3FLvhLtplWs3REOSaKT4A9WOOONt3V+mVbOW3Ka3NeCTWLjf9LplV
1EyJloZwXspGroPMTEWitLE+LLsrO1r47OQRr62j5YSViUDKgzLVCvEek2iJ+3D181Z3qxV7a7WloQ9DUGiPaU8gml4cmiyqQimxlQ1ww
McMpclFOlsSx7UjCG+D41dM/vh5UZxrQwn7liMOmNdFXlB+TOjaJ+4K/n3TDjubrbebqx6H2+nErH1mBuCCSNlKfwc5earkNfXPuqgBNG
qCFJojgcQiOySquaq2DFXdUwYBLlURDnBLf+TtoSh4=

服务器响应：
HTTP/1.1 200 OK
Server: Apache-Coyote/1.1
Content-Length: 590
Date: Mon, 10 Sep 2018 10:10:55 GMT

PublicKey=DMCvFlzRwiGI4M9TRn/3xEmkddz2lqoZR7zUrOMhOc3FLvhLtplYu3ZMNSeKVLcZUv4iHNnxzl9B8SfuVSxXAwyijYAj6Wg
4YyxTj4stv4K7q54sUCikb1CbQ/HOm9QZGyhM1WjrHhppXT/CsOAquEy/2gxfrSt3nai38Hb/8QoTHvnXJR2EVpcY6u47jBeGiXM3ZUQg
CtcdB7JBXsOr71XWEsLX1flC3GofGCyOgObUkumWJfNKwBWfWzb95o6klDi8uP/wU+DS1uLs1VcCNOWNtX+DCajyzcYvecR8cgbsOF1Qf
MmiRr/dYAOkwF/bSMyuLkd+o6FmLBAh9keFtgkFa+PC5RlFGrmxpJx4lMoLfaNqUNwyAuRdKezvYBDUrRhGwgtwo/BRGUoWCeOB4YP/gH
HGroOM8f3/HlSqliuT55Xks/Btp1tpfO/OeJjELUA9Yu0o4TQlnM19PuOGsYhipM9NeGGexKjtotHotLT4CcsoO4nAf7TltDavoPvVGJG
iDbnN7l8wsUCsqcCRsiKhpmON2waLjdFa8PNJ62N6Dl6QRPKn9XLnlDFdtKSq5Vgn
```

7.3 交换因子（支持通信加密的场合）

应用场景：

设备推送设备因子，接收服务器返回的服务器因子。

客户端请求消息：

```
POST /iclock/exchange?SN=$(SerialNumber)&type=factors
Host: ${ServerIP}:${ServerPort}
```

```
Content-Length: ${XXX}
……
Factors=${XXX}
```

注释：

```
HTTP请求方法使用：POST方法
URI使用：/iclock/exchange
HTTP协议版本使用：1.1
Host头域：${Required}
其他头域：${Optional}
Factors：调用加密库返回的设备因子。
```

服务器响应：

```
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}
```

注释：

```
Factors：服务器返回的服务器因子。
```

示例：

```
客户端请求消息：
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/VtnItwfRAbVfyAJku6jKdobqCUIz2eTJOyXIwI8ZMi5wSyIQePPhVDGQvrppcssZ/zgX6qAWSKUXVIGRXNQ6kBk7+Kc6SaI
090cvvMeLHCc1h69TIsbfkMtLGd1npscnDmmOoC19qqIbtTha9zNHmf38dHDkGZf3vLv/I8iwqV3RAX7MaIBW4M+kYvbdYNgzR5kqcG+w
TZOet5QAf/8YoRg7qZmnkAG6sAYALQotTfwQcjGUCtVoJONw8WshzkpdzgNsoo7UtYEmmhbEPHtqgaDN5OLexoE9u6Ip3gMd+V2hf70rs
+mVUYR6frkEKe/6rA+oIdguhb2Ip6HnwfNQ==

服务器响应：
HTTP/1.1 200 OK
Server: Apache-Coyote/1.1
Content-Length: 180
Date: Mon, 10 Sep 2018 10:10:55 GMT
Factors=XQ10e0WiFtsIJW5ob221T/WCK42GXGP6mmiBB9yB93rD3CxIKZo3mavyqfTKFxtCn8AtkxL7MH4UeRvRnFTrv3Q4kKaYndiip
huvxOGQxrzcGGjHOsRzgPcTtAQu0U7A8vg2sMzZOxokqLuDVE5nIsx1/1V46wTK+oNU9q8fgKM=
```

7.4 注册

应用场景：

客户端发起连接请求后，如果没有返回注册码（即未注册），需要发起注册请求以注册设备

客户端请求消息：

```
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),MaxPackage
Size=${XXX},LockCount=${XXX},ReaderCount=${XXX},AuxInCount=${XXX},AuxOutCount=${XXX},MachineType=${XXX},~
IsOnlyRFMachine=${XXX},~MaxUserCount=${XXX},~MaxAttLogCount=${XXX},~MaxUserFingerCount=${XXX},MThreshold=
${XXX},IPAddress=${XXX},NetMask=${XXX},GATEIPAddress=${XXX},~ZKFPVersion=${XXX},~REXInputFunOn=${XXX},~Ca
rdFormatFunOn=${XXX},~SupAuthrizeFunOn=${XXX},~ReaderCFGFunOn=${XXX},~ReaderLinkageFunOn=${XXX},~RelaySta
teFunOn=${XXX},~Ext485ReaderFunOn=${XXX},~TimeAPBFunOn=${XXX},~CtlAllRelayFunOn=${XXX},~LossCardFunOn=${X
XX},SimpleEventType=${XXX},VerifyStyles=${XXX},EventTypes=${XXX},DisableUserFunOn=${XXX},DeleteAndFunOn=$
{XXX},LogIDFunOn=${XXX},DateFmtFunOn=${XXX},DelAllLossCardFunOn=${XXX},AutoClearDay=${XXX},FirstDelayDay=
${XXX},DelayDay=${XXX},StopAllVerify=${XXX},FvFunOn=${XXX},FaceFunOn=${XXX},FingerFunOn=${XXX},CameraOpen
=${XXX},AccSupportFunList=${XXX},AutoServerFunOn=${XXX},DelayOpenDoorFunOn=${XXX},UserOpenDoorDelayFunOn=
${XXX},MultiCardInterTimeFunOn=${XXX},OutRelaySetFunOn=${XXX},MachineTZFunOn=${XXX},DSTFunOn=${XXX},CardS
iteCodeFunOn=${XXX},MulCardUserFunOn=${XXX},UserNameFunOn=${XXX},StringPinFunOn=${XXX},MaxLockCount=${XXX}
,MaxZigbeeCount=${XXX},MaxMCUCardBits=${XXX},SupportReaderType=${XXX},CmdFormat=${XXX},authKey=${XXX},Mul
tiStageControlFunOn=${XXX},MasterControlOn=${XXX},SubControlOn=${XXX},BioPhotoFun=${XXX},
BioDataFun=${XXX}
```

注释： HTTP请求方法使用：POST方法 URI使用：/iclock/registry HTTP协议版本使用：1.1

DeviceType： 考勤模块: att, 安防模块: acc
DeviceName： 设备名称

FirmVer： 固件版本号

PushVersion: push SDK版本号

MAC:设备MAC地址

CommType： 通讯类型，可有以下方式：

ethernet：有线通信

usb-4G-modem：4G网络通信

serial-wireless：wifi(串口和Sdio)通信

MaxPackageSize: 通讯包最大尺寸

LockCount： 门的数量

ReaderCount： 读头数量

AuxInCount： 辅助输入数量

AuxOutCount： 辅助输出数量

MachineType： 机器型号 一体机101

~IsOnlyRFMachine： 是否仅支持射频卡的开关参数

~MaxUserCount： 用户最大容量信息(卡最大容量信息)

~MaxAttLogCount： 考勤记录最大容量信息

~MaxUserFingerCount： 单个用户指纹最大容量

MThreshold： 指纹1：N匹配阈值

IPAddress： 有线网络IP地址

NetMask： 有线网络的子网掩码

GATEIPAddress： 有线网络的网关地址

~ZKFPVersion： 指纹算法版本号

~REXInputFunOn： 出门按钮锁定功能开关参数，一体机未用

~CardFormatFunOn： 韦根读头是否支持自定义卡格式功能开关参数

~SupAuthrizeFunOn： 超级用户功能开关参数

~Ext485ReaderFunOn： 外接485读头

~TimeAPBFunOn： 入反潜时长功能开关参数

~CtlAllRelayFunOn： 开所有继电器功能开关参数

~LossCardFunOn： 丢失卡功能开关参数

SimpleEventType： 事件合并

VerifyStyles： 支持的验证方式，共32位，前16位每个位对应不同的验证方式组合，具体参考【附录3】【验证方式码 描述】，对应的位为1表示支 持，为0表示不支持，值为200代表门禁事件。

EventTypes： 支持的事件类型，共32个字节（0~31），每个字节8位（用2个十六进制数表示），共256位（0~255），每个位对应的 功能参考【附录2】 【实时事件描述】，对应的位为1表示支持，为0表示不支持。

如 BF0FE03D300001000000000070000000000000000000000000077002001000000，其中第0个字节为BF，二进制为11111101（低位在前），第6位为0，其它位为1，说明前八位控制的功能中，只有第6位表示的联动事件不支持，其他事件都支持。

DisableUserFunOn： 黑名单功能开关参数

DeleteAndFunOn： 且关系的删除功能开关参数

LogIDFunOn: 记录id功能开关参数

DateFmtFunOn: 日期格式功能开关参数（控制用户表中的有效期时间）

DelAllLossCardFunOn: 删除所有黑名单功能开关参数

AutoClearDay: 自动清除事件日志的间隔时间

FirstDelayDay: 未认证用户登记后首次授权开门的时间

DelayDay: 未认证用户登记后首次授权开门后追加的授权时间

StopAllVerify: 停止所有的验证功能，禁卡开门功能

FvFunOn: 指静脉识别功能开关参数

FaceFunOn: 人脸识别功能开关参数

FingerFunOn: 指纹识别功能开关参数

CameraOpen: 摄像头功能开关参数

AutoServerFunOn: 后台验证开关功能开关参数

DelayOpenDoorFunOn: 延时开门功能开关，不支持

UserOpenDoorDelayFunOn: 对某用户延长开门时间功能开关，不支持

MultiCardInterTimeFunOn: 可设置多卡刷卡间隔功能开关，不支持

OutRelaySetFunOn: 输出配置功能开关参数，不支持

MachineTZFunOn: 可设置机器时区功能开关参数，不支持

DSTFunOn: 夏令时功能功能开关参数，不支持

CardSiteCodeFunOn：卡的sitecode功能开关。

MulCardUserFunOn：多卡用户功能开关（设备的业务逻辑判断使用）

UserNameFunOn：用户姓名功能开关

StringPinFunOn：支持字符串工号

MaxLockCount:最大门数量

MaxZigbeeCount：最大Zigbee数量

MaxMCUCardBits：最大卡二进制位数

SupportReaderType：支持读头类型

CmdFormat: 命令格式，具体见【附录14】

MultiStageControlFunOn：多级控制参数，具体见【附录15】

MasterControlOn：多级控制参数，具体见【附录15】

SubControlOn：多级控制参数，具体见【附录15】

BioPhotoFun：支持比对照片参数

BioDataFun：支持可见光人脸模板参数

authKey：通信秘钥

AccSupportFunList:根据位来获取功能支持参数（该参数软件不可修改），如果没传，默认全0

注释：位置值是指AccSupportFunList参数值的字符串位置，从0开始并从左到右数

位置值	备注
0	0不支持;1支持 读头禁用功能
1	0不支持;1支持 485读头加密功能
2	0不支持;1支持 门锁定功能
3	0不支持;1支持 紧急双开、紧急双闭
4	0不支持;1支持 通过门磁及继电器状态对应到Wiegand读头灯的红绿
5	0不支持;1支持 长姓名的功能

```
      6     0不支持;1支持 韦根格式带sitecode
      7     0不支持;1支持 一人多卡功能（软件的业务逻辑判断使用，当为1，软件会下发设置设备参数
MulCardUserFunOn=1）
      8     0不支持;1支持 辅助输入、出门按钮受时间段控制
      9     0不支持;1支持 临时用户每天自动删除功能
     10     0不支持;1支持 支持wg test功能
     11     0不支持;1支持 支持account命令
     12     0不支持;1支持 不同用户不用时间段不同验证方式
     13     0不支持;1支持 控制485读头Led
     14     0不支持;1支持 门支持多种韦根格式，保留
     15     0不支持;1支持 权限表中doorid只表示一个门
     16     0不支持;1支持 用户超级权限独立表
     17     0不支持;1支持 支持添加门属性功能
     18     0不支持;1支持 读头属性功能
     19     0不支持;1支持 辅助输入属性表功能
     20     0不支持;1支持 辅助输出属性表功能
     21     0不支持;1支持 门参数表功能
     22     0不支持;1支持 反潜表功能
     23     0不支持;1支持 互锁表功能
     24     0不支持;1支持 Wireless-serial功能
     25     0不支持;1支持 4G-usb通信功能
     26     0不支持;1支持 表支持DevID字段（控制协议中是否有DevID字段）
     27     0不支持;1支持 是否支持从机
     28     0不支持;1支持 是否支持双网卡
     29     0不支持;1支持 是否支持授权表
     30     0不支持;1支持 是否身份证登记
     31     0不支持;1支持 设备是否支持主从切换
     32     0不支持;1支持 设备是否支持混合读头,韦根/485读头
     33     0不支持;1支持 设备是否支持韦根数据处理 字节交换 位反转 字节交换
     34     0不支持;1支持 设备是否支持desfire控卡功能
     35     0不支持;1支持 支不支持下发二维码命令
     36     0不支持;1支持 是否支持广告下发命令zksq200新增
     37     0不支持;1支持 是否支持室内机功能zksq200新增
     38     0不支持;1支持 设备是否支持上传比对照片功能
     39     0不支持;1支持 设备是否支持上传一体化可见光人脸模板功能
```

服务器正常响应：

```
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}
```

注释：

```
RegistryCode：服务器生成随机数，最长32个字节。
如果注册失败，则不会返回注册码，而是返回“406”
```

示例：

```
客户端请求消息：
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,LockCount=1,ReaderCount=2,AuxInCount=0,AuxOutCou
nt=0,MachineType=101,~IsOnlyRFMachine=0,~MaxUserCount=50,~MaxAttLogCount=10,~MaxUserFingerCount=10,MThres
hold=60,IPAddress=192.168.213.221,NetMask=255.255.255.0,GATEIPAddress=192.168.213.1,~ZKFPVersion=10,Icloc
kSvrFun=1,OverallAntiFunOn=0,~REXInputFunOn=0,~CardFormatFunOn=0,~SupAuthrizeFunOn=0,~ReaderCFGFunOn=0,~R
eaderLinkageFunOn=0,~RelayStateFunOn=1,~Ext485ReaderFunOn=0,~TimeAPBFunOn=0,~CtlAllRelayFunOn=0,~LossCard
FunOn=0,SimpleEventType=1,VerifyStyles=ff7f0000,EventTypes=BF0FE03D3000010000000000700000000000000000000000
000077002001000000,DisableUserFunOn=0,DeleteAndFunOn=0,LogIDFunOn=0,DateFmtFunOn=0,DelAllLossCardFunOn=0,
AutoClearDay=0,FirstDelayDay=0,DelayDay=0,StopAllVerify=0,FvFunOn=0,FaceFunOn=0,FingerFunOn=1,CameraOpen=
1,AccSupportFunList=01010100001110000001110101000110 10,AutoServerFunOn=1,DelayOpenDoorFunOn=1,UserOpenDoo
rDelayFunOn=1,MultiCardInterTimeFunOn=1,OutRelaySetFunOn=1,MachineTZFunOn=1,DSTFunOn=1,CardSiteCodeFunOn=
1,MulCardUserFunOn=1,UserNameFunOn=1,StringPinFunOn=1,MaxLockCount=4,MaxZigbeeCount=3,MaxMCUCardBits=37,S
upportReaderType=1,authKey=dassas
服务器响应：
HTTP/1.1 200 OK
Server: Apache-Coyote/1.1
Set-Cookie: JSESSIONID=30BFB04B2C8AECC72C01C03BFD549D15; Path=/; HttpOnly
Content-Type: text/plain;charset=ISO-8859-1
Content-Length: 23
```

```
Date: Mon, 09 Jan 2017 01:31:59 GMT
RegistryCode=Uy47fxftP3
```

7.5 下载配置参数

应用场景：

```
上一步注册请求成功返回注册码后，需要设备主动获取服务器的配置参数，才算完成整个初始化过程
```

客户端请求消息：

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

服务器响应：

```
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}
```

注释：

```
HTTP请求方法使用：GET方法
URI使用：/iclock/push
HTTP协议版本使用：1.1
ServerVersion： 服务器版本号
```

ServerName: 服务器名称

ErrorDelay: 联网失败后客户端重新联接服务器的间隔时间（秒），建议设置30~300秒，如果不配置客户端默认值为30秒

RequestDelay: 客户端获取命令请求间隔时间（秒），如果不配置客户端默认值为30秒

TransTimes: 定时检查传送新数据的时间点，如果不配置默认为"12:30;14:30”

TransInterval: 检查是否有新数据需要传送的间隔时间（分钟）,如果不配置客户端默认为2分钟

TransTables: 需要检查新数据并上传的数据，默认为"User Transaction",需要自动上传用户和门禁记录

Realtime: 客户端是否实时传送新记录。为1表示有新数据就传送到服务器,为0表示按照 TransTimes 和 TransInterval 规定的时间传送,如果不配置客户端默认值为1

SessionID: PUSH 通信会话 ID, 今后设备的请求都需要此字段来计算新的 Token

TimeoutSec: 设置网络超时时间，如果不配置客户端默认 10秒

特别说明:

token计算方法： 客户端对注册码（RegistryCode）、序列号（SerialNumber）和SessionID这三个的值进行MD5加密，之后转化为16进制显示字符串。该16进制显示的字符串即为token。

示例：

客户端请求消息：

```
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
```

服务器响应：

```
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

8. 授权

- 只针对主控设备授权请求，主控设备定义见【附录15】
- 主控设备主动推送子控授权表信息到软件

• 设备启动时推送授权表信息到软件

• 授权表信息变化时推送信息到软件

• 协议：

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

• Host: 113.108.97.187:8081

• DeviceAuthorize SN=%?\tOnline=%?\tIsAuthorize=%?\tAuthorizeInfo=%?\tRegisterInfo=%?

