ZKSoftware Fingerprint Identification System

Technical manual for lower communication protocol

Version: 1.0

Date: June, 2008

CONTENTS

1 Lower communication protocol (preface)

This manual describes how to communicate with ZKSoftware identification machine.   It assists user in developing DEMO program on the basis of SDK provided by ZKSoftware.

The manual has introduced lower communication method and described the structure of communication packet. For the involved command character, data structure and response, there is specific description.

2 Communication method:

2.1 Physical connection

- RS232\RS485
  Communicate through RS232/RS485.

2.2 Network part

- **UDP**
  In the machine, there is a UDP Server at 4370 port for monitoring. Data transmission and reception can be done through UDP protocol. There is few data in the machine. The maximum information read is fingerprint template, which is usually 600 bytes.
- **others**
  There is WebServer built-in some machines. It can communicate through Http. Secondly, some machines can support SOAP interface and communicate through SOAP protocol. Different machines may have different functions. This information is only for reference.

2.3 Structure definition of data communication packet

2.3.1 Structure definition of data communication header

```
typedef struct _CmdHdr_{
        Unsigned short Command, CheckSum, SessionID, ReplyID;
}TCmdHeader, *PCmdHeader;
```

Explanation:

Command: Command character

CheckSum: Check sum (including header and data).Algorithm: accumulate the whole packet on the basis of unsigned short. It exceeds 2147483648 (long 4 bytes), take the value 2 bytes lower for further accumulation. Take bitwise inverse of the accumulated value, and then change it to unsigned short (2 bytes) to get the check sum.

SessionID: Session ID. It is the only one the mark a connection. There is machine allocated and returned when the machine is in connection.

ReplyID: ReplyID. It is the only one to mark the sent command at present. It is an accumulated value used for identifying command. That is, from the beginning of connection, ReplyID of every command is different.

2.3.2 Structure definition of data communication packet

Data communication structure: header+data to be sent. The data packet responded by the machine has the same structure with this packet.

Than is, command character (2bytes)+check sum (2bytes) + session ID (2bytes) +a data packet with corresponding number (2bytes).The sent packet and the received packet are symmetrical.

2.4 Data communication example

Take connection request for example:

Low-level part:

Create socket:

Initialize header:

TCmdHeader. Command = CMD_CONNECT;

TCmdHeader. Checksum = 0;

TCmdHeader. SessionID = 0

TCmdHeader. ReplyID = 1

Data buf length is 0.

Calculate check sum: CheckSum = 64534.

Socket sends command.

sock.SendTo(buffer, len, 0, endPointDst)

Buffer is data packet: header+data.

Len is data packet length.

0 is the mark of socket.

endPointDst is destination IP address and port.

Wait for machine’s disposal. Socket receives data packet.

Judge command character TCmdHeader. Whether Command is CMD_ACK_OK., if it is, the command will be executed successfully.

Machine:

Check whether there is data packet received all along.

If there is data packet received, deal with the data packet:

Judge whether the obtained session is null and the command character chdr->Command is not CMD_CONNECT. Then the disposal fails. If the session is obtained, set chdr->CheckSumc as 0 and chdr->Command as CMD_ACK_OK.

Judge whether it is connected to request command. If it is, session will be created. The data packet responsed by machine has the same structure with this packet.Header chdr. Among them, session->SessionID (send to Lowlevel successfully) is 52060, chdr->SessionID = session->SessionID, and chdr->CheckSum is 11474 by algorithm.

The header at the moment:

TCmdHeader. Command = CMD_ACK_OK;

TCmdHeader. CheckSum = 11474;

TCmdHeader. SessionID = 52060

TCmdHeader. ReplyID = 1

Send data packet to Lowlevel.Part length of data is 0.