• 注释

• SN：二级控制器序列号

• Online：二级控制器是否在线

• IsAuthorize：表示是否授权，0未授权、1授权进行中、2授权完成

• AuthorizeInfo：未授权时二级控制器推送的简单设备信息

• RegisterInfo：软件授权后二级控制器推送的设备注册信息

- 软件根据推送的授权表信息对二级控制器进行授权添加

• 软件下发命令对二级控制器进行授权

• BioIR9000根据软件下发的授权信息允许相应的二级控制器推送相关注册信息

• 二级控制器推送相应的设备注册信息到BioIR9000

• BioIR9000立即将二级控制器注册信息推送软件

• 协议：

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

- 软件接收到二级控制器注册信息后下发最终授权

• 软件接收到二级控制器注册信息后下发最终授权命令

• BioIR9000根据软件下发的授权信息允许相应的二级控制器接入

• 完成授权

• 协议：

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

- 软件删除授权的二级控制器

```
• 协议：
• C:295:DATA DELETE DeviceAuthorize SN=%?
```

9. 心跳

应用场景：

```
用于与服务器保持心跳。当处理大数据上传时，用ping保持心跳，大数据处理完，用getrequest保持心跳
```

客户端请求消息：

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

服务器响应：

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

注释：

```
HTTP请求方法使用：POST方法
URI使用：/iclock/ping
HTTP协议版本使用：1.1
```

示例：

```
客户端请求消息：
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
服务器响应：
HTTP/1.1 200 OK
Server: Apache-Coyote/1.1
Content-Length: 2
Date: Tue, 10 Jan 2017 07:42:41 GMT
OK
```

10. 上传

10.1 上传的方式

实时上传：

> 即立即上传消息，设备本身默认支持，服务器是可以控制的(详细见“初始化信息交互”的“Realtime”参数)

间隔上传：

> 即隔一段时间间隔后上传消息，具体的间隔时间服务器是可以控制的(详细见“初始化信息交互”的“TransInterval”参数)

定时上传：

> 即在具体的时间点上传消息，具体的上传时间点服务器是可以控制的(详细见“初始化信息交互”的“TransTimes”参数)

10.2 上传实时事件

实时事件是指设备实时产生的事件，考虑网络延时，一般 15 秒内的事件都叫实时事件。实时事件的事件码和描述参见【附录2】

应用场景：

> 属于实时事件的情况发生时，需要设备立即上传实时事件的信息到软件。

客户端请求消息：

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

服务器响应：

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

注释：

```
HTTP请求方法使用：POST方法
URI使用：/iclock/cdata
HTTP协议版本使用：1.1
表名：rtlog
请求实体：${DataRecord}，实时事件数据，数据格式如下
time=${Time}${HT}pin=${Pin}${HT}cardno=${XXX}${HT}eventaddr=${XXX}${HT}event=${XXX}${HT}inoutstatus=${XXX}
${HT}verifytype=${XXX}${HT}index=$(xxx)
time: 时间，格式：XXXX-XX-XX XX:XX:XX
pin: 工号
cardno: 卡号
eventaddr: 事件点，默认为 doorid
event： 事件码，参阅【附录2】
inoutstatus： 出入状态，0 表示入,1 表示出
verifytype: 当前验证方式
index: 门禁记录id，门禁记录在设备中存储的ID，具有唯一性
```

示例：

```
客户端请求消息：
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
服务器响应：
HTTP/1.1 200 OK
Server: Apache-Coyote/1.1
Content-Length: 2
Date: Tue, 10 Jan 2017 03:49:32 GMT
OK
```

10.3 上传实时状态

实时状态是指物理设备当前状态或当前逻辑状态。如继电器、门磁、道闸、报警等。

应用场景：

```
用于定时向软件上传当前设备的一些门禁状态信息 或 当某些门禁状态发生变化时立即通知软件
```

客户端请求消息：

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

服务器响应：

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

注释：

HTTP请求方法使用：POST方法

URI使用：/iclock/cdata

HTTP协议版本使用：1.1

表名：rtstate

请求实体：${DataRecord}，实时状态数据，数据格式如下

time=${XXX}${HT}sensor=AABB${HT}relay=CC${HT}alarm=DDEEFFGGHHIIJJKK

time： 表示当前时间，格式：XXXX-XX-XX XX:XX:XX

sensor： 表示门磁状态，每个门占用两个二进制位，AA 表示表示 1-4 门，BB 表示 5-8 门，1 门占用第一个字节的第 1、2 位，依此类推，0b00 表示当前门磁类型被设为无门磁，0b01 表示当前门是关着的(有门磁)，0b10 门是开着的(无门磁)；

relay： 表示继电器状态，每个门占用一个二进制位，0b0 表示继电器吸合，0b1 表示继电器断开，第一位为 1 门继电器状态，依此类推；

alarm： 表示报警状态，4个门占用一个字节，每个门占用8个二进制位。最多可表达 8 种报警，DD 为 1 门报警状态，依此类推，目前报警定义如下：

第一位： 意外开门事件

第二位： 防拆报警

第三位： 胁迫密码报警

第四位： 胁迫指纹报警

第五位： 门磁超时报警

第六位-第八位：保留

示例：

客户端请求消息：

```
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
```

服务器响应：

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

```
OK
```

10.4 上传命令返回结果

应用场景：

```
用于服务器下发的命令结束后，返回执行结果以告知服务器命令的执行情况
```

客户端请求：

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

服务器响应：

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

注释：

```
HTTP请求方法使用：POST方法
URI使用：/iclock/devicecmd
HTTP协议版本使用：1.1
请求实体：${DataRecord}，命令返回结果，数据格式如下
    ID=${XXX}&Return=${XXX}&CMD=${XXX}&SN=${XXX}

    ID： 命令ID，软件下发缓存命令时带有
    Return： 命令执行情况返回值
        根据多级控制参数，见【附录15】
            当服务器下发的子控命令到主控设备，主控收到返回-5000，表示已接收待执行。
            当服务器下发的主控命令，或者非多级设备命令，或者子控执行命令完毕后，返回值为通用错误码参见【附录1】，大于等于0表示成功，小于0表示失败。
    CMD： 命令类型
```

SN：子控序列号，只有子控执行命令返回的才有SN。子控定义，见【附录15】

软件下发命令时带有多条记录之间使用${LF}连接

示例：

客户端请求：

```
POST /iclock/devicecmd?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
Content-Type: application/push;charset=UTF-8
Content-Language: zh-CN
Content-Length: 35
ID=1&Return=0&CMD=DATA UPDATE
```

服务器响应：

```
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
```

10.5 上传用户信息

应用场景：

设备主动上传用户信息，常用于设备登记用户后自动上传用户给服务器

客户端请求消息：

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

注释：

HTTP请求方法使用：POST方法

URI使用：/iclock/cdata

HTTP协议版本使用：1.1

表类型table：tabledata

表名：user

count： 本次消息上传的用户数

请求实体：${DataRecord}，用户数据，数据格式如下

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： 用户在设备中的ID

cardno： 卡号，卡号是无符号整型数据存储，软件下发的值支持两种格式上传

a、十六进制数据，格式为[%02x%02x%02x%02x]，从左到右表示第1、2、3、4个字节，如卡号为123456789，则为：Card=[15CD5B07]

b、字符串数据，如卡号为123456789，则为：Card=123456789

pin： 用户的工号

password： 用户的密码，最大长度为6位

group： 用户的门禁组

starttime： 用户有效期开始时间

如果设备参数DateFmtFunOn的值为1，举例：starttime=583512660，starttime由【附录5】算法计算出来，使用【附录6】的方法得到年月日时分秒。

如果设备参数DateFmtFunOn的值为0，格式为YYYYMMDD

endtime： 用户有效期结束时间

如果设备参数DateFmtFunOn的值为1，举例：endtime=583512660，endtime由【附录5】算法计算出来，使用【附录6】的方法得到年月日时分秒。

如果设备参数DateFmtFunOn的值为0，格式为YYYYMMDD

name： 用户姓名，当设备为中文时，使用的是GB2312编码，其他语言时，使用UTF-8编码

privilege： 用户权限

disable： 是否是黑名单

verify： 支持的验证方式

多条记录之间使用${LF}连接

服务器响应：

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

注释：

user： 本次服务器接收到的用户数量

示例：

客户端请求消息：

```
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
```

服务器响应：

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

user=1
```

10.6 上传身份证信息

应用场景：

设备主动上传身份证信息，常用于设备登记身份证后自动上传身份证给服务器

客户端请求消息：

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

```
${DataRecord}
```

注释：

```
HTTP请求方法使用：POST方法
URI使用：/iclock/cdata
HTTP协议版本使用：1.1
表类型table：tabledata
表名：identitycard
count： 本次消息上传的用户数
请求实体：${DataRecord}，用户数据，数据格式如下
    identitycard
pin=${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}
    注：
    pin:用户工号
    SN_Num：身份证物理卡号
    ID_Num：公民身份证号码
    DN_Num：居民身份证证卡序列号（卡体管理号）
    Name：姓名，使用UTF-8编码
    Gender：性别代码
        1,"男"
        2,"女"
    Nation：民族代码
        0,"解码错"
        1,"汉"
        2,"蒙古"
        3,"回"
        4,"藏"
        5,"维吾尔"
        6,"苗"
        7,"彝"
        8,"壮"
        9,"布依"
        10,"朝鲜"
        11,"满"
        12,"侗"
        13,"瑶"
        14,"白"
```

15, "土家"
16, "哈尼"
17, "哈萨克"
18, "傣"
19, "黎"
20, "傈僳"
21, "佤"
22, "畲"
23, "高山"
24, "拉祜"
25, "水"
26, "东乡"
27, "纳西"
28, "景颇"
29, "柯尔克孜"
30, "土"
31, "达斡尔"
32, "仫佬"
33, "羌"
34, "布朗"
35, "撒拉"
36, "毛南"
37, "仡佬"
38, "锡伯"
39, "阿昌"
40, "普米"
41, "塔吉克"
42, "怒"
43, "乌孜别克"
44, "俄罗斯"
45, "鄂温克"
46, "德昂"
47, "保安"
48, "裕固"
49, "京"
50, "塔塔尔"
51, "独龙"
52, "鄂伦春"
53, "赫哲"
54, "门巴"
55, "珞巴"

```
        56,"基诺"
        57,"编码错"
        97,"其它"
        98,"外国血统"
    Birthday：出生日期（格式：yyyyMMdd）
    Valid_Info：有效期,开始日期和结束日期（格式：yyyyMMddyyyyMMdd）
    Address：地址，使用UTF-8编码
    Additional_Info：机读追加地址，使用UTF-8编码
    Issuer：签发机关，使用UTF-8编码
    Photo：身份证存储的照片数据，数据是加密的，并转化成base64数据内容进行传输。
    PhotoJPG：身份证存储的照片数据，对数据解密，并转化成base64数据内容进行传输。
    FP_Template1：指纹1_指纹特征数据
    FP_Template2：指纹2_指纹特征数据
    Reserve：保留字段
    Notice：备注信息，使用UTF-8编码
    多条记录之间使用${LF}连接
```

服务器响应：

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

注释：

```
identitycard:  本次服务器接收到的身份证数量
```

示例：

```
客户端请求消息：
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=张三 Gender=1
Nation=1    Birthday=19911218   ValidInfo=2013091220230912  Address=广东省东莞市塘厦镇xxxxx
AdditionalInfo= Issuer=xxx公安局   Photo=V0xmAH4AMgAA/apmyEd8
PhotoJPG=Y0tKqWNPzyEd8Pn0EhRsgAAjeWPxiUzLaPU1w=
FPTemplate1=QwGIEgELUQAAAAAAAAAAAAAAACgBmnlcAf///////0wZqfwuJK78PzJg/lw
FPTemplate2=QwGIEgEQUAAAAAAAAAAAAAAAADIBmkbyAP///////3UYCPxXQs38qEVW/rpLovwyUBv8WV8 Reserve=    Notice=
服务器响应：
HTTP/1.1 200 OK
Server: Apache-Coyote/1.1
Content-Length: 6
Date: Thu, 12 Jan 2017 00:49:31 GMT
identitycard=1
```

10.7 上传指纹模版

应用场景：

```
设备主动上传指纹模板，常用于设备登记用户的指纹后自动上传给服务器
```

客户端请求消息：

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

注释：

```
HTTP请求方法使用：POST方法
URI使用：/iclock/cdata
HTTP协议版本使用：1.1
count： 一次请求里包含的指纹模板数量
表类型table：tabledata
表名：user
请求实体：${DataRecord}，指纹模板，数据格式如下
```

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： 表示数据类型为指纹模板，**支持的指纹算法版本不超过10.0**

size： 指纹模板转换为base64格式后的大小

uid： 指纹模板在设备中的ID

pin： 指纹模板对应的用户的工号

fingerid： 每个用户的指纹id，用于区分不同的手指，0~9表示普通指纹，普通指纹id加上16（如6+16=22）表示胁迫指纹。设备中的胁迫指纹上传到软件时需加上16.

valid： 设备内的胁迫指纹标志，0表示无效指纹，1表示普通指纹，3表示胁迫指纹，软件上则以fingerid来区分普通和胁迫指纹。上传到软件时如果valid值为3需先设置为1。

template： 指纹模板数据，传输指纹模版时，需要对原始二进制指纹模版进行base64编码

resverd：

endtag：

多条记录之间使用${LF}连接

服务器响应：

```
HTTP/1.1 200 OK
Server: Apache-Coyote/1.1
Content-Length: ${XXX}
Date: ${XXX}
templatev10=${XXX}
```

注释：

templatev10： 服务器接收到的10.0算法指纹模板数量

示例：

客户端请求消息：

```
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=TAlTUzlxAAAFSksECAUHCc7QAAAdS2kBAAAAhfcvU0o3AlAPswD/AltFJQBWAG0PcABfSo0PogBhAE0PeEpnAlEPqACjAAlF
cAB0AAQPvwCBSoEPvACTANEPGUqcA0oPSgBkAGtE0wCpA000AACpSo8PFgC9AKwPWUrBAGcPLAAPAFpFjQDNAHYPEADlShoPagDRAKcPf
0riAFcPLQAtAFxElADtAGEPpwDqSkoP3ADxA0MP/krzACMPmgA8ABRFHgD8ANk0+AAFS00PLAACAZU0x0oDAZg0dADPATdFvAA0AY8OnA
AKSzoPyAAVAU8NT0oWAT8PowDkAQNF3wAhAYENUQApSxIPQQAwAfsPBUs5AYIOUAD4AS5FEAE9AYwO8QBES0A0ewBHAd8PR0pRATIPGwC
WAV9FahcnAzN/9wRDQT+H/+9bh+JrUMtkBblAFH4s/XNDXH1GAPr51YZQy/ePqYVZhap9qDfHA0f5oYvaCB7Ep/zegJt8RIE7Q5aEgYEm
BC0RoDUjdV8bLQ8ngAM0PAbG+bJ+fPtnQvN1aSNCFEMVE8ef9Yf5wf6EDm9SzPaZDwXjRIJsQVAPUFOtDxPslLOD6F4YXgQDkBNH4/z/C
bem5bMP9leGuYiBgmuMtLZgD933wXjoBPSxOQsdEdJ0DPv/ohfewPslB5+ItLY8C9rwjP9s9QhALATt95EDJPVnMm9+lfqW9t8CUEfP+R
PgYRVY/RRJCPmaApf7MO20sNDvgQ2LD25rZFsYBTkbRgk3805c8PEpCg70yxgjYAodP0PuHtgjJQQAAvloUg3FOQVahMBKQ/8FxYoHQz4
lAHoHCY5TCkokD/04wMAF/m2KYAsAKhYDllFJRAELK/TBMTpHx30TAAky/cP1/lYRwP/AcQYAlzKGi8NpCwBaNsNCxQF2BgBSOH29wQFK
tjsTfQ0A0VDxHUDAVf8LA01U9XE8WwQAuVvJQQlKn16Mc8GAB2QCSqdhAzZMDMWYZ8xThHfACQC8Y4PKaMEKAIFmw0/EBsEHAKtnCTvB+
hMIAHloff9GcQlKdHQGXsH/BDb6QgF2gobCwUbBAEp/hQA2CgC3h4U0wsE/BAAsSWdeWQESjun//4L+bbVC/0wGAL9XDPiLWwYAv5cTgc
EQSguc58AvwPZgXYr8ZBEAwatWwnfBw8H/wsCF1QC25o1xg3HB/lEJBf+sDP9DSxbFGr+jaf5MNv7/OlVitxcABb7i/pP/NwtHRMBNBQC
dwWzOEwDRypeEBcKDtMOTZwQAKQ5gakQBkMwDLv8FMsSKmgcA2cwW8jwASmfQacKICMWJ1T3AxMHBkwTF19RQQw0AaNVkB8GBFsBbDwB3
5KXFicLEUMHCEgBU6nKNxcbBwcN8Vsl5RwFm6977wDr8K3L+BACZ7QPnDgXY8WfFyMLCAcDGi8P/wP8DADryG7UKAF7zU8M6w8fCgwsAZ
PNMBMRsw8MGAN3zJJDBH0sU9KBqc2+vw8QuxXLB/8EExfzzbW4FAJ33DDh2AUqU+S3GwgPFnPhZwQUQHABWBIwBWogBMKEIEPwDVTFZBx
A/BEy2TgJaKgVWgyYF1ZIDasKUGRDTCIv+xInDxP+ywcIBgvqIw//DwH8Q1XQIfonBwYh+weMJFRMTQMPAIoHCEEJQQnDEwQMQiRo4igQ
Qpx8DwAb8GIsUIaTB/nSuwoUliIPA/8HCr/sCWqYIDJAzCdWYK13DjF4gERXrpMR9d8CIhMKGBmLHi/7Bwf39+cEQSgo2nQQQVEDogwBa
f0oewnME1f9hN0tSQgALQ8QABUFEUg== resverd= endtag=

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

template=T0VTUzlxAAAFpqkECAUHCc7QAAAdp2kBAAAAhUs1zKYXAPUPzgD2AP+pCgE6AAY0FgBDpgEPtQBSAL0Pk6ZVAPYP8ACRAA0p
mABoAPQPDABopgcPNgCCACs02KaHAIYPqQBNAPiprACTAHYPMQCRphMPrwCeAMcPN6aIAOoOEQBvA0etQQCqAGoO2wC3pmQLpACyALMPI
6a4A0oLZgAEAPWpvwDJAA0PxAHRphwOqADWAM4PEqbcA0IPuQAnAISpowDIAAMPjwDjpukPYQDqACsPq6buABEP/wA6ACapaAAAAfMP1A
ADp2UPkQALAblPqqYPASAPbgDRAfGplQAVARYPvAAZp/8PFQAgAaI0aKYmAf8P9wDiASipIQAqAS803AA2p+IOpgAzAf00Z6Y0Af4P3AD
yATGpbAA4AXgO9gBMp+UPZQBMAa8Ps6ZUAS4NoACZAUKowQBdASMNAP/CUtp/vf1BByeG26fyCgMPNQIT/Pcuh4T6eDJ1AAWPJr76Dghe
ASIATKdzfbr5ooDnkUOs2wXKhueK1HenJnIRtIu5/vcJQdG9e1YABY/KDyqpOY81BtWObIDopPyGtXy5+1/5rCaDgI4FOH90hADY4HbWA
Cpp8YdopDAL1gYqiyoluCjcBxKZpv/aD+61xAiRjWEDt/rjJGd/XXCZebyCnFJM+HmGof5z9nJeGArF/loCyBBINiQLpvoiC3LzwKdABr
r0Vf8jhlImPWNxew5YEA/jDiMQpfjR9YABJQWs6WH7HQwgAHxNUIXVB00FoADELtwBvvgeDiYPYF8QGeYwVX1/AVIm3An2JXoOSYtYp7y
WOginEHYIkdnIeFkSOgFmfULb5PbRbuJuqQ/ATAP6LRo+JMcS/VHj3DvwsHjtYw+9ASBQAQMF4k4Apr8BdML/wMAAx6Fq/0sLAOfCgMFk
/GvAbwMAKgxyZgwA8BR3wKNWbmcQAQQeg34HUsVZcVQSAQsmRXFh+8P/a2UGAdcsf1llEgEMNICv/4FYxUPBVAUAEkYFZysKARVHkwbAj
P4GALlQ9y+CCAU0VHBq/1cNxbFQ1v/CwGvB/zjGx6UB9FUJwgrFnWZSwcH9//5RyACQz3DBWsDB/gD+N6wBnWr0wP86wPlmSgcAzWOA9E
YWpxR+l4WWwQXAxWRkbQQArYQ/MAimpll6wsBpl8BloAGtigBK/sgArDZ2c37AwMC+AwVekxDBAwAZUlf7rwGxk/r+//BUHacNk5NtecM
7wcRma2vA/cXDBBYEsJWew/+HwFxqbviYDgCpl3cEa3bPbgkAs5r97jf7qQGrn3p+agXBxlhdCACzoAaRNgCmobB6wsDBzwCtFPw1//7A
V8YAkB92wg8Aabw1wDiJ/v82wAQA08BnYP4KAGLDab1kUqoBdMV0iclwcVn/vwJAMHKzP5E8AcAPNBkcwYLBQrSBv7/OP+GBgUC13qEw
AwAadkMWP7/wDZCDcW25CCIc8NnwBLFTeFBVf9GwP7+OjtRtwFk5uvAODr9+1lH/8IxCwBy5wxb/j7/VQgAmO1oZHFXBwCy7cz9+IjB/x
AAqu5JxMdZw39w/sljyQC0VBI4NcBGGsQS8gRrwnnAfsKx/4P8NREAbf7w8P34Z/81wEDBA9QCBIT/EBCrC4xUwYJkT/0/CRCzyBf4Zv8
ywggQsdceNllzBxCXFgzkOQ22fxv9/f78O/3FoxFwIv3/Ks0Qdl4B/xUfCBCsKmFZwET/BRD77yk4oxH2Ky1VBdWYKYv8HwQQZTQy//+i
EaU20jkE1aozlikGEGk5bQdRAraC0YDHQ//BENicMEwGEH8/rMLFlAcQjkN6xzrA/rcRNUfnwVs5wvhm/fz+/jYD1Y9O6vwDEINMXgQFF
cVPaf/A/RrVAVxPwYH/wUb9jv74mv5LX1JCAM5DBKYBC0VS resverd= endtag=

服务器响应：

```
HTTP/1.1 200 OK
Server: Apache-Coyote/1.1
Content-Length: 13
Date: Thu, 12 Jan 2017 00:49:32 GMT
templatev10=2
```