3 Command, response and data structure

3.1 Definition table of command character

Definition table of command character

Command	Value	description
Command character of connection and device		
CMD_CONNECT	1000	Connection request
CMD_EXIT	1001	Disconnect
CMD_ENABLEDEVICE	1002	Enable machine in normal work state
CMD_DISABLEDEVICE	1003	Disable machine in work state, display “in the work …” on LCD
CMD_RESTART	1004	Restart machine
CMD_POWEROFF	1005	Power off
CMD_SLEEP	1006	Enable machine in sleep
CMD_RESUME	1007	Awake sleeping machine (not support at present)
CMD_CAPTUREFINGER	1009	Capture fingerprint image
CMD_TEST_TEMP	1011	Test whether a fingerprint exists
CMD_CAPTUREIMAGE	1012	Capture all image
CMD_REFRESHDATA	1013	Refresh data in the machine
CMD_REFRESHOPTION	1014	Refresh configuration parameter
CMD_TESTVOICE	1017	Play voice
CMD_GET_VERSION	1100	Get firmware version
CMD_CHANGE_SPEED	1101	Change transmission speed
CMD_AUTH	1102	Connection authorization
CMD_PREPARE_DATA	1500	Prepare to transmit data
CMD_DATA	1501	send a data packet
CMD_FREE_DATA	1502	Free buffer memory
Command character of data management:		
CMD_DB_RRQ	7	Read a data in machine
CMD_USER_WRQ	8	Upload user information (from PC to terminal)
CMD_USERTEMP_RRQ	9	Read a fingerprint template or all data
CMD_USERTEMP_WRQ	10	Upload a fingerprint template
CMD_OPTIONS_RRQ	11	Read a configuration parameter in the machine
CMD_OPTIONS_WRQ	12	Set machine configuration parameter
CMD_ATTLOG_RRQ	13	Read all attendance record
CMD_CLEAR_DATA	14	Clear data

CMD_CLEAR_ATTLOG	15	Clear attendance log
CMD_DELETE_USER	18	Delete some user
CMD_DELETE_USERTEMP	19	delete a fingerprint template
CMD_CLEAR_ADMIN	20	Clear administrator
Command character of access control		
CMD_USERGRP_RRQ	21	Read user subgroup
CMD_USERGRP_WRQ	22	Set user subgroup
CMD_USERTZ_RRQ	23	Read user time zone setting
CMD_USERTZ_WRQ	24	Write user time zone setting
CMD_GRPTZ_RRQ	25	Read group time zone setting
CMD_GRPTZ_WRQ	26	write group time zone setting
CMD_TZ_RRQ	27	Read time zone setting
CMD_TZ_WRQ	28	Write time zone setting
CMD_ULG_RRQ	29	Read unlocking combination
CMD_ULG_WRQ	30	Write unlocking combination
CMD_UNLOCK	31	unlock
CMD_CLEAR_ACC	32	Recover access control setting as default state
CMD_CLEAR_OPLOG	33	Delete all attendance log in the machine
CMD_OPLOG_RRQ	34	Read management record
CMD_GET_FREE_SIZES	50	Get machine state, such as user record and so on
CMD_ENABLE_CLOCK	57	Enable machine in normal work state
Command character of module:		
CMD_STARTVERIFY	60	Enable machine in verification state
CMD_STARTENROLL	61	Start to enroll a user, enable machine in enrolling user state
CMD_CANCELCAPTURE	62	Enable machine in waiting for command state, refer to CMD_STARTENROLL for detailed information.
CMD_STATE_RRQ	64	Get machine state
CMD_WRITE_LCD	66	Write LCD
CMD_CLEAR_LCD	67	Clear LCD subtitle (clear screen)
CMD_GET_PINWIDTH	69	Get user PIN length
Related command of SMS		
CMD_SMS_WRQ	70	Upload SMS
CMD_SMS_RRQ	71	Download SMS
CMD_DELETE_SMS	72	Delete SMS
CMD_UDATA_WRQ	73	Set user SMS
CMD_DELETE_UDATA	74	Delete user SMS
CMD_DOORSTATE_RRQ	75	Get door state
Related command character of Mifare Card：		
CMD_WRITE_MIFARE	76	Write Mifare card
CMD_READ_MIFARE	77	Read Mifare card
CMD_EMPTY_MIFARE	78	Clear Mifare card
Command character of device		
CMD_GET_TIME	201	Get machine time
CMD_SET_TIME	202	Set machine time
Command character of real-time event:		
CMD_REG_EVENT	500	Register event