10.8 上传比对照片

应用场景：

```
可见光设备上传比对照片，常用于设备登记用户比对照片后自动上传给服务器
```

客户端请求消息：

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

注释：

```
HTTP请求方法使用：POST方法
URI使用：/iclock/cdata
HTTP协议版本使用：1.1
Host头域：${Required}
Content-Length头域：${Required}
其他头域：${Optional}
请求实体：${DataRecord}，比对照片数据，数据格式如下
biophoto${SP}pin=${XXX}${HT}filename=${XXX}${HT}type=${XXX}${HT}size=${XXX}${HT}content=${XXX}
    注：
pin=${XXX}：用户工号。
filename=${XXX}：生物特征图片的文件名，目前只支持jpg格式。
type=${XXX}：生物识别类型。
    值 意义
    0 通用的
    1 指纹
    2 面部（近红外）
    3 声纹
    4 虹膜
```

5 视网膜

6 掌纹

7 指静脉

8 手掌

9 可见光人脸

size=${XXX}：生物特征图片base64 编码之后的长度

content=${XXX}：传输时，需要对原始二进制生物特征图片进行base64 编码。

多条记录之间使用${LF}连接。

服务器响应：

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

OK:${XXX}
```

注释：

HTTP状态行：使用标准的HTTP协议定义

HTTP响应头域：

Content-Length头域：根据HTTP 1.1协议，一般使用该头域指定响应实体的数据长度，如果是在不确定响应实体的大小时，也支持Transfer-Encoding: chunked，Content-Length及Transfer-Encoding头域均是HTTP协议的标准定义，这里就不在详述。

响应实体：当服务器接收数据正常并处理成功时回复OK:${XXX}，其中${XXX}表示成功处理的记录条数，当出错时，回复错误描述即可。

示例：

客户端请求：

```
POST /iclock/cdata?SN=0316144680030&table=biophoto&Stamp=9999 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........
```

服务器响应：

```
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
```

10.9 上传抓拍照片

应用场景：

```
可见光设备上传抓拍照片，常用于设备自动上传抓拍照片给服务器
```

客户端请求消息：

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

注释：

```
HTTP请求方法使用：POST方法
URI使用：/iclock/cdata
HTTP协议版本使用：1.1
count=${XXX}： 一次请求里包含的抓拍照片数量
Host头域：${Required}
Content-Length头域：${Required}
其他头域：${Optional}
请求实体：${DataRecord}，考勤照片数据，数据格式如下
pin=${XXX}${HT}sn=${XXX}${HT}size=${XXX}${HT}photo=${XXX}
    注：
pin=${XXX}：考勤照片名称。
sn=${XXX}：设备序列号。
size=${XXX}：抓拍照片原始大小。
photo=${XXX}：传输时，需要对原始二进制生物特征图片进行base64 编码。
```

多条记录之间使用${LF}连接。

服务器响应：

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

ATTPHOTO=${XXX}
```

注释：

HTTP状态行：使用标准的HTTP协议定义

HTTP响应头域：

Content-Length头域：根据HTTP 1.1协议，一般使用该头域指定响应实体的数据长度，如果是在不确定响应实体的大小时，也支持Transfer-Encoding: chunked，Content-Length及Transfer-Encoding头域均是HTTP协议的标准定义，这里就不在详述。

响应实体：当服务器接收数据正常并处理成功时回复ATTPHOTO=${XXX}，其中${XXX}表示成功处理的照片数量，当出错时，回复错误描述即可。

示例：

客户端请求：

```
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/2wBDABsSFBcUERsXFhceHBsgKEIrKCUlKFE6PTBCYFVlZF9VXVtqeJmBanGQc
1tdhbWGkJ6jq62rZ4C8ybqmx5moq6T/2wBDARweHigjKE4rK06kbl1upKSkpKSkpKSkpKSkpKSkpKSkpKSkpKSkpKSkpK
SkpKSkpKSkpKT/wAARCAUAAtADASIAAhEBAxEB/8QAHwAAAQUBAQEBAQEAAAAAAAAAAAECAwQFBgcICQoL/8QAtRAAAgEDAwIEAwU
FBAQAAAF9AQIDAAQRBRIhMUEGE1FhByJxFDKBkaEII0KxwRVS0fAkM2JyggkKFhcYGRolJicoKSo0NTY3ODk6Q0RFRkdISUpTVFVWV1hZ
WmNkZWZnaGlqc3R1dnd4eXqDhIWGh4iJipKTlJWWl5iZmqKjpKWmp6ipqrKztLW2t7i5usLDxMXGx8jJytLT1NXW19jZ2uHi4+Tl5ufo6
erx8vP09fb3+Pn6/8QAHwEAAwEBAQEBAQEBAQAAAAAAAAECAwQFBgcICQoL/8QAtREAAgECBAQDBAcFBAQAAQJ3AAECAxEEBSExBhJBUQ
dhcRMiMoEIFEKRobHBCSMzUvAVYnLRChYkNOEl8RcYGRomJygpKjU2Nzg5OkNERUZHSElKU1RVVldYWVpjZGVmZ2hpanN0dXZ3eHl6go0
EhYaHiImKkpOUlZaXmJmaoqOkpaanqKmqsrO0tba3uLm6wsPExcbHyMnK0tPU1dbX2Nna4uPk5ebn6Onq8vP09fb3+Pn6/9oADAMBAAIR
AxEAPwBXXIPFVpl+fbHWrn1pjJnOKgZ

```

```
服务器响应：
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: 10
Connection: close
Pragma: no-cache
Cache-Control: no-store

ATTPHOTO=1
```

10.10 上传用户照片

应用场景：

```
设备上传用户照片，常用于设备自动上传用户照片给服务器
```

客户端请求消息：

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

注释：

```
HTTP请求方法使用：POST方法
URI使用：/iclock/cdata
HTTP协议版本使用：1.1
count=${XXX}： 一次请求里包含的用户照片数量
Host头域：${Required}
Content-Length头域：${Required}
其他头域：${Optional}
请求实体：${DataRecord}，考勤照片数据，数据格式如下
userpic pin=${XXX}${HT}filename=${XXX}${HT}size=${XXX}${HT}content=${XXX}
    注：
pin=${XXX}：用户工号。
filename=${XXX}：用户照片的文件名，目前只支持jpg格式。
```

size=${XXX}：用户照片base64编码之后的长度。

content=${XXX}：传输时，需要对原始二进制用户照片进行base64编码。

多条记录之间使用${LF}连接。

服务器响应：

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

userpic=${XXX}
```

注释：

HTTP状态行：使用标准的HTTP协议定义

HTTP响应头域：

Content-Length头域：根据HTTP 1.1协议，一般使用该头域指定响应实体的数据长度，如果是在不确定响应实体的大小时，也支持Transfer-Encoding: chunked，Content-Length及Transfer-Encoding头域均是HTTP协议的标准定义，这里就不在详述。

响应实体：当服务器接收数据正常并处理成功时回复userpic=${XXX}，其中${XXX}表示成功处理的照片数量，当出错时，回复错误描述即可。

示例：

客户端请求：

```
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/2wBDABsSFBcUERsXFhceHBsgKEIrKCUlKFE6PTBCYFVlZF9VXVtqeJmBanG
Qc1tdhbWGkJ6jq62rZ4C8ybqmx5moq6T/2wBDARweHigjKE4rK06kbl1upKSkpKSkpKSkpKSkpKSkpKSkpKSkpKSk
pKSkpKSkpKSkpKT/wAARCAOoAnQDASIAAhEBAxEB/8QAHwAAAQUBAQEBAQEAAAAAAAAAAAECAwQFBgcICQoL/8QAtRAAAgEDAwIEA
wUFBAQAAAF9AQIDAAQRBRIhMUEGE1FhByJxFDKBkaEII0KxwRVS0fAkM2JyggkKFhcYGRolJicoKSo0NTY3ODk6Q0RFRkdISUpTVFVWV1
hZWmNkZWZnaGlqc3R1dnd4eXqDhIWGh4iJipKTlJWWl5iZmqKjpKWmp6ipqrKztLW2t7i5usLDxMXGx8jJytLT1NXW19jZ2uHi4+Tl5uf
o6erx8vP09fb3+Pn6/8QAHwEAAwEBAQEBAQEBAQAAAAAAAAECAwQFBgcICQoL/8QAtREAAgECBAQDBAcFBAQAAQJ3AAECAxEEBSExBhJB
UQdhcRMiMoEIFEKRobHBCSMzUvAVYnLRChYkNOEl8RcYGRomJygpKjU2Nzg5OkNERUZHSElKU1RVVldYWVpjZGVmZ2hpanN0dXZ3eHl6g
```

oOEhYaHiImKkpOUlZaXmJmaoqOkpaanqKmqsrO0tba3uLm6wsPExcbHyMnK0tPU1dbX2Nna4uPk5ebn6Onq8vP09fb3+Pn6/9oADAMBAAIRAxEAPwDUooqQCiigAooAK

服务器响应：

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

10.11 上传一体化模板

应用场景：

后续新增的生物识别模板将统一格式上传下载，使用数据中的Type来区分是何种生物识别模板，使用一体化格式的有：手掌模板等。

客户端请求消息：

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

注释：

```
HTTP请求方法使用：POST方法
URI使用：/iclock/cdata
HTTP协议版本使用：1.1
count=${XXX}： 一次请求里包含的一体化模板数量
Host头域：${Required}
Content-Length头域：${Required}
其他头域：${Optional}
请求实体：${DataRecord}，考勤照片数据，数据格式如下
biodata pin=${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}
    注：
pin=${XXX}：人员工号。
```

no=${XXX}：生物具体个体编号，默认值为0

【指纹】编号是：0-9，对应的手指是：左手：小拇指/无名指/中指/食指/拇指，右手：拇指/食指/中指/无名指/小拇指；

【指静脉】 和指纹相同

【面部】 都为0

【虹膜】 0为左眼 1为右眼

【手掌】 0为左手 1为右手

index=${XXX}：生物具体个体模板编号，如1个手指存储多枚模板。从0开始计算。

valid=${XXX}：是否有效标示，0：无效，1：有效，默认为1。

duress=${XXX}：是否胁迫标示，0：非胁迫，1：胁迫，默认为0。

type=${XXX}：生物识别类型

值	意义
0	通用的
1	指纹
2	面部
3	声纹
4	虹膜
5	视网膜
6	掌纹
7	指静脉
8	手掌
9	可见光面部

majorver=${XXX}：主版本号，如：指纹算法版本10.3，主版本是10，副版本是3

【指纹】9.0、10.3和12.0

【指静脉】 3.0

【面部】 5.0、7.0和8.0

【手掌】1.0

minorver=${XXX}：副版本号，如：指纹算法版本10.3，主版本是10，副版本是3

【指纹】9.0、10.3和12.0

【指静脉】 3.0

【面部】 5.0、7.0和8.0

【手掌】1.0pin=${XXX}：考勤照片名称。

format=${XXX}：模板格式，如指纹有ZK\ISO\ANSI等格式

【指纹】

值	格式
0	ZK
1	ISO
2	ANSI

【指静脉】

值	格式
0	ZK

【面部】

值 格式

0 ZK

【手掌】

值 格式

0 ZK

tmp=${XXX}：模板数据，需要对原始二进制指纹模版进行base64编码

多条记录之间使用${LF}连接。

服务器响应：

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

biodata=${XXX}
```

注释：

HTTP状态行：使用标准的HTTP协议定义

HTTP响应头域：

Content-Length头域：根据HTTP 1.1协议，一般使用该头域指定响应实体的数据长度，如果是在不确定响应实体的大小时，也支持Transfer-Encoding: chunked，Content-Length及Transfer-Encoding头域均是HTTP协议的标准定义，这里就不在详述。

响应实体：当服务器接收数据正常并处理成功时回复biodata=${XXX}，其中${XXX}表示成功处理的一体化模板数量，当出错时，回复错误描述即可。

示例：

客户端请求：

```
POST /iclock/cdata?SN=5165181600015&table=tabledata&tablename=biodata&count=9 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+cAC44IAAEAAJfKWIBWAQQFAAAAAAAAEAcHAcQAAAA3PJc4P04W/AxrTnYNXQ7cJO+ObT0vBE81KyGPkzSHuh7ul+wP647kW
```

```
NnMTDA8WMwlanRcNTpzTjc2hzUGZ488FweSBlaNOkY2GdZeNHlVXCsx81nl5MOkZ2YzJD2kMxaWrBlWM9hWXEt4VzlLNP4JzMdLF6OUY5
Kn8GPepNRjUp+OZ2BPOkd5STsfeZuXaxaKzUOWvnUk06a2L1Prlu1jzwLvYf3U6wBrBG80yLbMc+gODPL9MClyPTCNk5OFzws8xBsN

服务器响应：
HTTP/1.1 200 OK
Server Apache-Coyote/1.1 is not blacklisted
Server: Apache-Coyote/1.1
Content-Length: 9
Date: Fri, 19 Oct 2018 06:55:30 GMT

biodata=1
```

11. 下载

11.1 下载缓存命令

应用场景：

```
客户端向服务器获取已生成的命令
```

客户端请求消息：

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

注释：

```
HTTP请求方法使用：GET方法
URI使用：/iclock/getrequest
HTTP协议版本使用：1.1
```

服务器响应:

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

```
${CmdRecord}
```

注释：

```
${CmdRecord}：服务器响应实体：命令数据，数据格式如下
    C:${CmdID}:$（CmdDesc）${SP}$(CmdDetail)
CmdID： 命令id
CmdDesc: 命令描述分为数据命令、控制命令和配置命令
CmdDetail： 命令详细内容
多条记录之间使用${LF}连接
关于服务器下发命令的详细描述见【服务器命令详细说明】
```

示例：

```
客户端请求消息：
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
服务器响应:
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=192.168.213.1,NetMask=255.255.255
```

12.服务器命令详细说明

如果服务器需要对设备进行操作，需先生成命令格式，等待设备发起请求时，再将命令发送到设备，对于命令的执行结果返回已在【上传命令返回结果】中说明。客户端发起请求以及服务器响应的格式已在上文［下载缓存命令】中说明。此处重点讲解命令的格式。

12.1 DATA命令

对设备数据进行操作时使用。

12.1.1 UPDATE 子命令

增加或更新数据到设备，如果设备原来没有此主键的数据，则新增；否则修改更新原设备数据。

格式：

单条数据格式：
C:${CmdID}:DATA$(SP)UPDATE$(SP)表名$(SP)字段名1=值1$(HT)字段名2=值2$(HT)字段名3=值3
多条数据格式：
C:${CmdID}:DATA$(SP)UPDATE$(SP)表名$(SP)字段名1=值1$(HT)字段名2=值2$(HT)字段名3=值3$(LF)
字段名4=值4$(HT)字段名5=值5$(HT)字段名6=值6
（其中123是第一条数据，456是第二条数据，通过换行符$(LF)分开）

1.下发用户信息

应用场景：

服务器下发人员数据到客户端，一般用于服务器上编辑用户后自动同步用户信息到客户端

命令格式如下：

服务器下发命令：
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}
客户端上传执行结果：
ID=${CmdID}&return=${XXX}&CMD=DATA

注释：

user： 表名，指用户
CardNo： 卡号，卡号是无符号整型数据存储，软件下发的值支持两种格式下发
a、十六进制数据，格式为[%02x%02x%02x%02x]，从左到右表示第1、2、3、4个字节，如卡号为123456789，则为：Card=[15CD5B07]
b、字符串数据，如卡号为123456789，则为：Card=123456789
c、开启一人多卡功能(参考AccSupportFunList值)，这个表字段无效，使用一人多卡协议存储卡号
Pin： 用户工号
Password： 密码
Group： 用户所属组，默认属于1组
StartTime： 用户有效期开始时间：
如果设备参数DateFmtFunOn的值为1，举例：StartTime=583512660，StartTime由【附录5】算法计算出来
如果设备参数DateFmtFunOn的值为0，格式为YYYYMMDD
值为0，表示该用户无开始时间限制

EndTime： 用户有效期结束时间：

如果设备参数DateFmtFunOn的值为1，举例：EndTime=583512660，EndTime由【附录5】算法计算出来

如果设备参数DateFmtFunOn的值为0，格式为YYYYMMDD

值为0，表示该用户无结束时间限制

Name： 用户姓名，当设备为中文时，使用的是GB2312编码，其他语言时，使用UTF-8编码

Privilege： 用户权限值，值意义如下

0 普通用户

2 登记员

6 管理员

10 用户自定义

14 超级管理员

多条记录之间使用${LF}连接

示例：

软件下发一个工号为1，没有卡，密码234，用户所属组为默认，没有设置有效期，权限为普通用户的用户:

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

客户端上传执行成功结果：

ID=295&return=0&CMD=DATA

2.下发扩展用户信息

应用场景：

服务器下发扩展用户数据到客户端，一般用于服务器上编辑后自动同步扩展用户信息到客户端

命令格式如下：

服务器下发命令：

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

客户端上传执行结果：

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

注释：

extuser： 表名，指扩展用户

Pin： 用户工号

FunSwitch：功能开关，按位控制各个功能。

00000000 00000000 00000000 10001000

按位从低位到高位数：

1：表示扩展锁驱动时长功能开关

2：表示临时用户（访客）功能开关

比如：FunSwitch=1（00000001）表示打开扩展锁驱动时长功能。

FunSwitch=2（00000010）表示打开临时用户功能。

FunSwitch=3（00000011）表示打开临时用户以及扩展锁驱动时长功能。

FirstName：保留，暂未使用

LastName： 保留，暂未使用

PersonalVS： 个人验证方式

多条记录之间使用${LF}连接

示例：

```
服务器下发：
    C:500:DATA UPDATE extuser Pin=1 FunSwitch=1 FirstName=  LastName=   PersonalVS=0
客户端上传执行成功结果：
    ID=500&return=0&CMD=DATA
```

3.下发一人多卡信息

应用场景：

服务器下发一人多卡信息数据到客户端，一般用于服务器上编辑后自动同步一人多卡信息到客户端

命令格式如下：

```
服务器下发命令：
C:${CmdID}:DATA$(SP)UPDATE$(SP)mulcarduser$(SP)Pin=${XXX}$(HT)CardNo=${XXX}$(HT)LossCardFlag=${XXX}$(HT)C
ardType=${XXX}
客户端上传执行结果：
    ID=${CmdID}&return=${XXX}&CMD=DATA
```

注释：

```
mulcarduser： 表名，指扩展用户
Pin： 用户工号
CardNo： 卡号，字符串存储，软件下发的值为16进制格式字符串。比如CardNo=DF349C3BEA5277ED
LossCardFlag： 丢失卡标记，0表示正常使用卡，1表示丢失卡。
CardType： 主卡和副卡，0表示主卡，1表示副卡
多条记录之间使用${LF}连接
```

示例：

服务器下发：

C:501:DATA UPDATE mulcarduser Pin=1 CardNo=CC9932DD LossCardFlag=0 CardType=0

客户端上传执行成功结果：

ID=501&return=0&CMD=DATA

4.下发身份证信息

应用场景：

服务器下发身份证信息数据到客户端，一般用于服务器上编辑后自动同步身份证信息到客户端

命令格式如下：

服务器下发命令：

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}

客户端上传执行结果：

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

注释：

identitycard: 表名，指身份证信息

Pin:用户工号

SN_Num：身份证物理卡号

ID_Num：公民身份证号码

DN_Num：居民身份证证卡序列号（卡体管理号）

Name：姓名

Gender：姓名代码

Nation：民族代码

0,"解码错"

1,"汉"

2,"蒙古"

3,"回"

4,"藏"

5,"维吾尔"

6,"苗"

7,"彝"

8,"壮"

9,"布依"

10, "朝鲜"
11, "满"
12, "侗"
13, "瑶"
14, "白"
15, "土家"
16, "哈尼"
17, "哈萨克"
18, "傣"
19, "黎"
20, "傈僳"
21, "佤"
22, "畲"
23, "高山"
24, "拉祜"
25, "水"
26, "东乡"
27, "纳西"
28, "景颇"
29, "柯尔克孜"
30, "土"
31, "达斡尔"
32, "仫佬"
33, "羌"
34, "布朗"
35, "撒拉"
36, "毛南"
37, "仡佬"
38, "锡伯"
39, "阿昌"
40, "普米"
41, "塔吉克"
42, "怒"
43, "乌孜别克"
44, "俄罗斯"
45, "鄂温克"
46, "德昂"
47, "保安"
48, "裕固"
49, "京"
50, "塔塔尔"

51,"独龙"

52,"鄂伦春"

53,"赫哲"

54,"门巴"

55,"珞巴"

56,"基诺"

57,"编码错"

97,"其它"

98,"外国血统"