EF_ATTLOG	1	Pass real-time verification
EF_FINGER	(1<<1)	Press fingerprint at the real time (return data type sign at the real time)
EF_ENROLLUSER	(1<<2)	Enroll user at the real time
EF_ENROLLFINGER	(1<<3)	Enroll fingerprint at the real time
EF_BUTTON	(1<<4)	Press button at the real time
EF_UNLOCK	(1<<5)	Unlock at the real time
EF_VERIFY	(1<<7)	Verify fingerprint at the real time
EF_FPFTR	(1<<8)	Extract fingerprint feature at the real time
EF_ALARM	(1<<9)	Alarm signal
Related command character returned by machine		
CMD_ACK_OK	2000	Returned value after successful execution
CMD_ACK_ERROR	2001	Returned value after failed execution
CMD_ACK_DATA	2002	Return value
CMD_ACK_RETRY	2003	Registered event occurred
CMD_ACK_REPEAT	2004	
CMD_ACK_UNAUTH	2005	Unauthorized connection
CMD_ACK_UNKNOWN	0xffff	Unknown command
CMD_ACK_ERROR_CMD	0xfffd	Command error
CMD_ACK_ERROR_INIT	0xfffc	/* Not Initialized */
CMD_ACK_ERROR_DATA	0xfffb	

Data (attendance log, fingerprint and so on) type sign:

Command	**Value**	**description**
FCT_ATTLOG	(U8)1	Attendance record
FCT_WORKCODE	(U8)8	WorkCode
FCT_FINGERTMP	(U8)2	Fingerprint data
FCT_OPLOG	(U8)4	Operation log
FCT_USER	(U8)5	User record
FCT_SMS	(U8)6	SMS
FCT_UDATA	(U8)7	User SMS

3.2 Description and explanation of command character

- **CMD_CONNECT**

  This command is used to connect with machine. If it is successful, return CMD_ACK_OK.Send every data on the basis of packet structure.

  If there is password set in the machine, machine will return not connected authorized command after successful execution. Therefore, it is necessary to send connection password to complete the connection. Refer to connection authorized command for detailed information.

- **CMD_EXIT**

  Disconnect

- **CMD_ENABLEDEVICE**

  Enable machine in normal working state. Usually, the peripheral device (keyboard, LCD, sensor) will be screened during communication, and this command is to make the peripheral devices in normal working state.

u **CMD_DISABLEDEVICE**

Screen peripheral keyboard, LCD, and sensor. If the execution is successful, “work” will be displayed on LCD.

u **CMD_CAPTUREFINGER\ CMD_CAPTUREIMAGE**

Sensor captures image.ZEM500（A4，F4+and so on）serial products don’t support this function. The replied data is bitmap original row data. For the sending data amount is large during capturing bitmap, the terminal machine sends a special packet firstly whose front 4 byte storage will receive the total length of data and the next 4 byte will store the size of every packet. Of course, the size of the last packet may be less than or equal to the specified size and every packet size won’t exceed 1Kbytes. When reading some large data (fingerprint template and attendance record), usually data sign will be added to the data to indicate the data type to be read. When capturing non-complete image, the front 4 byte of packet data is set as 500 (to capture part image). If image of user-defined size is needed to be captured, input 4 byte width value from the first idle byte of data part. For the image is proportional, the height transmission can be ignored. Send complete image and the received data (without head) should be 640*180, or the front 4 byte will store bitmap DPI, the next 4 byte will store bitmap width and byte 8—12 will store bitmap height.

u **CMD_TEST_TEMP**

Test whether the fingerprint template is existed. Data part will send some fingerprint template, if the fingerprint exists, return CMD_ACK_OK and data part is returned with user PIN.PIN byte count is determined by user PIN byte supported by machine.

u **CMD_REFRESHDATA**

Refresh data in the machine, mainly completing data synchronization, refreshing fingerprint library and so on. Usually, this command is executed after large data is uploaded.