Birthday：出生日期

Valid_Info：有效期

Address：地址

Additional_Info：机读追加地址

Issuer：签发机关

Photo：照片的加密数据，并转化成base64数据内容进行传输。

PhotoJPG：照片解密数据，并转化成base64数据内容进行传输。

FP_Template1：指纹1_指纹特征数据

FP_Template2：指纹2_指纹特征数据

Reserve：保留字段

Notice：备注信息

示例：

服务器下发：

C:502:DATA UPDATE identitycard Pin=1 SN_Num=26121292460088826975 IDNum=210102199212182xxxx DNNum=
Name=张三 Gender=1 Nation=1 Birthday=19911218 ValidInfo=2013091220230912 Address=广东省东莞市塘
厦镇xxxxx AdditionalInfo= Issuer=xxx公安局 Photo=V0xmAH4AMgAA/apmyEd8
PhotoJPG=Y0tKqWNPzyEd8Pn0EhRsgAAjeWPxiUzLaPU1w=
FPTemplate1=QwGIEgELUQAAAAAAAAAAAAAAACgBmnlcAf///////0wZqfwuJK78PzJg/lw
FPTemplate2=QwGIEgEQUAAAAAAAAAAAAAAAADIBmkbyAP///////3UYCPxXQs38qEVW/rpLovwyUBv8WV8 Reserve= Notice=
客户端上传执行成功结果：

ID=502&return=0&CMD=DATA

5.下发用户门禁权限

应用场景：

服务器下发用户门禁权限到客户端，一般在服务器下发用户时跟随用户信息一起下发

命令格式如下：

服务器下发命令：

C:${CmdID}:DATA$(SP)UPDATE$(SP)userauthorize$(SP)Pin=${XXX}$(HT)AuthorizeTimezoneId=${XXX}$(HT)AuthorizeDoorId=${XXX}$(HT)DevID=${XXX}

客户端上传执行结果：

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

注释：

userauthorize： 表明，指用户门禁权限

AuthorizeTimezoneId: 用户所在时间规则的ID

AuthorizeDoorId： 用户所属门的ID

表明这个权限包含该设备哪些门，取值是通过二进制编码而得，每个门用一个二进制位表示，如果该位为1，则该门属于该权限，具体可参照如下：

1 代表LOCK1；

2 代表LOCK2；

3 代表LOCK1和LOCK2；

4 代表LOCK3；

5 代表LOCK1和LOCK3；

6 代表LOCK2和LOCK3；

7 代表LOCK1、LOCK2和LOCK3；

8 代表LOCK4；

9 代表LOCK1和LOCK4；

10 代表LOCK2和LOCK4；

11 代表LOCK1、LOCK2和LOCK4；

12 代表LOCK3和LOCK4；

13 代表LOCK1、LOCK3和LOCK4；

14 代表LOCK2、LOCK3和LOCK4；

15 代表LOCK1、LOCK2、LOCK3和LOCK4

15可以这样计算，四个门的编号分别为1,2,3,4，则：

1<<(1-1)+1<<(2-1)+1<<(3-1)+1<<(4-1)=15

DevID：设备编号，AccSupportFunList参数值，从0开始数第26位值控制。

多条记录之间使用${LF}连接

示例：

服务器下发一个工号为1，时间规则ID为1，所属门ID为1的用户：

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

客户端上传执行成功结果：

ID=296&return=0&CMD=DATA

6.下发节假日

应用场景：

服务器节假日数据到客户端，一般用于服务器上编辑后自动同步节假日到客户端

命令格式如下：

服务器下发命令：

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

客户端上传执行结果：

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

注释：

holiday： 表名，指节假日

Holiday：20100101表示2010年1月1日

HolidayType： 假日类型，只能为 1、2、3，根据需要定义其代表的意义。

Loop：1表示年循环且月日相等，2表示必须年月日相等

多条记录之间使用${LF}连接

示例：

服务器下发：

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

客户端上传执行成功结果：

ID=503&return=0&CMD=DATA

7.下发时间规则

应用场景：

服务器下发时间规则到客户端，一般用于服务器上编辑修改时间规则后自动下发

格式：

服务器下发命令：

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}

客户端上传执行结果：

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

注释：

timezone： 表明，值时间组

TimezoneId： 时间规则ID

SunTime1~SatTime3： 从周日到周六，每天三个时间段

Hol1Time1~Hol3Time3： 3个假日，每个假日有3个时间段

特别说明： 每个时间段在软件上由StartTime开始时间和EndTime结束时间组成，下发到设备前，软件按照（StartTime<<16+EndTime ）的约定转化为一个整型，例8:30 ~12:00转换后的值为(830 << 16 + 1200)，即0x33e04b0

示例：

服务器下发一个时间规则ID为2，周一时间段1为00:01~00:03，时间段2为10:00~11:00的时间规则：

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

客户端上传执行结果：

ID=307&return=0&CMD=DATA

8.下发指纹模板

应用场景：

服务器下发指纹模板到客户端，一般用于服务器下发用户时，如果用户有指纹模板就一起下发到客户端

格式：

服务器下发命令：

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

客户端上传执行结果：

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

注释：

templatev10： 表示数据类型为指纹模板，支持的指纹算法版本不超过10.0

Pin： 用户工号

FingerID： 手指编号，取值为0到9

Valid： 描述模版有效性及胁迫标示，值意义如下：

值	描述
0	无效模版
1	正常模版

3　　胁迫模版

Template：传输指纹模版时，需要对原始二进制指纹模版进行base64编码

多条记录之间使用${LF}连接

示例：

服务器下发一个所属用户工号为1，指纹id为6的普通指纹：

C:333:DATA UPDATE templatev10 Pin=1 FingerID=6 Valid=1

Template=Si9TUzIxAAADbG8ECAUHCc7QAAAbbWkBAAAAg5Eac2wNAG4PmwDZAAVjrQAiAIIPWwAzbAIPsgCOAMgOWmyOAFgPwwBWAI5i2wCUAJsPTACUbFOPrwCqAKIPgGy8AFAPwwAHAINjZADMANQPPADdbKgPjgDhAIAOoGzmADAOIQApADJiqwD5ALIOYwAEbasM/wAJAWcPn2wVAYoNhADfAfJhbQAIAdYPCgAkbZOPbAA2ASAPOGw4AZQPI2jqbyCDiIW1BQuYSH1i7Y4XXYE5/d6Xgu24q3p2M/ymhOhngIH19IIorBJt/II0bg4eU//33jP73D7onn2jDvmapxjD9F9QX4XA/ManEgtvK04TZ3tIFIqVxQBEgd6C+e9BAGWC3AiC6WSCofgxIEd4mZdKCj8T0eComsmOhGr9H+4NKPCai55NHQmD9bIWXnx79YNfpfkK94RjfiCZAiAvxAIRcMoEAHYBdLkEA88FhpMEAKvNAzBmAaQLgIhitQQDAhFtXAYAn9z9/icPANEZjMAEmcOuZ3sKAJcdsn53AQQAsCIJ/48LA8QkfcLAwmavBwPOLf3+/cFGzQCZXHuDwHIHAGcyAJP/UxMA91pbw8MUw//EaVzCOsMQbPpmmmnBwUHCwq3Bb2wVAQe4oHfohcT/eMHBtREDMYrgRjExK44QA8KPfcPF/8IEwPyvwcBg/xcBwoykG32MwYNneK8HAzmRXsJrwRTF2JPwkMLEwv/CQcDBk8HB/sEIAJ+TVAtZBwDHIxz9VQZshplgfhkBwpyorsHBfpPCg6pna6sKAKqqcMUGwVVQCwCCwFDABFtPYAG/wJPJxwfDwqjAwOcJAKsD/fiXWFQZAQe+dIdq+sPDgMDCwLvAWmUBr8sQInYHCwPEOUPF/8N2BXsJbKzQHP3AwjrBaGUBrtktfMCuGwJj26lSwUWXB8LArXZvZFkYADDcpAjBw8LCw8UGwv2uecH+wEQJxY3IKsD9KXYPAFHnw0b99/z//sEFwFBwAAvpsG3CBcDBrKHDw3rAizrAWJMGAJLvPcUAVRI9DQinZMDABcDBrsPExsOJUDv/wV4EEKAXF8GOBBO/Khf/xAkQqjLhkvz7/sBmBdWTS+zCVwgQaUoi//+S/OQNEONMSIfCrKDBaVJCAM5DAmwBCOVS

客户端返回执行结果：

ID=333&return=0&CMD=DATA

9.下发首卡开门信息

应用场景：

服务器首卡开门数据到客户端，一般用于服务器上编辑后自动同步首卡开门数据到客户端

命令格式如下：

服务器下发命令：

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

客户端上传执行结果：

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

注释：

firstcard：　表名，指首卡开门表

Pin：用户工号

DoorID：门ID

TimezoneID： 时间段编号

DevID：设备编号，AccSupportFunList参数值，从0开始数第26位值控制。

多条记录之间使用${LF}连接

示例：

服务器下发：

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

客户端上传执行成功结果：

ID=504&return=0&CMD=DATA

10.下发多卡开门组合信息

应用场景：

服务器多卡开门组合数据到客户端，一般用于服务器上编辑后自动同步多卡开门组合数据到客户端

命令格式如下：

服务器下发命令：

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}

客户端上传执行结果：

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

注释：

multimcard： 表名，指首卡开门表

Index：索引

DoorId：门ID

Group1： 组编号1

Group2： 组编号2

Group3：组编号3

Group4： 组编号4

Group5： 组编号5

DevID：设备编号，AccSupportFunList参数值，从0开始数第26位值控制。

多条记录之间使用${LF}连接

示例：

服务器下发：

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

客户端上传执行成功结果：

ID=505&return=0&CMD=DATA

11.下发联动详细信息

应用场景：

服务器联动详细信息数据到客户端，一般用于服务器上编辑后自动同步联动详细信息数据到客户端

命令格式如下：

服务器下发命令：

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}

客户端上传执行结果：

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

注释：

inoutfun： 表名，指联动详细信息表

Index：索引

EventType：触发事件

InAddr： 触发事件位置 即门ID或辅佐输入的ID 1~4 0为任意位置

OutType： 输出类型 0为门锁，1为辅助输出

OutAddr：输出动作位置 门锁为1~4, 辅助输出为1~4

OutTime： 输出动作时间 0关闭，1~254打开n秒，255常开

Reserved： 保留 以保证对齐,低四位用来表示联动类型（0：门联动，1：读头联动，2：IPC联动），高四位用来表示操作类型中的锁定、解锁动作（16：门锁定，32：门解锁）

InDevID：设备编号，AccSupportFunList参数值，从0开始数第26位值控制。

OutDevID：设备编号，AccSupportFunList参数值，从0开始数第26位值控制。

多条记录之间使用${LF}连接

示例：

服务器下发：

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

客户端上传执行成功结果：

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

12.下发定时输出信息

应用场景：

```
服务器定时输出信息数据到客户端，一般用于服务器上编辑后自动同步定时输出信息数据到客户端
```

命令格式如下：

```
服务器下发命令：
C:${CmdID}:DATA$(SP)UPDATE$(SP)outrelaysetting$(SP)Num=${XXX}$(HT)OutType=${XXX}$(HT)ActionType=${XXX}$(HT)TimezoneId=${XXX}$(HT)DevID=${XXX}
客户端上传执行结果：
    ID=${CmdID}&return=${XXX}&CMD=DATA
```

注释：

```
outrelaysetting:  表名，指定时输出表
Num：输出点,辅助输出属性表的主键编号
OutType：0表示门，1表示辅助输出
ActionType：表示0无、2常闭、1常开
TimezoneId:  为timezone的编号
DevID：设备编号，AccSupportFunList参数值，从0开始数第26位值控制。
多条记录之间使用${LF}连接
```

示例：

```
服务器下发：
    C:507:DATA UPDATE outrelaysetting Num=1 OutType=1   ActionType=0    TimezoneId=1    DevID=14
客户端上传执行成功结果：
    ID=507&return=0&CMD=DATA
```

13.下发夏令时信息

应用场景：

```
服务器夏令时信息数据到客户端，一般用于服务器上编辑后自动同步夏令时信息数据到客户端
```

命令格式如下：

服务器下发命令：

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

客户端上传执行结果：

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

注释：

DSTSetting： 表名，指夏令时表

Year：年

StartTime：开始时间(格式：MMIIWWHH，分别的含义：MM-几月，II-第几个星期，WW-星期几，HH-几时)

EndTime：结束时间(格式：MMIIWWHH，分别的含义：MM-几月，II-第几个星期，WW-星期几，HH-几时)

Loop：是否循环

多条记录之间使用${LF}连接

示例：

服务器下发：

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

客户端上传执行成功结果：

ID=508&return=0&CMD=DATA

14.下发设备属性信息

应用场景：

服务器设备属性信息数据到客户端，一般用于服务器上编辑后自动同步设备属性信息数据到客户端

命令格式如下：

服务器下发命令：

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}

客户端上传执行结果：

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

注释：

DevProperty： 表名，指设备属性表

ID：设备编号

Type：设备类型(0:Door Unit,1:Zigbee锁,2:主机BioCom48)，目前都是0

BusType：总线类型(0:485A总线,1:485B总线,2:ZIGBEE总线,3:TCP/IP)

MachineType：机器类型,见【附录7】

Address：设备地址

Mac：设备MAC地址

IPAddress:设备IP地址

SN：设备序列号

IsMaster：是否主机，0：主机，1：从机

多条记录之间使用${LF}连接

示例：

服务器下发：

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

客户端上传执行成功结果：

ID=509&return=0&CMD=DATA

15.下发设备参数信息

应用场景：

服务器设备参数信息数据到客户端，一般用于服务器上编辑后自动同步设备参数信息数据到客户端

命令格式如下：

服务器下发命令：

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

客户端上传执行结果：

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

注释：

DevParameters： 表名，指设备参数表

ID：设备编号

Name：参数名，见【附录12】

Value：参数值

多条记录之间使用${LF}连接

示例：

服务器下发：

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

客户端上传执行成功结果：

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

16.下发门属性信息

应用场景：

```
服务器门属性信息数据到客户端，一般用于服务器上编辑后自动同步门属性信息数据到客户端
```

命令格式如下：

```
服务器下发命令：
C:${CmdID}:DATA$(SP)UPDATE$(SP)DoorProperty$(SP)ID=${XXX}$(HT)DevID=${XXX}$(HT)Address=${XXX}$(HT)Disable
=${XXX}
客户端上传执行结果：
    ID=${CmdID}&return=${XXX}&CMD=DATA
```

注释：

```
DoorProperty:  表名，指门属性表
ID：门ID
DevID：从机编号
Address：锁继电器 1：LOCK1 2：LOCK2 3：LOCK3 4：LOCK4
Disable:启用或禁用：0：启用 1：禁用
多条记录之间使用${LF}连接
```

示例：

```
服务器下发：
    C:511:DATA UPDATE DoorProperty ID=1 DevID=14    Address=1    Disable=0
客户端上传执行成功结果：
    ID=511&return=0&CMD=DATA
```

17.下发门参数信息

应用场景：

```
服务器门参数信息数据到客户端，一般用于服务器上编辑后自动同步门参数信息数据到客户端
```

命令格式如下：

服务器下发命令：

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

客户端上传执行结果：

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

注释：

DoorParameters： 表名，指门参数表

ID：门ID

Name：参数名，见【附录10】

Value：参数值

DevID:设备属性编号

多条记录之间使用${LF}连接

示例：

服务器下发：

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

客户端上传执行成功结果：

ID=512&return=0&CMD=DATA

18.下发读头属性信息

应用场景：

服务器读头属性信息数据到客户端，一般用于服务器上编辑后自动同步读头属性信息数据到客户端

命令格式如下：

服务器下发命令：

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}

客户端上传执行结果：

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

注释：

ReaderProperty： 表名，指读头属性表

ID：读头编号

DoorID：所属门编号

Type：读头类型，二进制位计算（0：表示不使用，1：表示485读头，2：表示韦根读头，3：表示485读头或者韦根读头，4：表示网络读头，8：表示zigbee读头）

Address:读头地址,针对韦根和485读头地址

IPAddress：读头IP地址

Port：读头端口

MAC：读头MAC地址

InOut:出入读头，0：入，1：出

Disable：启用或禁用：0：启用 1：禁用

VerifyType：读头验证方式

Multicast：组播地址

DevID:所属设备编号

OfflineRefuse：是否离线通行，0：正常通行，1：拒绝通行

多条记录之间使用${LF}连接

示例：

服务器下发：

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

客户端上传执行成功结果：

ID=513&return=0&CMD=DATA

19.下发读头参数信息

应用场景：

服务器读头参数信息数据到客户端，一般用于服务器上编辑后自动同步读头参数信息数据到客户端

命令格式如下：

服务器下发命令：

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

客户端上传执行结果：

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

注释：

ReaderParameters： 表名，指读头参数表

ID：读头编号

DevID：所属设备编号

Name：参数名，见【附录11】

Value：参数值

多条记录之间使用${LF}连接

示例：

服务器下发：

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

客户端上传执行成功结果：

ID=514&return=0&CMD=DATA

20.下发辅助输入属性信息

应用场景：

服务器辅助输入属性信息数据到客户端，一般用于服务器上编辑后自动同步辅助输入属性信息数据到客户端

命令格式如下：

服务器下发命令：

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

客户端上传执行结果：

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

注释：

AuxIn： 表名，指辅助输入属性表

ID：辅助输入ID

DevID：所属设备编号

Address：辅助输入 1：AuxIn1 2：AuxIn2

Disable：启用或禁用：0：启用 1：禁用

多条记录之间使用${LF}连接

示例：

服务器下发：

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

客户端上传执行成功结果：

ID=515&return=0&CMD=DATA

21.下发辅助输出属性信息

应用场景：

服务器辅助输出属性信息数据到客户端，一般用于服务器上编辑后自动同步辅助输出属性信息数据到客户端

命令格式如下：

服务器下发命令：

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

客户端上传执行结果：

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

注释：

AuxOut： 表名，指辅助输出属性表

ID：辅助输出ID

DevID：所属设备编号

Address：辅助输出继电器 1：AuxOut1

Disable：启用或禁用：0：启用 1：禁用

多条记录之间使用${LF}连接

示例：

服务器下发：

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

客户端上传执行成功结果：

ID=516&return=0&CMD=DATA

22.下发默认韦根格式信息

应用场景：

服务器默认韦根格式信息数据到客户端，一般用于服务器上编辑后自动同步默认韦根格式信息数据到客户端

命令格式如下：

服务器下发命令：

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

客户端上传执行结果：

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

注释：

DefaultWGFormat： 表名，指默认韦根格式表

ID：索引

CardBit：位数

SiteCode：区位码

FormatName：韦根自定义名称

CardFormat：韦根格式

多条记录之间使用${LF}连接

示例：

服务器下发：

C:517:DATA UPDATE DefaultWGFormat ID=2 CardBit=26 SiteCode=0 FormatName=韦根26 CardFormat=pccccccccccccccccccccccp:eeeeeeeeeeeeeooooooooooooo

客户端上传执行成功结果：

ID=517&return=0&CMD=DATA

23.下发韦根格式信息

应用场景：

服务器韦根格式信息数据到客户端，一般用于服务器上编辑后自动同步韦根格式信息数据到客户端

命令格式如下：

服务器下发命令：

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

客户端上传执行结果：

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

注释：

WGFormat： 表名，指韦根格式表

ID：索引

CardBit：位数

SiteCode：区位码

FormatName：韦根自定义名称

CardFormat：韦根格式

多条记录之间使用${LF}连接

示例：

服务器下发：

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

客户端上传执行成功结果：

ID=518&return=0&CMD=DATA

24.下发韦根读头对应的韦根格式信息

应用场景：

服务器韦根读头对应的韦根格式信息数据到客户端，一般用于服务器上编辑后自动同步韦根读头对应的韦根格式信息数据到客户端

命令格式如下：

服务器下发命令：

C:${CmdID}:DATA$(SP)UPDATE$(SP)ReaderWGFormat$(SP)ReaderID=${XXX}$(HT)DevID=${XXX}$(HT)WGFormatID=${XXX}$(HT)ParityVerifyDisable=${XXX}$(HT)ReversalType=${XXX}