u **CMD_REFRESHOPTION**

Refresh machine configuration and inform firmware to guide configuration again, such as setting machine ID, baud rate, and time and so on.

u **CMD_TESTVOICE**

Play voice. Play voice according to voice address and length (only support ZEM100, for ZEM100 adopts language chip). The front 2 byte of packet data part separately sends address and length. Index can also be used to play fixed voice.

u **CMD_CHANGE_SPEED**

Change data transmission speed.0 means reducing the speed and 1 means increasing speed.

u **CMD_AUTH**

Connection authorization. If there is communication password set in the machine, it is necessary to send communication password to start verification during connection.

u **CMD_PREPARE_DATA**

Inform the machine to send data (PC→Device), or machine is to send data (PC→Device).Length of data to be sent is to be input in the front 4 byte. The machine will create a buffer to receive data to be sent on the basis of session ID after it receives the command. After the command is executed successfully, use CMD_DATA to send data. When data is sent successfully, CMD_FREE_DATA can be sent to machine to free buffering area. This process is usually used during data recovery, firmware upgrade and other data communications. Of course, after the data is sent, it is necessary to inform the machine what function the data has.

u **CMD_DB_RRQ**

Read all data in the machine. The data is sent in firmware defined structure. (Refer to firmware structure definition for specific structure.) Send the data in large amount (refer to fingerprint bitmap capture command description). The data requested to be sent is decided by the first byte of packet data part. If the first byte is filled with attendance data sign, then all attendance logs can be read.

Notice: All read large data command must follow the above method, such as attendance record, user, fingerprint and so on. For other read large data command, there won't be specific description.

- **CMD_USER_WRQ**

  Upload user information. Except for over-large data transmission, there is user information, fingerprint template, access control privilege and so on. Fill in packet data part in the form of data structure. After data transmission, the machine can return data according to execution.

- **CMD_USERTEMP_RRQ**

  The command can be used to read a user's fingerprint template. Specified user PIN (2Bytes) and fingerprint index (0—9) are needed to be input into packet data part. Data part specifies 1byte data type and read some special data.

- **CMD_USERTEMP_WRQ**

  Upload a user's fingerprint template. Refer to firmware data structure for packet data transmission. Notice: The premise to upload fingerprint template successfully is that the user must exist and the corresponding user must be empty.

- **CMD_OPTIONS_RRQ**

Read some configuration parameter value in the machine. Configuration item is needed to be input in packet data part. Return the configuration value after successfully execution.

- **CMD_OPTIONS_WRQ**

  Set parameter. It is necessary to input configuration item in packet data part.

- **CMD_ATTLOG_RRQ**

Read all attendance logs (refer to related description of read large data). The command with the only function is used to read attendance logs.

- **CMD_CLEAR_DATA**

  Clear some data. If there is no data type specified, delete all data, or delete the specified data.

- **CMD_CLEAR_ATTLOG**

  Delete the attendance record

- **CMD_DELETE_USER**

  Delete some user Input user PIN (2bytes) in packet data part.

- **CMD_DELETE_USERTEMP**

Delete a fingerprint template of user. Input user PIN (2bytes) in the front 2 bytes of packet data part and fingerprint ID (0—9) in the third byte.

- **CMD_CLEAR_ADMIN**

  Clear administrator

- **CMD_USERGRP_RRQ**

Read user subgroup. Operate ZKSoftware access control device (F4, F4+, A6 and so on). Input user PIN (2bytes) in the front 2 bytes of packet data part.

- **CMD_USERGRP_WRQ**

Set user subgroup. Input user PIN (2bytes) in the front 4 bytes of packet data part and group number in the next 4 byte.

- **CMD_USERTZ_RRQ**

  Read time zone used by user. Input user PIN (2bytes) in packet data part. The first 4 byte of returned data is user PIN. And the next 12 bytes are three time zone numbers. Every number occupies 4 bytes.

- **CMD_USERTZ_WRQ**

  Set time zone used by user. Input time zone using count in the first 4 bytes of packet data. Only 3 time zones can be set at present. Therefore, 3 is input. The next 12 bytes are three time zone numbers. Every number occupies 4 bytes.

- **CMD_GRPTZ_RRQ**

  Read group time zone. Refer to read user time zone for the function. Group time zone function is similar to that of user time zone. Therefore, there is no further description.

- **CMD_TZ_RRQ\ CMD_TZ_WRQ**

  Read time zone setting. Input time zone number in the first 4 byte of packet data part. 50 time groups can be supported by access control device at most. Every time zone period is week. The setting format of everyday time zone is 24 hours. For example: 09091616 is the time zone of some day, which means the valid time zone is from 09:09 to 16:16 of the day. The whole time zone takes week as its unit, arraying in everyday time zone. The time zone is started from Sunday.

- **CMD_ULG_RRQ\ CMD_ULG_WRQ**

  Read group unlocking combination setting. Return order number, namely group combination setting. 5 groups and 10 unlocking combination can be support by access control device. The returned combination is separated by ":".And write combination is also separated by ":".Send or receive combination information input in packet data part.

- **CMD_UNLOCK**

  Unlocking command. Inform access control device to unlock the door. Set unlock delay in the first 4 bytes of packet data part.

- **CMD_GET_FREE_SIZES**

  No

- **CMD_ENABLE_CLOCK**

  Set LCD dot (twinkling ':'). 0 means stop twinkling and 1 means start twinkling. After successful execution, the firmware will render LCD.

- **CMD_STARTVERIFY**

  Enable machine in verification state. If user PIN (2bytes) is filled in packet, verifying user will be started and the machine will remind user to press fingerprint. If there is no user PIN sent, the machine will recover to the normal verification state.

- **CMD_STARTENROLL**

  Enable machine in enrollment state. Fill the first 2 bytes with user PIN (2bytes), and fingerprint number in the next 2 bytes. After successful execution, LCD will remind user to press finger and start enrollment. Notice: Before enrollment setting, it is necessary to send CMD_CANCELCAPTURE to enter enrollment state. After enrollment, CMD_STARTVERIFY can be used to recover normal verification state.

- **CMD_STATE_RRQ**

  No

- **CMD_WRITE_LCD**

  The command character will be displayed on LCD. The first 2 bytes of packet data part will send row value to be displayed and the third byte is set as 0. Then the character to be sent will be input in the following bytes. It can be used together with CMD_CLEAR_LCD.

- **CMD_GET_PINWIDTH**

  Obtain user PIN length. Usually the PIN is 5 bits or more than 5 bits.

- **CMD_SMS_WRQ**

  Upload SMSFill in the packet part according to SMS structure. Refer to data structure description.

- **CMD_SMS_RRQ**

  Download SMS. Input number of SMS to be downloaded in the first 2 bytes of packet data part and return SMS structure. Notice: Only machine supporting SMS can support this command (for example A6).

- **CMD_DELETE_SMS**

  Delete some SMS. Input SMS number (2bytes) in packet data part.

- **CMD_UDATA_WRQ**

  Set user SMS Fill packet data part according to user SMU structure.

- **CMD_DELETE_UDATA**

  Delete some SMS of user. Input user SMS data to be deleted according to SMS structure in packet data part.

u **CMD_WRITE_MIFARE**

Inform machine to write Mifare card. Only machine with Mifare card reader can support this function. Fill in the first 4 bytes of packet data part with user PIN and the following 12 bytes with template information. Among the 12 bytes, the first and the second bytes are filled with template length and the third byte is filled with template index (corresponding to a fingerprint of user). The first three bytes cannot be empty and must be with template. 4 fingerprint templates can be written once. The later 9 bytes are filled with length of other three fingerprints and index. The following structure can be referred:

4 bytes | [1-2bytes (length of fingerprint template 1), the third byte (fingerprint index)], [4-6….] | user PIN 4 bytes | fingerprint template information 12 bytes | 1-4 fingerprint template |

u **CMD_GET_TIME**

Get machine time. Return 4 bytes time value filled in packet data part. Time value is user-defined code. The coding mode is as the following:

(year%100）*12*31 + ((mouth - 1)*31 + day - 1)*(24*60*60) + (hour*60 + minutes)*60 + second）. Decoding can be done based on this mode.

u **CMD_REG_EVENT**

Register real-time event. If the event is registered, the machine will send data (user pass verification, press keyboard and so on) to all successful connector at the real time.