客户端上传执行结果：

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

注释：

ReaderWGFormat： 表名，指韦根读头对应的韦根格式表

ReaderID：读头编号

DevID：设备编号，AccSupportFunList参数值，从0开始数第26位值控制。

WGFormatID：对应韦根格式表索引

ParityVerifyDisable：奇偶校验禁用，1：禁用，0：启用

ReversalType：1:整个Bit位翻转,2:0和1取反,3:高低字节交换位置,4:单个字节高低位交换位置

多条记录之间使用${LF}连接

示例：

服务器下发：

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

客户端上传执行成功结果：

ID=519&return=0&CMD=DATA

25.下发输入控制（受时间段限制）信息

应用场景：

服务器输入控制（受时间段限制）信息数据到客户端，一般用于服务器上编辑后自动同步输入控制（受时间段限制）信息数据到客户端

命令格式如下：

服务器下发命令：

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

客户端上传执行结果：

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

注释：

InputIOSetting： 表名，指输入控制（受时间段限制）表

Number：输入编号，门属性表中的门编号或辅助输入属性表中的编号

InType：0：出门按钮，1：辅助输入

TimeZoneId：timezone编号

DevID：设备编号，AccSupportFunList参数值，从0开始数第26位值控制。

多条记录之间使用${LF}连接

示例：

服务器下发：

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

客户端上传执行成功结果：

ID=520&return=0&CMD=DATA

26.下发不同时段的验证方式信息

应用场景：

服务器不同时段的验证方式信息数据到客户端，一般用于服务器上编辑后自动同步不同时段的验证方式信息数据到客户端

命令格式如下：

服务器下发命令：

C:${CmdID}:DATA$(SP)UPDATE$(SP)DiffTimezoneVS$(SP)TimezoneID=${XXX}$(HT)SunTime1=${XXX}$(HT)SunTime1VSUser=${XXX}$(HT)SunTime1VSDoor=${XXX}$(HT)SunTime2=${XXX}$(HT)SunTime2VSUser=${XXX}$(HT)SunTime2VSDoor=${XXX}$(HT)SunTime3=${XXX}$(HT)SunTime3VSUser=${XXX}$(HT)SunTime3VSDoor=${XXX}$(HT)MonTime1=${XXX}$(HT)MonTime1VSUser=${XXX}$(HT)MonTime1VSDoor=${XXX}$(HT)MonTime2=${XXX}$(HT)MonTime2VSUser=${XXX}$(HT)MonTime2VSDoor=${XXX}$(HT)MonTime3=${XXX}$(HT)MonTime3VSUser=${XXX}$(HT)MonTime3VSDoor=${XXX}$(HT)TueTime1=${XXX}$(HT)TueTime1VSUser=${XXX}$(HT)TueTime1VSDoor=${XXX}$(HT)TueTime2=${XXX}$(HT)TueTime2VSUser=${XXX}$(HT)TueTime2V

SDoor=${XXX}$(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=${XXX}$(HT)WedTime3VSUser=${XXX}$(HT)WedTime3VSDoor=${XXX}$(HT)ThuTime1=${XXX}$(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=${XXX}$(HT)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)SatTime3VSUser=${XXX}$(HT)SatTime3VSDoor=${XXX}$(HT)Hol1Time1=${XXX}$(HT)Hol1Time1VSUser=${XXX}$(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=${XXX}$(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)Hol3Time2=${XXX}$(HT)Hol3Time2VSUser=${XXX}$(HT)Hol3Time2VSDoor=${XXX}$(HT)Hol3Time3=${XXX}$(HT)Hol3Time3VSUser=${XXX}$(HT)Hol3Time3VSDoor=${XXX}

客户端上传执行结果：

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

注释：

DiffTimezoneVS： 表名，指不同时段的验证方式表

TimezoneID：索引号

SunTime1~SatTime3： 从周日到周六，每天三个时间段

Hol1Time1~Hol3Time3： 3个假日，每个假日有3个时间段

SunTime1VSUser~SatTime3VSUser： 从周日到周六，每天三个时间段，人不同时间段验证方式

Hol1Time1VSUser~Hol3Time3VSUser： 3个假日，每个假日有3个时间段，人不同时间段验证方式

SunTime1VSDoor~SatTime3VSDoor： 从周日到周六，每天三个时间段，门不同时间段验证方式

Hol1Time1VSDoor~Hol3Time3VSDoor： 3个假日，每个假日有3个时间段，门不同时间段验证方式

特别说明： 每个时间段在软件上由StartTime开始时间和EndTime结束时间组成，下发到设备前，软件按照（StartTime<<16+EndTime ）的约定转化为一个整型，例8:30 ~12:00转换后的值为(830 << 16 + 1200)，即0x33e04b0

多条记录之间使用${LF}连接

示例：

服务器下发：

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
客户端上传执行成功结果：

ID=521&return=0&CMD=DATA

27.下发门不同的时间段验证方式信息

应用场景：

服务器门不同的时间段验证方式信息数据到客户端，一般用于服务器上编辑后自动同步门不同的时间段验证方式信息数据到客户端

命令格式如下：

服务器下发命令：

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

客户端上传执行结果：

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

注释：

DoorVSTimezone： 表名，指门不同的时间段验证方式表

DoorID：门编号

TimezoneID：对应DiffTimezoneVS

DevID：设备编号，AccSupportFunList参数值，从0开始数第26位值控制。

多条记录之间使用${LF}连接

示例：

服务器下发：

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

客户端上传执行成功结果：

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

28.下发人不同的时间段验证方式信息

应用场景：

服务器人不同的时间段验证方式信息数据到客户端，一般用于服务器上编辑后自动同步人不同的时间段验证方式信息数据到客户端

命令格式如下：

```
服务器下发命令：
C:${CmdID}:DATA$(SP)UPDATE$(SP)PersonalVSTimezone$(SP)PIN=${XXX}$(HT)DoorID=${XXX}$(HT)TimezoneID=${XXX}$(HT)DevID=${XXX}
客户端上传执行结果：
    ID=${CmdID}&return=${XXX}&CMD=DATA
```

注释：

```
PersonalVSTimezone：  表名，指人不同的时间段验证方式表
PIN：用户的工号
DoorID：门编号
TimezoneID：对应DiffTimezoneVS
DevID：设备编号，AccSupportFunList参数值，从0开始数第26位值控制。
多条记录之间使用${LF}连接
```

示例：

```
服务器下发：
    C:523:DATA UPDATE PersonalVSTimezone PIN=1  DoorID=4    TimezoneID=1    DevID=14
客户端上传执行成功结果：
    ID=523&return=0&CMD=DATA
```

29.下发超级用户权限信息

应用场景：

服务器超级用户权限信息数据到客户端，一般用于服务器上编辑后自动同步超级用户权限信息数据到客户端

命令格式如下：

```
服务器下发命令：
```

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

客户端上传执行结果：

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

注释：

SuperAuthorize： 表名，指超级用户权限表

Pin：用户的工号

DoorID：门编号

DevID：设备编号，AccSupportFunList参数值，从0开始数第26位值控制。

多条记录之间使用${LF}连接

示例：

服务器下发：

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

客户端上传执行成功结果：

ID=524&return=0&CMD=DATA

30.下发比对照片

应用场景：

服务器下发比对照片

命令格式如下：

服务器下发命令：

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}

客户端上传执行结果：

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

注释：

PIN=${XXX}：用户工号

Type=${XXX}：生物特征类型：

值 意义

0 通用的

1 指纹

2 面部（近红外）

3 声纹

4 虹膜

5 视网膜

6 掌纹

7 指静脉

8 手掌

9 可见光人脸

Size=${XXX}：生物特征图片二进制数据经过base64 编码之后的长度。

Content=${XXX}：传输生物特征图片时，需要对原始二进制生物特征图片进行base64 编码。

Url=${XXX}：服务器文件存放地址，目前只支持 jpg 格式。

Format=${XXX}：下发方式，0：base64方式，1：url方式。

多条记录之间使用${LF}连接。

注：

Url是相对路径的场合，直接发相对路径即可。

31.下发用户照片

应用场景：

服务器上下发用户照片

命令格式如下：

服务器下发命令：

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

客户端上传执行结果：

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

注释：

pin=${XXX}：用户工号。

size=${XXX}：用户照片二进制数据经过base64 编码之后的长度。

content=${XXX}：传输用户照片时，需要对原始二进制生物特征图片进行base64 编码。

多条记录之间使用${LF}连接。

示例：

服务器下发：

C:524:DATA UPDATE userpic pin=2 size=138620

content=/9j/4AAQSkZJRgABAQAAAQABAAD/2wBDAAEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQ

EBAQEBAQEBAQEBAQEBAQEBAQH/2wBDAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQE

BAQ

EBAQEBAQEBAQH/wAARCAGQAoADASIAAhEBAxEB/8QAHwAAAQUBAQEBAQ.....

客户端上传执行成功结果：

ID=524&return=0&CMD=DATA

12.1.2 DELETE 子命令

应用场景：

用于在软件上控制删除设备中的数据，如用户、指纹模板等

格式：

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

注释：

$(Cond):删除条件列表,格式如下：

条件字段名1=值1$(HT)条件字段名2=值2$(HT)条件字段名3=值3

删除条件列表为*时，表示清空表；当删除条件列表有多个时，多个条件为“且“的关系

多个条件使用$(LF)隔开。

1.删除用户信息

应用场景：

常用于删除某个用户或删除全部用户

格式：

服务器下发命令：

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

客户端上传执行结果：

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

注释：

user： 表名，指用户信息

$(Cond)： 常用“Pin=x”表示删除工号为x的用户，或用“*”表示删除全部用户

特别说明:

一般服务器删除用户时，可以把对应的验证方式、门禁权限等一并删除，因此服务器下发删除用户信息的命令时，往往需要伴随着删除门禁权限、删除对应验证方式（如指纹）的命令。

示例：

删除工号1的用户（可以把包含用户相关的数据一并删除）：

C:335:DATA DELETE userauthorize Pin=1

C:336:DATA DELETE templatev10 Pin=1

C:337:DATA DELETE user Pin=1

（其中1、2命令分别为删除门禁权限和删除指纹模板的命令，删除用户的命令排在最后）

客户端返回执行结果：

ID=335&return=0&CMD=DATA

ID=336&return=0&CMD=DATA

ID=337&return=0&CMD=DATA

2.删除扩展用户信息

应用场景：

常用于删除某个扩展用户信息或删除全部扩展用户信息

格式：

服务器下发命令：

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

客户端上传执行结果：

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

注释：

extuser： 表名，指扩展用户信息

$(Cond)：常用“Pin=x”表示删除工号为x的扩展用户信息，或用“*”表示删除全部扩展用户信息

示例：

删除工号1的扩展用户信息：

C:337:DATA DELETE extuser Pin=1

客户端返回执行结果：

ID=337&return=0&CMD=DATA

3.删除一人多卡数据

应用场景：

常用于删除某个一人多卡数据或 删除全部一人多卡数据

格式：

服务器下发命令：

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

客户端上传执行结果：

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

注释：

mulcarduser： 表名，指一人多卡数据

$(Cond)：常用“Pin=x”表示删除工号为x的一人多卡数据，或用“*”表示删除全部一人多卡数据

示例：

删除工号1的一人多卡数据：

C:337:DATA DELETE mulcarduser Pin=1

客户端返回执行结果：

ID=337&return=0&CMD=DATA

4.删除用户门禁权限

应用场景：

常用于服务器删除一个人员后，将该用户对应的门禁权限一起删除。

格式：

服务器下发命令：

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

客户端上传执行结果：

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

注释：

userauthorize： 表名，指用户门禁权限

$(Cond)：常用Pin=x表示删除工号为x的的用户门禁权限，或用“*”表示删除全部用户门禁权限

示例：

删除工号为1的用户的门禁权限

C:335:DATA DELETE userauthorize Pin=1

客户端上传执行结果：

ID=335&return=0&CMD=DATA

5.删除门禁记录

应用场景：

常用于清除设备全部门禁记录。

格式：

服务器下发命令：

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

客户端上传执行结果：

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

注释：

transaction： 表名，指门禁记录

$(Cond)：常用 *表示产出全部记录

示例：

删除全部门禁记录：

C:123:DATA DELETE transaction *

客户端上传执行结果：

ID=123&return=0&CMD=DATA

6.删除指纹模板

应用场景：

删除设备的指纹模板。一般用于服务器单独删除设备某个用户的指定一个指纹模板或服务器删除一个用户时联通其指纹模板一起删除。

格式：

服务器下发命令：

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

客户端上传执行结果：

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

注释：

```
templatev10： 表名，指用户指纹模板
$(Cond)：常用Pin=x表示删除工号为x的用户的所有指纹模板，可组合FingerID=x删除指定的某个指纹模板
          或用“*”表示删除全部用户
```

示例：

```
删除工号1的用户指纹id为6的指纹模板：
    C:342:DATA DELETE templatev10 Pin=1 FingerID=6
客户端上传执行结果：
    ID=342&return=0&CMD=DATA
```

7.删除节假日

应用场景：

```
常用于清除设备全部节假日。
```

格式：

```
服务器下发命令：
    C:(CmdID):DATA$(SP)DELETE$(SP)holiday$(SP)$(Cond)
客户端上传执行结果：
    ID=${CmdID}&return=${XXX}&CMD=DATA
```

注释：

```
holiday： 表名，指节假日
$(Cond)：常用 *表示产出全部节假日
```

示例：

```
删除全部节假日：
    C:123:DATA DELETE holiday *
客户端上传执行结果：
    ID=123&return=0&CMD=DATA
```

8.删除时间段

应用场景：

常用于删除某个时间段数据或 删除全部时间段数据

格式：

服务器下发命令：

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

客户端上传执行结果：

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

注释：

holiday： 表名，指时间段

$(Cond)： 常用Timezoneld=x表示删除Timezoneld为x的时间段

或用“*”表示删除全部时间段

示例：

删除时间段ID为1的时间段：

C:342:DATA DELETE timezone Timezoneld=1

客户端上传执行结果：

ID=342&return=0&CMD=DATA

9.删除首卡开门

应用场景：

常用于删除某个首卡开门数据或 删除全部首卡开门数据

格式：

服务器下发命令：

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

客户端上传执行结果：

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

注释：

firstcard： 表名，指首卡开门

$(Cond)： DoorID=x表示删除DoorID为x的首卡开门数据

Pin=x表示删除Pin为x的首卡开门数据

用“*”表示删除全部首卡数据

示例：

删除DoorID为1或者Pin为1的首卡数据：

C:342:DATA DELETE firstcard DoorID=1

C:343:DATA DELETE firstcard Pin=1

客户端上传执行结果：

ID=342&return=0&CMD=DATA

ID=343&return=0&CMD=DATA

10.删除多卡开门

应用场景：

常用于删除某个多卡开门数据或 删除全部多卡开门数据

格式：

服务器下发命令：

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

客户端上传执行结果：

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

注释：

multimcard： 表名，指多卡开门表

$(Cond)： Index=x表示删除Index为x的多卡开门数据

用“*”表示删除全部多卡开门数据

示例：

删除Index为1的多卡开门数据：

C:342:DATA DELETE multimcard Index=1

客户端上传执行结果：

ID=342&return=0&CMD=DATA

11.删除联动详细信息

应用场景：

常用于删除某个联动详细信息数据或 删除全部联动详细信息数据

格式：

服务器下发命令：

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

客户端上传执行结果：

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

注释：

inoutfun： 表名，指联动详细信息表

$(Cond)： Index=x表示删除Index为x的多卡开门数据

用“*”表示删除全部多卡开门数据

示例：

删除Index为1的联动详细信息数据：

C:342:DATA DELETE inoutfun Index=1

客户端上传执行结果：

ID=342&return=0&CMD=DATA

12.删除定时输出

应用场景：

常用于删除某个定时输出数据或 删除全部定时输出数据

格式：

服务器下发命令：

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

客户端上传执行结果：

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

注释：

outrelaysetting： 表名，指定时输出表

$(Cond)： Num=x表示删除Num为x的定时输出数据

用“*”表示删除全部定时输出数据

示例：

删除全部定时输出数据：

C:342:DATA DELETE outrelaysetting Num=1

客户端上传执行结果：

ID=342&return=0&CMD=DATA

13.删除夏令时

应用场景：

常用于删除全部夏令时数据

格式：

服务器下发命令：

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

客户端上传执行结果：

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

注释：

DSTSetting： 表名，指夏令时表

$(Cond)：用“*”表示删除全部夏令时数据

示例：

删除Num为1的定时输出数据：

C:342:DATA DELETE DSTSetting *

客户端上传执行结果：

ID=342&return=0&CMD=DATA

14.删除设备属性

应用场景：

常用于删除全部设备属性数据

格式：

服务器下发命令：

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

客户端上传执行结果：

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

注释：

DevProperty： 表名，指设备属性表

$(Cond)：用“*”表示删除全部设备属性数据

示例：

删除全部设备属性数据：

C:342:DATA DELETE DevProperty *

客户端上传执行结果：

ID=342&return=0&CMD=DATA

15.删除设备参数

应用场景：

常用于删除全部设备参数数据

格式：

服务器下发命令：

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

客户端上传执行结果：

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

注释：

DevParameters： 表名，指设备参数表

$(Cond)：用“*”表示删除全部设备参数数据

示例：

删除全部设备参数数据：

C:342:DATA DELETE DevParameters *

客户端上传执行结果：

ID=342&return=0&CMD=DATA

16.删除门属性

应用场景：

常用于删除全部门属性数据

格式：

服务器下发命令：

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

客户端上传执行结果：

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

注释：

DoorProperty： 表名，指门属性表

$(Cond)：用“*”表示删除全部门属性数据

示例：

删除全部门属性数据：

C:342:DATA DELETE DoorProperty *

客户端上传执行结果：

ID=342&return=0&CMD=DATA

17.删除门参数

应用场景：

常用于删除全部门参数数据

格式：

服务器下发命令：

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

客户端上传执行结果：

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

注释：

DoorParameters： 表名，指门参数表

$(Cond)：用“*”表示删除全部门参数数据

示例：

删除全部门参数数据：

C:342:DATA DELETE DoorParameters *

客户端上传执行结果：

ID=342&return=0&CMD=DATA

18.删除读头属性

应用场景：

常用于删除全部读头属性数据

格式：

服务器下发命令：

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

客户端上传执行结果：

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

注释：

ReaderProperty： 表名，指读头属性表

$(Cond)：用“*”表示删除全部读头属性数据

示例：

删除全部读头属性数据：

C:342:DATA DELETE ReaderProperty *

客户端上传执行结果：

ID=342&return=0&CMD=DATA

19.删除读头参数

应用场景：

常用于删除全部读头参数数据

格式：

服务器下发命令：

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

客户端上传执行结果：

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

注释：

```
ReaderParameters： 表名，指读头参数表
$(Cond)：用“*”表示删除全部读头参数数据
```

示例：

```
删除全部读头参数数据：
    C:342:DATA DELETE ReaderParameters *
客户端上传执行结果：
    ID=342&return=0&CMD=DATA
```

20.删除辅助输入

应用场景：

```
常用于删除全部辅助输入数据
```

格式：

```
服务器下发命令：
    C:${CmdID}:DATA$(SP)DELETE$(SP)AuxIn$(SP)$(Cond)
客户端上传执行结果：
    ID=${CmdID}&return=${XXX}&CMD=DATA
```

注释：

```
AuxIn： 表名，指辅助输入表
$(Cond)：用“*”表示删除全部辅助输入数据
```

示例：

```
删除全部辅助输入数据：
    C:342:DATA DELETE AuxIn *
客户端上传执行结果：
    ID=342&return=0&CMD=DATA
```

21.删除辅助输出

应用场景：

常用于删除全部辅助输出数据

格式：

服务器下发命令：

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

客户端上传执行结果：

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

注释：

AuxOut： 表名，指辅助输出表

$(Cond)： 用“*”表示删除全部辅助输出数据

示例：

删除全部辅助输出数据：

C:342:DATA DELETE AuxOut *

客户端上传执行结果：

ID=342&return=0&CMD=DATA

22.删除默认韦根数据

应用场景：

常用于常用于删除某个默认韦根数据或 删除全部默认韦根数据

格式：

服务器下发命令：

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

客户端上传执行结果：

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

注释：

DefaultWGFormat： 表名，指默认韦根表

$(Cond)： ID=x表示删除ID为x的默认韦根数据

CardBit=x表示删除CardBit为x的默认韦根数据

用“*”表示删除全部默认韦根数据

示例：

删除ID=1或者CardBit=26的韦根数据：

C:342:DATA DELETE DefaultWGFormat ID=1

C:343:DATA DELETE DefaultWGFormat CardBit=26

客户端上传执行结果：

ID=342&return=0&CMD=DATA

ID=343&return=0&CMD=DATA

23.删除韦根数据

应用场景：

常用于常用于删除某个韦根数据或 删除全部韦根数据

格式：

服务器下发命令：

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

客户端上传执行结果：

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

注释：

WGFormat： 表名，指韦根表

$(Cond)： ID=x表示删除ID为x的韦根数据

CardBit=x表示删除CardBit为x的韦根数据

用“*”表示删除全部韦根数据

示例：

删除ID=1或者CardBit=26的韦根数据：

C:342:DATA DELETE WGFormat ID=1

C:343:DATA DELETE WGFormat CardBit=26

客户端上传执行结果：

ID=342&return=0&CMD=DATA

ID=343&return=0&CMD=DATA

24.删除读头韦根数据

应用场景：

常用于常用于删除某个读头韦根数据或 删除全部读头韦根数据

格式：

服务器下发命令：

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

客户端上传执行结果：

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

注释：

WGFormat： 表名，指读头韦根表

$(Cond)： ReaderID=x表示删除ReaderID为x的读头韦根数据

DevID=x表示删除DevID为x的读头韦根数据

用“*”表示删除全部读头韦根数据

示例：

删除ReaderID=1或者DevID=24的读头韦根数据：

C:342:DATA DELETE WGFormat ReaderID=1

C:343:DATA DELETE WGFormat DevID=24

客户端上传执行结果：

ID=342&return=0&CMD=DATA

ID=343&return=0&CMD=DATA

25.删除输入控制（受时间段限制）

应用场景：

常用于删除全部输入控制（受时间段限制）数据

格式：

服务器下发命令：

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

客户端上传执行结果：

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

注释：

InputIOSetting： 表名，指输入控制（受时间段限制）表

$(Cond)：用“*”表示删除全部输入控制（受时间段限制）数据

示例：

删除全部输入控制（受时间段限制）数据：

C:342:DATA DELETE InputIOSetting *

客户端上传执行结果：

ID=342&return=0&CMD=DATA

26.删除不同时段的验证方式数据

应用场景：

常用于常用于删除某个不同时段的验证方式数据或 删除全部不同时段的验证方式数据

格式：

服务器下发命令：

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

客户端上传执行结果：

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

注释：

DiffTimezoneVS： 表名，指不同时段的验证方式表

$(Cond)：TimezoneID=x表示删除TimezoneID为x的不同时段的验证方式数据

用“*”表示删除全部不同时段的验证方式数据

示例：

删除TimezoneID=1的不同时段的验证方式数据：

C:342:DATA DELETE DiffTimezoneVS TimezoneID=1

客户端上传执行结果：

ID=342&return=0&CMD=DATA

27.删除门不同时段的验证方式数据

应用场景：

常用于常用于删除某个门不同时段的验证方式数据或 删除全部门不同时段的验证方式数据

格式：

服务器下发命令：

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

客户端上传执行结果：

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

注释：

DoorVSTimezone： 表名，指门不同时段的验证方式表

$(Cond)： DoorID=x表示删除DoorID为x的门不同时段的验证方式数据

用“*”表示删除全部门不同时段的验证方式数据

示例：

删除DoorID=1的门不同时段的验证方式数据：

C:342:DATA DELETE DoorVSTimezone DoorID=1

客户端上传执行结果：

ID=342&return=0&CMD=DATA

28.删除人不同时段的验证方式数据

应用场景：

常用于常用于删除某个人不同时段的验证方式数据或 删除全部人不同时段的验证方式数据

格式：

服务器下发命令：

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

客户端上传执行结果：

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

注释：

PersonalVSTimezone： 表名，指人不同时段的验证方式表

$(Cond)： PIN=x表示删除PIN为x的人不同时段的验证方式数据

PIN=x且DoorID=x表示删除PIN为x且DoorID为x的人不同时段的验证方式数据

用“*”表示删除全部人不同时段的验证方式数据

示例：

删除PIN=1或者PIN=1且DoorID=1的人不同时段的验证方式数据：

C:342:DATA DELETE PersonalVSTimezone PIN=1

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

客户端上传执行结果：

ID=342&return=0&CMD=DATA

ID=343&return=0&CMD=DATA

29.删除超级用户权限数据

应用场景：

常用于常用于删除某个超级用户权限数据或 删除全部超级用户权限数据

格式：

服务器下发命令：

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

客户端上传执行结果：

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

注释：

SuperAuthorize： 表名，指超级用户权限表

$(Cond)： Pin=x表示删除Pin为x的超级用户权限数据

用“*”表示删除全部超级用户权限数据

示例：

删除Pin=1的超级用户权限数据：

C:342:DATA DELETE SuperAuthorize Pin=1

客户端上传执行结果：

ID=342&return=0&CMD=DATA

30.删除比对照片

应用场景：

常用于删除用户比对照片

格式：

服务器下发命令：

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

客户端上传执行结果：

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

注释：

PIN=${XXX}：用户工号。

31.删除用户照片

应用场景：

删除用户照片

命令格式如下：

服务器下发命令：

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

客户端上传执行结果：

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

注释：

pin=${XXX}：用户工号。

12.1.3 COUNT 子命令

应用场景：

统计设备中指定表的数据数量或指定表中满足条件的数据数量

格式：

服务器下发命令：

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

客户端上传统计结果：

POST

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

&packidx=${XXX} HTTP/1.1

……

$(TableName)=${XXX}

客户端上传执行结果：

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

注释：

```
$(Cond):过滤条件列表,条件字段名1=值1\t 条件字段名2=值2\t 条件字段名3=值3。当过滤条件列表为*时，表示计算整个表；当过滤条件列表有多个时，多个条件为“且“的关系。
/iclock/querydata: 客户端上传符合条件的数据所使用的URI
type=count: 客户端上传数据类型为统计数量
count: 返回的数据条数。
packcnt: 总的包裹数。当数据较多时，需要分成多个包裹，该值用于说明总共分为多少个包裹发送。
packidx: 包裹的ID，表示当前发送的是所有包裹中的第几个包裹。
$(TableName)=${XXX}：等号左边是表名，右边的值为表中符合条件的数据的数量。
COUNT 子命令包含三次交互的过程： 客户端获取命令->服务器返回count命令->客户端返回统计结果->服务器返回ok->客户端返回命令执行结果->服务器返回oK。下文不再重复说明，只列出重点部分。
```

1.获取用户数量

应用场景：

```
常用于获取设备用户数量
```

格式：

```
服务器下发命令：
    C:${CmdID}:DATA$(SP)COUNT$(SP)user$(SP)$(Cond)
客户端上传统计结果：
    user=${XXX}
客户端上传执行结果：
    ID=${CmdID}&return=${XXX}&CMD=DATA
```

注释：

```
user： 表名，指获取用户信息表的数量
$(Cond)： 一般不设置条件，表示获取所有用户的数量
```

示例：

```
服务器下发命令：
    C:288:DATA COUNT user
客户端上传统计结果：
```

```
    POST
/iclock/querydata?SN=3383154200002&type=count&cmdid=288&tablename=user&count=1&packcnt=1&packidx=1
HTTP/1.1
    Cookie: token=af65a75608cf5b80fbb3b48f0b4df95a
    Host: 192.168.213.17:8088
    Content-Length: 6
    ......
    user=5
客户端上传执行结果：
    ID=288&return=0&CMD=DATA
```

2.获取设备门禁记录数量

应用场景：

```
用于获取客户端门禁记录数量
```

注释：

```
交互过程及格式与获取用户数量相似，只需将表名替换为“transaction”即可。
```

3.获取指纹模板数量

应用场景：

```
用于获取客户端指纹模板的数量
```

注释：

```
交互过程及格式与获取用户数量相似，只需将表名替换为“template10”即可。
```

12.1.4 QUERY 子命令

服务器主要求上传符合查询条件的数据

格式：

```
服务器下发命令：
    C:${CmdID}:DATA$(SP)QUERY$(SP)tablename=$(XXX),fielddesc=$(XXX),filter=$(XXX)
客户端上传符合条件的数据：
```

POST /iclock/querydata?SN=$(SerialNumber)&type=tabledata&cmdid=$(XXX)&tablename=$(TableName)&count=$(XXX)&packcnt=$(XXX)&packidx=$(XXX) HTTP/1.1

……

$(DataRecord)

服务器响应：

$(TableName)=$(XXX)

客户端上传执行结果：

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

注释：

tablename 表示要查询的表名

fielddesc 表示字段，当此字段为*时，表示查询表所有字段，否则查询指定字段

filter 表示查询条件，当此字段为*时，表示不过滤，当此字段为 NewRecord 时且tablename为transaction，表示查询新记录。

部分设备支持当此字段为 starttime=\tendtime=，表示查询指定时段的记录。

/iclock/querydata: 客户端上传符合条件的数据所使用的URI

type: 数据的类型，QUERY命令中为tabledata，表示为表中的数据

count: 返回的数据条数。

packcnt： 总的包裹数。当数据较多时，需要分成多个包裹，该值用于说明总共分为多少个包裹发送。

packidx： 包裹的ID，表示当前发送的是所有包裹中的第几个包裹。

$(DataRecord)： 需要上传的符合条件的数据，具体格式根据表名。

return： 返回值为本次上传的用户数

1.查询用户：

应用场景：

服务器主动要求上传需要的用户，常用于获取设备所有用户

格式：

服务器下发命令;

C:415:DATA$(SP)QUERY$(SP)tablename=user,fielddesc=$(XXX),filter=$(XXX)

客户端上传数据：

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

……

$(DataRecord)

服务器响应：

```
    user=$(XXX)
客户端上传执行成功结果
    ID=${CmdID}&return=$(XXX)&CMD=DATA
```

注释：

```
tablename： 为user时指用户表，接下来处理的是用户数据
$(DataRecord)： 表名为user时，数据格式为用户数据格式，用户数据格式已在【上传用户信息】中说明。
```

示例：

```
服务器下发命令;
    C:415:DATA QUERY tablename=user,fielddesc=*,filter=*
客户端上传3个用户的数据：
    POST
/iclock/querydata?SN=3383154200002&type=tabledata&cmdid=415&tablename=user&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
服务器响应已接受到3个用户：
    user=3
客户端上传执行成功结果
    ID=415&return=3&CMD=DATA
```

2.查询指纹模板

应用场景：

```
服务器主动要求客户端上传需要的指纹模板，常用于获取全部用户后要求获取全部指纹模板。
```

注释：

```
与查询用户相似，表名替换为 templatev10，$(DataRecord)格式为指纹模板格式，指纹模板格式已在【上传指纹模板】
中说明。
```

3.查询门禁记录

应用场景：

服务器主动要求客户端上传需要的门禁记录，常用于获取全部门禁记录或 account命令对账失败后获取指定区间内的门禁记录

格式：

服务器下发命令;

C:415:DATA$(SP)QUERY$(SP)tablename=transaction,fielddesc=$(XXX),filter=$(XXX)

客户端上传数据：

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

……

$(DataRecord)

服务器响应：

transaction=$(XXX)

客户端上传执行成功结果

ID=${CmdID}&return=$(XXX)&CMD=DATA

注释：

命令格式与查询用户相似，表名替换为 transaction

$(DataRecord)： 格式为门禁记录格式，具体如下：

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)

transaction: 表名，表示数据类型为门禁记录

cardno： 卡号

pin: 工号

verified： 验证方式码，详细请查阅【附录3】【验证方式码描述】

doorid： 所属门的ID

eventtype： 事件类型

inoutstate： 出入状态，0 表示入,1 表示出

time_second： 记录产生的时间戳，举例：time_second=583512660，time_second由【附录5】算法计算出来，使用【附录6】的方法得到年月日时分秒。

index： 门禁记录的ID

sitecode：区位码

devid：设备编码

多条记录之间使用${LF}连接

特别说明:

当filtert条件为“NewRecord”时，表示查询未上传过的新记录。

示例：

服务器下发命令：

C:415:DATA QUERY tablename=user,fielddesc=*,filter=*

客户端上传门禁记录:

POST /iclock/querydata?SN=3383154200002&type=tabledata&cmdid=415&tablename=transaction&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

服务器响应：

transaction=1

客户端上传执行成功结果：

ID=415&return=1&CMD=DATA

4.WIFI列表：

应用场景：

软件下发查询wifi列表命令，设备查询到附近的wifi后，将相应的ssid等信息上传给软件

格式：

服务器下发命令;

C:415:DATA$(SP)QUERY$(SP)tablename=[APList],fielddesc=$(XXX),filter=$(XXX)

客户端上传数据：

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

……

$(DataRecord)

服务器响应：

APList=$(XXX)

客户端上传执行成功结果

ID=${CmdID}&return=$(XXX)&CMD=DATA

注释：

tablename： 为[APList]，指Wifi虚拟表

$(DataRecord)： Wifi数据格式如下：

APList$(SP)ecn=$(XXX)$(HT)ssid=$(XXX)$(HT)rssi=$(XXX)$(HT)mac=$(XXX)

APList：表示Wifi虚拟表

ecn：直接拥塞指示标志

ssid：WIFI的用户名

rssi：信号

mac：MAC地址

示例：

服务器下发命令;

C:415:DATA QUERY tablename=[APList],fielddesc=*,filter=*

客户端上传的数据：

上传请求协议：

POST /iclock/querydata?SN=$(SerialNumber)&type=vmtable&cmdid=415&tablename=APList&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=%?

服务器响应：

APList=3

客户端上传执行成功结果

ID=415&return=3&CMD=DATA

12.2 ACCOUNT

目前仅用于上传门禁记录供服务器软件对账使用。【对账】指软件根据自身已有的门禁记录和设备上传的门禁记录进行比较，判断是否缺少了某些门禁记录。

12.2.1 一体机

格式：

服务器下发命令：

C:${CmdID}:ACCOUNT$(SP)transaction$(SP)ContentType=tgz$(SP)MaxIndex=0

客户端上传符合条件的数据：

POST /iclock/querydata?SN=$(SerialNumber)&type=tabledata&cmdid=${CmdID}&tablename=transaction&count=${XXX}&datafmt=3&Stamp=9999

……

name=transaction.tgz

size=${XXX}

datacode=base64

datatype=tgz$(NULL)$(DataRecord)

服务器响应：

transaction=${XXX}

客户端上传执行结果：

ID=${CmdID}&return=${XXX}&CMD=ACCOUNT&transaction&StartTime=${XXX}&EndTime=${XXX}&RecordSum=${XXX}

注释：

transaction： 表名，指门禁记录

ContentType: 文本格式，tgz表示压缩包格式

MaxIndex： 需要上传门禁记录ID大于该值的门禁记录，为0时表示上传全部门禁记录

/iclock/querydata: 客户端上传符合条件的数据时所使用的URI

type: 上传的数据类型，account命令中为tabledata，表示为表中的数据

tablename： 表名，目前account命令仅用于门禁记录，因此account中的表名都为transaction

count： 上传的数据条数

datafmt： 上传的数据的格式，该命令下现阶段固定写为3，表示为压缩包格式。

Stamp： 时间戳，该命令下现阶段固定写为9999。

name： 压缩包名称，transaction.tgz表示为门禁记录压缩包

size： 压缩包大小

datacode： 上传压缩包时使用的数据格式，base64表示为base64格式，需先将压缩包文件转为base64数据格式

datatype： 数据格式，tgz表示压缩包格式

$(DataRecord)： 多条门禁记录（格式参考【上传门禁记录】）组合成的数据打包为压缩包后，再转为为base64的数据。

transaction=${XXX}： ${XXX}为客户端上传的门禁记录条数

return： 上传的符合条件的门禁记录数

CMD： account命令中固定为ACCOUNT

StartTime： 门禁记录的开始时间，目前预留，上传时传空即可

EndTime： 门禁记录的结束时间，目前预留，上传时传空即可

RecordSum： 上传的符合条件的门禁记录数

示例：

需要以压缩包格式上传所有门禁记录：

服务器下发命令：

C:291:ACCOUNT transaction ContentType=tgz MaxIndex=0

客户端上传41条门禁记录组成的数据：

POST /iclock/querydata?SN=$(SerialNumber)&type=tabledata&cmdid=291&tablename=transaction&count=41&datafmt=3&Stamp=9999

……

name=transaction.tgz

size=636

datacode=base64

datatype=tgz\0【base64数据流】

服务器响应：

transaction=41

客户端上传命令执行结果：

ID=291&Return=41&CMD=ACCOUNT&transaction&StartTime=&EndTime=&RecordSum=41

12.2.2 控制器

格式：

服务器下发命令有三种：

1) C:${CmdID}:ACCOUNT$(SP)transaction$(SP)MaxIndex=${XXX}

2) C:${CmdID}:ACCOUNT$(SP)transaction$(SP)StartIndex=${XXX}$(HT)EndIndex=${XXX}

3) C:${CmdID}:ACCOUNT$(SP)transaction$(SP)StartTime=${XXX}$(HT)EndTime=${XXX}

客户端上传符合条件的数据：

首先，设备上传概要信息：

URL:

POST /iclock/cdata?SN=$(SerialNumber)&cmdid=${CmdID}&desc=overview HTTP/1.1

Content:

SumCount=${XXX}

FileName=${XXX}

ContentType=${XXX}

FileCount=${XXX}

服务器返回：

OK

然后，设备将门禁记录以文件的方式上传：

URL:

POST /iclock/file?SN=$(SerialNumber)&cmdid=${CmdID}&fileseq=${XXX}&contenttype=tgz&table=transaction&count=$(XXX) HTTP/1.1

Content:

transaction.tgz内容。

transaction.tgz产生过程：

a. 先将count数量的门禁记录转化成门禁记录push上传格式，存放在transaction.txt，例如

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. 将transaction.txt转化成transaction.tgz

服务器返回：

OK

客户端上传执行结果：

ID=${CmdID}&return=${XXX}&CMD=ACCOUNT

注释：

transaction： 表名，指门禁记录

MaxIndex： 需要上传门禁记录ID大于该值的门禁记录，为0时表示上传全部门禁记录

StartIndex：需要上传的门禁记录ID大于等于该值的门禁记录。

EndIndex：需要上传的门禁记录ID小于等于该值的门禁记录。

StartTime：需要上传的门禁记录时间大于等于该值的门禁记录。时间格式为XXXX-XX-XX XX:XX:XX，比如2018-02-22 16:19:20

EndTime:需要上传的门禁记录时间小于等于该值的门禁记录。时间格式为XXXX-XX-XX XX:XX:XX,比如2018-02-22 16:19:20

SumCount：门禁记录总条数

FileName：文件名称

ContentType：文本格式，tgz表示压缩包格式

FileCount：符合条件的门禁记录分包数

fileseq：上传文件分包索引，从1开始

12.3 Test Host命令

应用场景：

软件下发测试网络命令，设备接到命令，根据提供IP地址及端口进行连接测试，最后回复结果

格式：

服务器下发命令：

C:295:Test Host Address=110.80.38.74:8092

客户端上传执行结果：

ID=${CmdID}&return=${XXX}&CMD=Test(SP)Host

注释:

Address: 测试服务器的IP和端口。

12.4 CONTROL DEVICE 控制类命令

应用场景：

控制设备重启、远程开门、远程关门、取消报警、开启常开、禁用常开、控制辅助输出。

格式：

服务器下发命令：

C:$(CmdID):CONTROL$(SP)DEVICE$(SP)AABBCCDDEE

客户端上传执行结果：

```
ID=${CmdID}&return=${XXX}&CMD=CONTROL$(SP)DEVICE
```

注释:

AA 、BB、CC、DD 为四组双字节字符串，每一组分别由一个字节的整数经%02X转换而来。其中 AA表示控制命令类型，说明如下

AA	BB	CC	DD	EE
01 表示控制输出	00 表示开所有门，01-10 表示门 ID	01 表示控制锁，02 表示控制辅助输出	00 表示关闭，FF 表示常开，01-FF 表示打开时长	操作者
02 表示取消报警	00 表示取消所有报警，01-10 表示门 ID			操作者
03 表示重启设备	00 表示重启当前设备，01-10 表示从设备ID			操作者
04 表示控制常开	00 表示所有门，01-10 表示门 ID	00 表示禁用，01 表示开启		操作者
05 表示下发管理员到门	00 表示所有门，01-10 表示门 ID			操作者
06 表示锁定/解锁门	00 表示开所有门，01-10 表示门 ID	00 表示解锁，01 表示锁定		操作者
07 表示控制紧急双开、双关	01 紧急双开，02 紧急双关			操作者
08 zigbee 命令	01 重新组网，02 停止重新组网，03 允许入网，04 停止入网			操作者
09 韦根测试命令	01 开始韦根测试，02 结束韦根测试			操作者
10 485亮灯命令				操作者
11 切换网络命令	01 有线切4G，02 有线切WIFI，03 4G切有线，04 WIFI切有线			操作者
12 身份证登记模式命令	01 开始身份证登记模式，00 结束身份证登记模式	00 表示开所有门，01-10 表示门 ID	超时时间	操作者
13 查询子设备连接状态命令				操作者

示例：

1)服务器下发把门1开启5秒的命令

```
C:221:CONTROL DEVICE 01010105
```

2）服务器下发取消门1报警的命令：

```
C:222:CONTROL DEVICE 02010000
```

3）服务器下发重启当前设备命令：

```
C:223:CONTROL DEVICE 03000000
```

4）服务器下发韦根测试命令：

```
C:224:CONTROL DEVICE 0900000000
```

收到命令后，上传请求协议：

```
POST /iclock/cdata?SN=$(SerialNumber)&table=testdata HTTP/1.1
Cookie: token=${XXX}
Host: ${ServerIP}:${ServerPort}
Content-Length: ${XXX}
.....
type=wg\tbitscount=%?\tbits=%?\tsn=%?
```

服务器回复：

```
OK
```

5）服务器下发查询子设备连接状态命令：

```
C:225:CONTROL DEVICE 0D000000
```

6）切换网络命令：

有线切4G：C:226:CONTROL DEVICE 0B010000

有线切WIFI：

```
C:227:SET OPTIONS TempWirelessSSID=11,TempWirelessKey=11(其中TempWirelessSSID：WIFI SSID，WIFI秘钥)
C:228:CONTROL DEVICE 0B020000
```

4G切有线：C:229:CONTROL DEVICE 0B030000

WIFI切有线：C:230:CONTROL DEVICE 0B040000

7）客户端上传执行成功结果：

```
ID=224&return=0&CMD=CONTROL DEVICE
```

12.4.1 UPGRADE

远程固件升级

应用场景：

用于从服务器软件上远程升级客户端设备的固件

远程固件升级 方式一:

应用场景：

远程升级客户端固件，兼容控制器和新架构一体机，升级文件需要服务器转格式后下发到客户端

格式：

服务器下发命令：

C:${CmdID}:UPGRADE$(SP)checksum=${XXX},url=$(URL),size=${XXX}

客户端从命令中带有的URL下载升级包：

GET /iclock/file?SN=$(SerialNumber)&url=$(URL) HTTP/1.1

……

客户端上传执行结果：

ID=${CmdID}&Return=${XXX}&CMD=UPGRADE

注释：

Checksum：表示 md5 校验值

url：表示升级文件下载资源地址，升级文件名称为emfw.cfg

size 表示原始文件大小

特别说明：

本方法中，固件升级文件在下发时，由服务器转化为base64格式数据。客户端接收到后需要转化为二进制格式且名称为emfw.cfg的文件

示例：

服务器下发升级固件命令：

C:384:UPGRADE checksum=a5bf4dcd6020f408589224274aab132d,url=http*//localhost*8088\fireware\F20\admin\emfw.cfg,size=2312

客户端请求下载升级包：

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

……

客户端上传执行成功结果：

ID=384&Return=0&CMD=UPGRADE

远程固件升级 方式二:

应用场景：

远程升级客户端固件，直接获取文件，不需要转格式，客户端直接获取文件

格式：

服务器下发命令：

C:${CmdID}:UPGRADE$(SP)type=1,checksum=${XXX},size=${XXX},url=$(URL)

客户端请求下载升级包：

GET /iclock/file?SN=$(SerialNumber)&url=$(URL) HTTP/1.1

Cookie: token=af65a75608cf5b80fbb3b48f0b4df95a

Host: 192.168.213.17:8088

……

客户端上传执行结果：

ID=${CmdID}&Return=${XXX}&CMD=UPGRADE

注释：

type：为1 表示从url获取升级文件，暂时只支持1。

Checksum：表示 md5 校验值

url：表示升级文件下载资源地址，升级文件名称为emfw.cfg

size 表示升级包大小

特别说明：

本方法中，客户端直接获取到的就是固件升级文件，无需再转格式。

示例：

服务器下发命令：

C:123:UPGRADE
type=1,checksum=oqoier9883kjankdefi894eu,size=6558,url=http://192.168.0.13:89/data/emfw.cfg

客户端请求下载升级包：

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

……

客户端上传执行成功结果：

ID=384&Return=0&CMD=UPGRADE

12.5 配置类

对客户端进行配置或获取客户端的配置信息。

12.5.1 SET OPTIONS

应用场景：

设置客户端的一些配置参数，如ip、网关、锁驱动时长等

格式：

服务器下发命令：

C:$(CmdID):SET$(SP)OPTIONS$(SP)参数-值列表

客户端上传执行结果：

ID=$(CmdID)&Return=0&CMD=SET OPTIONS

注释：

参数列表采用 参数名=参数值 的形式，如果有多个参数则用逗号隔开，最后一个不用逗号

示例:

修改ip、网关、和掩码

C:403:SET OPTIONS IPAddress=192.168.213.221,GATEIPAddress=192.168.213.1,NetMask=255.255.255.0

设置客户端中的服务器IP和端口

C:399:SET OPTIONS WebServerIP=192.168.213.221,WebServerPort=8088

同步时间到客户端：

C:401:SET OPTIONS DateTime=583080894

注释：

DateTime：值为服务器的当前时间并使用【附录5】的方法转换得到的秒数。

部分设备收到这个命令后同步时间结束，部分设备执行完这个命令后设备会立即触发下面的请求，需要进行兼容处理。

请求：

GET /iclock/rtdata?SN=3383154200002&type=time HTTP/1.1

Cookie: token=af65a75608cf5b80fbb3b48f0b4df95a

Host: 192.168.213.17:8088

……

回复：

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

注释：

DateTime：值为格林威治时间并使用【附录5】的方法转换得到的秒数

ServerTZ：服务器的时区

修改通讯密码

C:402:SET OPTIONS ComPwd=1

设置门禁相关参数

C:405:SET OPTIONS Door1Drivertime=5,Door1KeepOpenTimeZone=0,Door1ValidTZ=1,Door1SupperPassWord=,Door1Intertime=0,Door1VerifyType=0,Door1SensorType=1,Door1Detectortime=15,Door1CloseAndLock=0,Door1ForcePassWord=,Reader1IOState=0,WiegandIDIn=1,WiegandID=1

Door1Drivertime： 门1锁驱动时长

Door1KeepOpenTimeZone： 门1的常开时间段

Door1ValidTZ： 门1的激活有效时间段

Door1SupperPassWord： 超级密码

Door1Intertime： 刷卡间隔（新架构设备不使用）

Door1VerifyType： 门1的验证方式

Door1SensorType： 门1的门磁类型 0 无，1，常开，2 常闭

Door1Detectortime： 门1门磁延时时长

Door1CloseAndLock： 是否支持闭门回锁功能

Door1ForcePassWord： 胁迫密码

Reader1IOState： 读头1的进出状态

WiegandIDIn：

WiegandID：

客户端上传执行成功结果：

ID=405&Return=0&CMD=SET OPTIONS

12.5.2 GET OPTIONS

应用场景：

获取设备的配置信息。

格式：

服务器下发命令：

C:$(CmdID):GET$(SP)OPTIONS$(SP)参数名列表

客户端上传需要获取的参数：

POST /iclock/querydata?SN=$(SerialNumber)&type=options&cmdid=$(CmdID)&tablename=options&count=1&packcnt=1&packidx=1 HTTP/1.1

……

参数-值列表

客户端上传执行成功结果：

ID=$(CmdID)&Return=0&CMD=GET OPTIONS

注释:

/iclock/querydata: 客户端上传符合条件的数据所使用的URI

type: 为options时表示客户端上传数据类型为参数

tablename：表名，为options表示为参数配置表

count: 返回的数据条数。

packcnt： 总的包裹数。当数据较多时，需要分成多个包裹，该值用于说明总共分为多少个包裹发送。

packidx： 包裹的ID，表示当前发送的是所有包裹中的第几个包裹。

如果有多个参数则用逗号隔开，最后一个参数不用逗号

示例：

C:408:GET OPTIONS

~SerialNumber,FirmVer,~DeviceName,LockCount,ReaderCount,AuxInCount,AuxOutCount,MachineType,~IsOnlyRFMachine,~MaxUserCount,~MaxAttLogCount,~MaxFingerCount,~MaxUserFingerCount,MThreshold,NetMask,GATEIPAddress,~ZKFPVersion,SimpleEventType,VerifyStyles,EventTypes,ComPwd

~SerialNumber：序列号

FirmVer： 固件版本号

DeviceName： 设备名称

LockCount： 门的数量

ReaderCount： 读头数量

AuxInCount： 辅助输入数量

AuxOutCount： 辅助输出数量

MachineType： 机器型号 一体机101

~IsOnlyRFMachine： 是否仅支持射频卡的开关参数

~MaxUserCount： 用户最大容量信息(卡最大容量信息)

~MaxAttLogCount： 考勤记录最大容量信息

~MaxUserFingerCount： 单个用户指纹最大容量

MThreshold： 指纹1：N匹配阈值

IPAddress： 有线网络IP地址

NetMask： 有线网络的子网掩码

GATEIPAddress： 有线网络的网关地址

~ZKFPVersion： 指纹算法版本号

SimpleEventType： 事件合并

VerifyStyles： 支持的验证方式，共32位，前16位每个位对应不同的验证方式组合，具体参考【附录3】【验证方式码 描述】，对应的位为1表示支持，为0表示不支持，值为200代表门禁事件。

EventTypes： 支持的事件类型，共32个字节（0~31），每个字节8位（用2个十六进制数表示），共256位（0~255），每个位对应的 功能参考【附录2】 【实时事件描述】，对应的位为1表示支持，为0表示不支持。

如 BF0FE03D300001000000000070000000000000000000000077002001000000，其中第0个字节为BF，二进制为11111101（低位在前），第6位为0，其它位为1，说明前八位控制的功能中，只有第6位表示的联动事件不支持，其他事件都支持。

ComPwd： 通讯密码

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

IclockSvrFun： ADMS功能开关,为1时打开，为0时关闭

OverallAntiFunOn: 区域反潜功能开关参数，未用

~REXInputFunOn: 出门按钮锁定功能开关参数

~CardFormatFunOn: 韦根读头是否支持自定义卡格式

~SupAuthrizeFunOn: 超级用户功能开关参数

~ReaderCFGFunOn: 读头配置功能开关参数,未用

~ReaderLinkageFunOn: 读头联动功能开关参数

~RelayStateFunOn: 支持继电器状态功能开关参数

~Ext485ReaderFunOn: 外接485读头功能开关参数

~TimeAPBFunOn： 入反潜时长功能开关参数

~CtlAllRelayFunOn： 开所有继电器功能开关参数

~LossCardFunOn: 删除所有黑名单功能

DisableUserFunOn： 黑名单功能开关参数

DeleteAndFunOn： 且关系的删除功能开关参数

LogIDFunOn: 记录id功能开关参数

DateFmtFunOn: 日期格式功能开关参数

DelAllLossCardFunOn: 删除所有黑名单功能开关参数

DelayOpenDoorFunOn: 延时开门功能开关，不支持

UserOpenDoorDelayFunOn: 对某用户延长开门时间功能开关，不支持

MultiCardInterTimeFunOn: 可设置多卡刷卡间隔功能开关，不支持

DSTFunOn: 夏令时功能功能开关参数，不支持

OutRelaySetFunOn: 输出配置功能开关参数，不支持

MachineTZFunOn: 可设置机器时区功能开关参数，不支持

AutoServerFunOn: 后台验证开关功能开关参数

PC485AsInbio485: 485读头/通信切换

MasterInbio485: 485读头/通信切换

RS232BaudRate: RS232/485波特率

AutoServerMode: 后台验证模式

IPCLinkFunOn: 开启对接IPC摄像机的软联动功能开关参数，不支持

IPCLinkServerIP: IPC摄像机地址,不支持

C:410:GET OPTIONS FingerFunOn,FvFunOn,FaceFunOn,~MaxFace7Count,~MaxFvCount

FingerFunOn: 指纹功能开关参数

FvFunOn: 指静脉功能开关参数

FaceFunOn: 人脸功能开关参数

MaxFace7Count: 最大面部模版数

MaxFvCount: 最大指静脉数

13. 后台验证

设备发送验证成功数据到服务器，服务器验证后，返回结果、验证成功数据及控制命令，详见如下

客户端请求消息

```
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}
```

注释：

```
HTTP请求方法使用：POST方法

URI使用：/iclock/cdata

HTTP协议版本使用：1.1

客户端配置信息：

    SN：${Required}表示客户端的序列号

    AuthType=device：表示后台全局反潜

Host头域：${Required}

Content-Length头域：${Required}

其他头域：${Optional}

请求实体：事件记录数据

    time：时间，格式为YYYY-MM-DD HH:MM:SS

    pin：工号，空时传0

    cardno：卡号

    addrtype：事件点类型，即谁产生的事件

        值        意义

        1        设备

        2        门

        3        读头

        4        辅助输入
```

```
        5       辅助输出
    eventaddr：事件地址，如果是设备，可以是设备地址，如果是读头，可以是读头地址，目前现在只用到了门
    event：事件类型
    inoutstatus：出入状态，表示是门的出还是入
    verifytype：验证类型
```

服务器正常响应消息

```
当存在用户信息时，回复信息如下：
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
```

注释：

```
第一行
    AUTH：表示是否验证成功
        值          意义
        SUCCESS     成功
        FAILED      失败
        TIMEOUT     超时
第二行
    请求的原始事件记录
第三行
    验证成功后，对应的控制器指令，详见"CONTROL DEVICE 控制类命令"
```

14. 附录

附录1 命令返回值描述

| 返回值| 描述|

| ---| ---|
>=0	成功
-1	命令发送失败
-2	命令没有回应
-3	需要的缓存不足
-4	解压失败
-5	读取数据长度不对
-6	解压的长度和期望的长度不一致
-7	命令重复
-8	连接尚未授权
-9	数据错误，CRC 校验失败
-10	数据错误，dataapi 无法解析
-11	数据参数错误
-12	命令执行错误
-13	命令错误，没有此命令
-14	通迅密码错误
-15	写文件失败
-16	读文件失败
-17	文件不存在
-18	设备没有空间
-19	校验和出错
-20	接受到的数据长度与给出的数据长度不一致
-21	设备中，没有设置平台参数
-22	固件升级，传来的固件的平台与本地的平台不一致
-23	固件升级,固件版本比设备中的固件版本老
-24	升级的文件标识出错
-25	固 件升级，传 来的文件名 不对，即不是emfw.cfg
-26	来的指纹模板长度为 0
-27	传来的指纹模板 PIN 号错误,找不到用户
-28	开时段执行开门命令
[-400,-100]为 pullsdk 内部返回值	
-100	表结构不存在
-101	表结构中，条件字段不存在
-102	字段总数不一致
-103	字段排序不一致
-104	实时事件数据错误
-105	解析数据时，数据错误
-106	数据溢出，下发数据超出 4M
-107	获取表结构失败
-108	无效 OPTIONS 选项
-201	LoadLibrary 失败
-202	调用接口失败
-203	通讯初始化失败
-206	串口代理程序启动失败，原因一般是串口不存在或串口被占用
-301	获取 TCP/IP 版本失败
[*,-600]为 push 通信独有的返回值[上面有的返回值，不能重复定义]	
-601	保留，固件内部使用
-603	内部错误:句柄无效
-604	设备内存不够
-609	内部错误:句柄无效
-612	设备保存参数失败
-614	远程取消报警失败
-615	远程重启失败
-616	远程常开或取消常开失败

-618	URL 格式错误
-619	固件内部错误:入参为空
-620	服务器下发数据格式错误
-621	表结构不支持此字段
-628	main 加载指纹到内存失败
-629	表名错误
-630	执行 shell 命令失败
-631	base64 长度不对
-632	文件名错误
-701	保留，固件内部使用
-702	保留，固件内部使用
-703	保留，固件内部使用
-704	保留，固件内部使用
-705	保留，固件内部使用
-706	保留，固件内部使用
-707	保留，固件内部使用
-708	固件不支持该命令
-709	上传 options 失败
-720	升级固件失败
-721	设备下载文件
-722	上传数据到软件失败
-723	保留，固件内部使用
-724	保留，固件内部使用
-725	设备等待结果超时
-726	服务器返回 HTTP 错误
-727	保留，固件内部使用
-728	上传表数据条数失败
-729	保留，固件内部使用
-730	服务器不应答
-731	服务器接收数据异常
-732	保留，固件内部使用
-801	命令格式错误,命令格式 【cmd:序号:命令】

附录2 实时事件描述

事件码说明：0-19、200-253为正常事件，20-99为异常事件，100-199为警告事件

| 事件码 | 原始描述 | 事件合并描述 | 备注|

|---|---|---|---|

| 0 | 正常刷卡开门 | 正常验证开门|0、14、17事件码合并,含义相同|

| 1 | 常开时间段内刷卡 |常开时间段内验证|1、16事件码合并,含义相同|

| 2 | 首人开门(刷卡) |首人开门|2、18、19事件码合并,含义相同|

| 3 | 多人开门(刷卡) |多人开门|3、15、203事件码合并,含义相同|

| 4 | 紧急状态密码开门|紧急状态密码开门 ||

| 5 | 常开时间段开门 |常开时间段开门 ||

| 6 | 触发联动 |触发联动 |联动事件记录中的验证方式的值为具体的事件类型|

| 7 | 取消报警 |取消报警||

| 8 | 远程开门 |远程开门 ||

| 9 | 远程关门 |远程关门 ||

| 10 | 禁用当天常开时间段 |禁用当天常开时间段 ||

| 11 | 启用当天常开时间段 |启用当天常开时间段 ||

| 12 | 辅助输出远程打开 |辅助输出远程打开 ||

| 13 | 辅助输出远程关闭 |辅助输出远程关闭 ||

| 14 | 正常按指纹开门 |正常验证开门|0、14、17事件码合并,含义相同|

| 15 | 多人开门(按指纹) |多人开门|3、15、203事件码合并,含义相同|

| 16 | 常开时间段内按指纹 |常开时间段内验证|1、16事件码合并,含义相同|

| 17 | 卡加指纹开门 |正常验证开门|0、14、17事件码合并,含义相同|

| 18 | 首人开门(按指纹) |首人开门|2、18、19事件码合并,含义相同|

| 19 | 首人开门(卡加指纹) |首人开门|2、18、19事件码合并,含义相同|

| 20 | 刷卡间隔太短 |操作间隔太短|20、31、50事件码合并,含义相同|

| 21 | 门非有效时间段(刷卡) |门非有效时间段验证开门|21、35、49事件码合并,含义相同|

| 22 | 非法时间段 |非法时间段 ||

| 23 | 非法访问 |非法访问 ||

| 24 | 反潜 |反潜 ||

| 25 | 互锁 |互锁 ||

| 26 | 多人验证(刷卡) |多人验证等待|26、32、51事件码合并,含义相同|

| 27 | 卡未注册 |人未登记|27、30、34事件码合并,含义相同|

| 28 | 门开超时 |门开超时 ||

| 29 | 卡已过有效期 |人已过有效期|29、33、53事件码合并,含义相同|

| 30 | 密码错误 |人未登记|27、30、34事件码合并,含义相同|

| 31 | 按指纹间隔太短 |操作间隔太短|20、31、50事件码合并,含义相同|

| 32 | 多人验证(按指纹) |多人验证等待|26、32、51事件码合并,含义相同|

| 33 | 指纹已过有效期 |人已过有效期|29、33、53事件码合并,含义相同|

| 34 | 指纹未注册 |人未登记|27、30、34事件码合并,含义相同|

| 35 | 门非有效时间段(按指纹) |门非有效时间段验证开门|21、35、49事件码合并,含义相同|

| 36 | 门非有效时间段(按出门按钮) |门非有效时间段(按出门按钮) ||

| 37 | 常开时间段无法关门 | 常开时间段无法关门 ||

| 38 | 卡已挂失 |卡已挂失 ||

| 39 | 黑名单 |黑名单 ||

| 40 | 多人验证失败(按指纹) |多人验证失败|40、48、52事件码合并,含义相同|

| 41 | 验证方式错误 |验证方式错误 ||

| 42 | 韦根格式错误 |韦根格式错误 ||

| | | | |

| 44 | 后台验证失败 |后台验证失败 ||

| 45 | 后台验证超时 |后台验证超时 ||

| | | | |

| 47 | 发送命令失败|发送命令失败||

| 48 | 多人验证失败(刷卡) |多人验证失败|40、48、52事件码合并,含义相同|

| 49 | 门非有效时间段(密码)|门非有效时间段验证开门|21、35、49事件码合并,含义相同|

| 50 | 按密码间隔太短|操作间隔太短|20、31、50事件码合并,含义相同|

| 51 | 多人验证(密码)|多人验证等待|26、32、51事件码合并,含义相同|

| 52 | 多人验证失败(密码)|多人验证失败|40、48、52事件码合并,含义相同|

| 53 | 密码已过有效期|人已过有效期|29、33、53事件码合并,含义相同|

| 54 | 电池电压过低|电池电压过低||

| 55 | 立即更换电池|立即更换电池||

| 56 | 非法操作|非法操作||

| 57 | 后备电源|后备电源||

| 58 | 常开报警|常开报警||

| 59 | 非法管理|非法管理||

| 60 | 门被反锁|门被反锁||

| 61 | 重复验证|重复验证||

| 62 | 禁止用户|禁止用户||

| | | | |

| 100 | 防拆报警 |防拆报警 ||

| 101 | 胁迫密码开门 |胁迫开门报警|101、103事件码合并,含义相同|

| 102 | 门被意外打开|门被意外打开||

| 103 | 胁迫指纹开门 |胁迫开门报警|101、103事件码合并,含义相同|

| 104 | 无效卡刷卡报警 |无效卡刷卡报警||

| 105 | 网络掉线 |无法连接服务器||

| 106 | 电池掉电 |电池掉电||

| 107 | 市电掉电 |市电掉电||

| 108 | 无法连接主控 |无法连接主控||

| | | | |

| 200 | 门已打开 |门已打开 ||

| 201 | 门已关闭 |门已关闭 ||

| 202 | 出门按钮开门 |出门按钮开门 ||

| 203 | 多人开门(卡加指纹) |多人开门|3、15、203事件码合并,含义相同|

| 204 | 门常开时间段结束 |门常开时间段结束 ||

| 205 | 远程开门常开 |远程开门常开 ||

| 206 | 设备启动 |设备启动 ||

| 207 | 密码开门 |密码开门 ||

| 208 | 超级用户开门 |超级用户开门 ||

| 209 | 触发出门按钮(被锁定) |触发出门按钮(被锁定) ||

| 210 | 启动消防开门 |启动消防开门 ||

| 211 | 超级用户关门 |超级用户关门 ||

| 212 | 开启电梯控制功能|开启电梯控制功能 ||

| 213 | 关闭电梯控制功能 |关闭电梯控制功能||

| 214 | 成功连接服务器 |成功连接服务器 ||

| 215 | 首卡开门(密码) |首卡开门(密码) ||

| 216 | 常开时间段内按密码 |常开时间段内按密码 ||

| 217 | 成功连接主控 |成功连接主控||

| 218 | 身份证通行 |身份证通行 ||

| | | | |

| 220 | 辅助输入点断开 |辅助输入点断开 ||

| 221 | 辅助输入点短路 |辅助输入点短路 ||

| 222 | 后台验证成功 |后台验证成功 ||

| 223 | 后台验证 |后台验证 ||

| | | | |

| 225 | 辅助输入点正常 |辅助输入点正常 ||

| 226 | 辅助输入点触发 |辅助输入点触发 ||

| 227 | 门双开 |门双开 ||

| 228 | 门双关 |门双关 ||

| 229 | 辅助输出定时常开 |辅助输出定时常开 ||

| 230 | 辅助输出定时关闭常开 |辅助输出定时关闭常开 ||

| 231 | IPC固件联动事件 |IPC固件联动事件 |联动事件记录中的验证方式的值为具体的事件类型|

| 232 | 验证通过 |验证通过 ||

| | | | |

| 237 | 辅助输入不在时间段内操作 |辅助输入不在时间段内操作 ||

| | | | |

| 254 | 扩展事件编号定义 |扩展事件编号定义|由于0-255不够用，254当作扩展事件编号位定义，固件新支持|

| 255 | 门状态 |门状态 ||

附录3 验证方式码描述

比如：VerifyStyles的值为FB1E1F00（11111011 00011110 00011111 00000000）：

|---|---|

|对应二进制|1|1|1|1|1|0|1|1|0|0|0|1|1|1|1|0|...|

|对应的位置|0|1|2|3|4|5|6|7|8|9|10|11|12|13|14|15|...|

比如位置15对应的二进制为0，表示验证方式不支持人脸（人脸验证方式的值15）

| 验证方式 | 描述 |

|---|---|

| 0 | 指静脉或人脸或指纹或卡或密码 (自动识别)|

| 1 | 仅指纹 |

| 2 | 工号验证 |

| 3 | 仅密码 |

| 4 | 仅卡 |

| 5 | 指纹或密码 |

| 6 | 指纹或卡 |

| 7 | 卡或密码 |

| 8 | 工号加指纹 |

| 9 | 指纹加密码 |

| 10 | 卡加指纹 |

| 11 | 卡加密码 |

| 12 | 指纹加密码加卡 |

| 13 | 工号加指纹加密码 |

| 14 | 工号加指纹 或 卡加指纹 |

| 15 | 人脸 |

| 16 | 人脸加指纹 |

| 17 | 人脸加密码 |

| 18 | 人脸加卡 |

| 19 | 人脸加指纹加卡 |

| 20 | 人脸加指纹加密码 |

| 21 | 指静脉|

| 22 | 指静脉加密码|

| 23 | 指静脉加卡 |

| 24 | 指静脉加密码加卡 |

| 200 | 其他 |

附录4

语言编号	意义
83	简体中文
69	英文
97	西班牙语
70	法语
66	阿拉伯语
80	葡萄牙语
82	俄语
71	德语
65	波斯语
76	泰语
73	印尼语
74	日本语
75	韩语
86	越南语
116	土耳其语
72	希伯来语
90	捷克语
68	荷兰语
105	意大利语
89	斯洛伐克语
103	希腊语
112	波兰语
84	繁体

附录5

中控算法：将年、月、日、时、分、秒转化成秒的算法

```
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;
}
```

附录6

中控算法：将秒转化成年、月、日、时、分秒的算法

```
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;
}
```

附录7

设备类型值	对应机型
1	C3-200
2	C3-400
3	C4-400
4	C3-100
5	C4-200、INBIO280
6	C4（4门转2门）
7	C3（4门转2门）
8	C3-160、INBIO160
9	C3-260、INBIO260

10	C3-460、INBIO460
23	C5-100
24	C5-200
25	C5-400
26	INBIO5-100
27	INBIO5-200
28	INBIO5-400
40	BIOIR9000
101	一体机
102	指静脉
103	iface7
301	无线锁

附录8

反潜值	描述
0	无反潜
1	门1与2反潜
2	门3与4反潜
3	门1与2反潜，3与4反潜
4	门1或2与3或4反潜
5	门1与2或3反潜
6	门1与2或3或4反潜
16	读头1间反潜
32	读头2间反潜
48	读头1，2间各自同时反潜
64	读头3间反潜
80	读头1，3间各自同时反潜
96	读头2，3间各自同时反潜
112	读头1，2，3间各自同时反潜
128	读头4间反潜
144	读头1，4间各自同时反潜
160	读头2，4间各自同时反潜
176	读头1，2，4间各自同时反潜
196	读头3，4间各自同时反潜
208	读头1，3，4间各自同时反潜
224	读头2，3，4间各自同时反潜

| 240 | 读头1，2，3，4间各自同时反潜 |

附录9

互锁值	描述
0	无
1	1与2号门互锁
2	3与4号门互锁
3	1与2与3号门互锁
4	1与2号门间互锁，或 3与4号门间互锁
5	1与2与3与4门互锁

附录10

门参数表参数名	描述
DoorAutoMatch	韦根格式自动匹配，0 自定义，1 自动
DoorDrivertime	锁驱动时长
DoorKeepOpenTimeZone	门常开时间段，0 不常开
DoorCancelKeepOpenDelay	取消常开时间，(linux年) * 10000 + (linux月) * 100 + 日
DoorValidTZ	门有效时间段
DoorSupperPassWord	紧急密码，最大8位
DoorIntertime	刷卡间隔（秒），默认2秒
DoorVerifyType	开门方式，见【附录3 验证方式表】
DoorSensorType	门磁类型，0 无，1 常开，2 常闭
DoorDetectortime	门磁延时（秒），0 延时不报警，>0 延时报警
DoorCloseAndLock	闭门回锁，1 启用，0不启用，默认1
DoorReaderType	读头类型
DoorForcePassWord	协迫密码，最大6位
DoorInTimeAPB	入反潜时长（秒）
DoorREXInput	出门按钮状态，1：不锁定(默认不锁定) 0：锁定
DoorREXTimeOut	出门按钮延时（秒），门锁定时按键开关事件发生后的延时
WiegandIDIn	韦根输入类型
WiegandID	韦根输出类型
DoorDelayOpenTime	开门延时，验证完延时开门
ExtDoorDelayDrivertime	针对残疾人，延长通行时间（秒）

| DoorMultiCardInterTime | 多人开门操作间隔（秒） |

| DoorFirstCardOpenDoor | 首卡常开，0：不启用 1：首卡常开 2：首卡激活 |

| DoorMultiCardOpenDoor | 多卡开门，0：不启用 1：启用 |

| DoorDisable | 门启用或禁用，1：启用，0：禁用 |

| DoorInOutAntiPassback | 单门出入反潜，0：不反潜，1：入反潜，2：出反潜，3：出入反潜 |

| DoorMaskFlag | 门锁定，1：锁定 |

附录11

| 读头参数表参数名 | 描述 |

| ----- | ---- |

| IdentityCardVerifyMode | 0：普通模式；1：身份证模式 |

附录12

| 设备参数表参数名 | 描述 |

| ----- | ---- |

| DSTOn | 夏令时功能 |

| CurTimeMode | 当前模式，1当前为夏令时，2当前不是夏令时 |

| DLSTMode | 夏令时模式 |

| IPAddress | IP地址 |

| GATEIPAddress | 网关 |

| NetMask | 子网掩码 |

| BackupTime | 定时备份事件记录，精确度：小时 |

| DelRecord | 删除记录数，事件记录满之后要删除的事件记录数量 |

| MachineTZ | 时区 |

| AntiPassback | 反潜 ，见【附录8】 |

| InterLock | 互锁，见【附录9】 |

附录13 协议版本规则

- 已发布的协议版本：

```
3.0.1
```

- 设备端：

```
设备将当前push使用的协议版本通过下面协议推送给服务端

GET /iclock/cdata?SN=${SerialNumber}&options=all&pushver=${XXX}
```

服务端针对这个请求返回服务端使用哪个发布协议版本开发，将协议版本返回给设备。

PushProtVer=xxx，没返回这个参数的话，设备默认服务器使用的协议版本为3.0.1

设备根据当前push使用的协议版本与服务端返回的协议版本比较，使用较低的那个版本交互。

- 服务端：

服务端根据下面的请求获得设备端push使用的协议版本，如果没有pushver字段，那么默认设备使用的是3.0.1协议版本。

GET /iclock/cdata?SN=${SerialNumber}&options=all&pushver=${XXX}

服务端需要返回软件使用哪个发布协议版本：

PushProtVer=xxx

服务端根据软件使用的协议版本与设备端上传的协议版本比较，使用较低的那个版本交互。

附录14

- 参数CmdFormat的值含义，这个参数不存在默认为0。

值为0时，如下格式：

C:XXX:COMMAND……

值为1时，格式如下：

DataType=1,SN=%s,Priority=%d,CmdID=%s,CmdDesc=%s

注释：

DataType：数据格式类型，默认为1

SN：命令分发到指定的序列号

Priority：命令优先级，默认为0

CmdID：命令ID

如

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=1 Name=DoorAutoMatch Value=1 DevID=1

ID=1 Name=DoorDrivertime Value=5 DevID=1

ID=1 Name=DoorKeepOpenTimeZone Value=0 DevID=1

ID=1 Name=DoorValidTZ Value=1 DevID=1

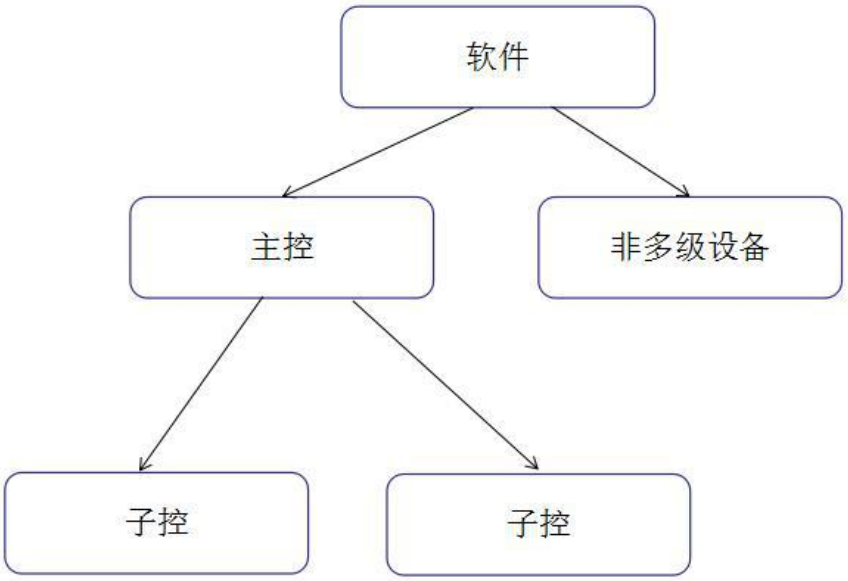
ID=1　　Name=DoorSupperPassWord Value=　DevID=1

针对CmdFormat=1，所有的C:XXX:COMMAND命令需要增加下面的前缀数据

DataType=1,SN=%s,Priority=%d,CmdID=%s,CmdDesc=

附录15

多级控制效果图



多级控制参数描述

- MultiStageControlFunOn //=1开启 =0 关闭 多级控制功能开启，默认值为0
- MasterControlOn //=1 主控开启 =0 主控关闭，默认值为0
- SubControlOn //=1 子控开启 =0 子控关闭，默认值为0

应用场景配置

- 非多级设备，见【附录7】

• MultiStageControlFunOn、SubControlOn、MasterControlOn不存在或者等于以下值的时候
• MultiStageControlFunOn=1或者MultiStageControlFunOn=0
• SubControlOn=0
• MasterControlOn=0

- 子控

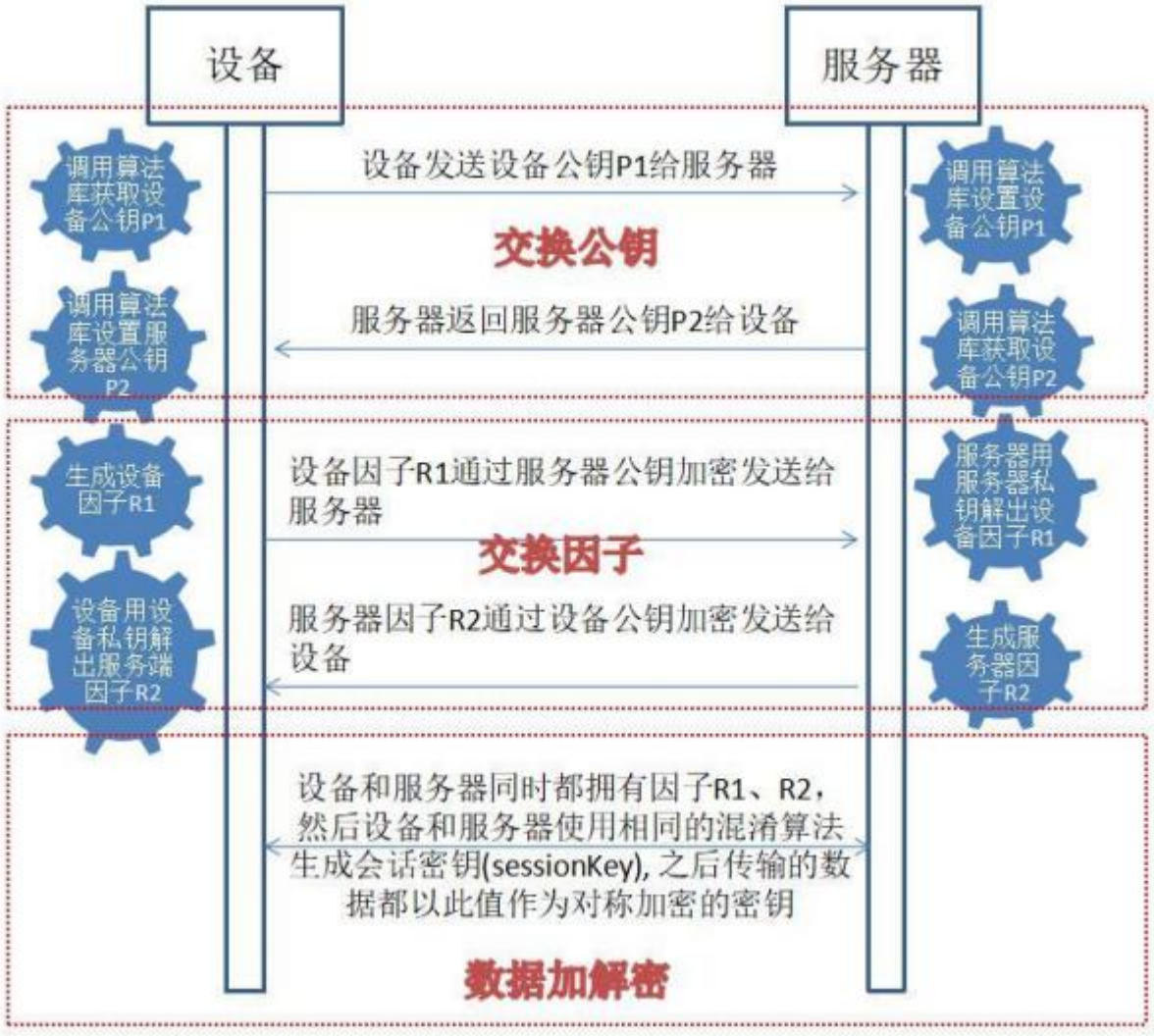
- MultiStageControlFunOn=1
- SubControlOn=1
- MasterControlOn=0

- 主控

- MultiStageControlFunOn=1
- MasterControlOn=1
- SubControlOn=0

附录16

数据加密秘钥交换方案



- 算法：加密算法库将统一进行封装，设备使用的算法库为静态库。
- 方案：
  a) 设备和服务器重连的时候初始化非对称加密的公私钥。
  b) 设备和服务器交换公钥：
     ■设备发送设备公钥 P1 给服务器。
     ■服务器返回服务器公钥 P2 给设备。

■完成公钥交换。设备和服务器同时都拥有公钥 P1、P2。

c) 设备和服务器交换因子：

■设备生成因子 R1，并通过服务器公钥加密发送给服务端。

■服务器用服务器私钥解出设备因子 R1。

■服务器生成因子 R2，并通过设备公钥加密发送给设备。

■设备用设备私钥解出服务端因子 R2。

■完成因子交换。设备和服务器同时都拥有因子 R1、R2。

d) 设备和服务器同时都拥有因子 R1、R2，然后设备和服务器使用相同的混淆算法生成会话密钥(sessionKey)，之后传输的数据都以此值作为对称加密的密钥。

兼容方案

根据设备和服务器使用的协议版本实现兼容，情况如下：

- 情况一


设备
http/https
服务器
Options=all请求推送设备的推送协议版本或者没推送协议版本
服务器返回后者未返回服务器开发时使用的协议版本，两者协议没有都支持数据加密
对通信数据明码传输

- 情况二


设备
http/https
服务器
Options=all请求推送设备的协议版本
服务器返回服务器开发时使用的协议版本，两者协议支持数据加密
调用算法库获取设备公钥P1
发送设备公钥P1给服务器
调用算法库设置设备公钥P1
调用算法库设置服务器公钥P2
服务器返回服务器公钥P2给设备
调用算法库获取获取公钥P2
生成设备因子R1
设备因子R1通过服务器公钥加密发送给服务器
服务器用服务器私钥解出设备因子R1
设备用设备私钥解出服务端因子R2
服务器因子R2通过设备公钥加密发送给设备
生成服务器因子R2
设备和服务器同时都拥有因子R1、R2，然后设备和服务器使用相同的混淆算法生成会话密钥(sessionKey)，之后传输的数据都以此值作为对称加密的密钥
对通信数据签名,使用crc32校验算法

注释：

a) 设备根据设置的服务器地址来判断使用https还是http传输。

b) 设备现有的第一个请求协议头中增加pushver字段为设备当前通信协议版本号，软件返回的数据内容中增加PushProtVer表示软件是基于哪个协议版本开发的。设备和服务器两个协议版本比较取最低的，按最低协议版本进行通信。

情况一： 当服务器和设备的协议版本不都支持，则使用对数据通信进行明码传输。

情况二： 设定某个协议版本是支持数据加密的，当服务器和设备的协议版本都支持，则使用数据加密方案。

交互顺序如下：

■新增协议对设备和服务器的公钥 P1、P2 进行交换。

■新增协议对设备和服务器的因子 R1、R2 进行交换。

■对通信数据签名进行 crc32 校验，设备和服务器同时都拥有因子 R1、R2，然后设备和服务器使用相同的混淆算法生成会话密钥(sessionKey)，之后传输的数据都以此值作为对称加密的密钥。

全国免费技术咨询热线：4006-900-999
广东省东莞市塘厦平山188工业大道26号中控智慧产业园
广东省深圳市龙岗区坂田五和大道北中控智慧大厦
厦门市集美区软件园三期B02栋20层
www.zkteco.com