3.3 Data structure

The data structure in packet sending and receiving data has been defined and it is the same with that of data structure in firmware. Please refer to the following:

u **user data structure**

Use the following data structure before firmware version is 5.04, aligning in 1 byte.

```
typedef struct _UserOld_{
    U16 PIN;
    U8 Privilege; // privilege, 0---common user, 1—registrar, 2—administrator, 3—super administrator
    char Password[5];
    char Name[8];
}TUserOld, *PUserOld;
```

After Version 5.04, user data structure is as the following, aligning in 1 byte.

```
typedef struct _User_{
    U16 PIN;                // user PIN in machine
    U8 Privilege;       // for 0—7 bit, 000—common user, 001—registrar, 110—administrator, 111—super
administrator, the last bit of privilege byte is 0, which means user is valid (1 means invalid).
```

7	6	5	4	3	2	1	0
0	0	0	0	0	0	1	0

```
                //
                //。
    char Password[5];   // password
    char Name[8];       //   user name
    U8 Card[5];         Card number, used for saving corresponding ID card number
    U8 Group;           Group where user is
    U16 TimeZones;      // time zone and bit sign usable for user
```

```
        U32 PIN2;            // 32 bit user PIN 2
    }TUser, *PUser;
```

Notice: Before version 5.04, machine only supports 5 bits code. In order to support multi-bit user code, new user data structure is defined after version 5.04, namely the later 9 bits code machine. During communication, many places in the front will remind sending 2 bytes for user PIN, namely U16 PIN.   U16 PINs in other places are the same.

- **Fingerprint template structure**

```
    typedef struct _Template_{
        U16 Size; //fingerprint template length
        U16 PIN;   // corresponding to PIN in user data structure
        char FingerID; // fingerprint index
        char Valid;     // fingerprint is valid or invalid           char *Template; //fingerprint template
    }TTemplate, *PTemplate;
```

- **Attendance log structure**

The following is non-extended attendance log structure, namely the attendance log is compressed for storage.

```
    typedef struct _AttLog_{
        Int PIN;   //U16 PIN, user number
        char verified;// verifying method
        time_t time_second; //tiem, time code is user-defined time code.
        char status; // attendance state
    }TAttLog, *PAttLog;
```

Notice: Only read command can be used to read all attendance logs. Attendance log can be compressed in long or short mode. The compressing method (if char *Buffer is being read now, the hand will be at the first byte) is: the first 2 bytes are used for storing user PIN (U 16 PIN), the front three bits of the third byte are used for storing verifying state. The fourth and fifth bits are to store verifying method. The sixth bit is to store short and long time sign. If it is short time format, time value is the last two bits of the third byte and the third byte plus the recent long time value (therefore, time format is stored as the misregistration value of the former long time value). Then decode it according to time coding mode (refer to user-defined coding mode) to get the correct time.

The following are extended attendance log structure

```
    typedef struct _ExtendAttLog_{
        U32 PIN;
        time_t time_second; // here is integrity time
        BYTE status;
        BYTE verified;
        BYTE reserved[2]; // temporarily is useless.
        U32 workcode;
    }TExtendAttLog, *PExtendAttLog;
```

If data storing format is extended mode, read attendance log according to the structure.

- **SMS structure**

```
    typedef struct _SMS_{
        U8 Tag;         // type, public SMS, non-public SMS
        U16 ID;         // data content sign, 0 means the record is invalid.
        U16 ValidMinutes;       // valid minute count, 0=forever
        U16 Reserved;
        U32 StartTime;          //start time
        U8 Content[MAX_SMS_CONTENT_SIZE+1];     // SMS content, MAX_SMS_CONTENT_SIZE=60
    }TSms, *PSms;
```

User short message data structure

```
typedef struct _UData_{
       U16 PIN;                // 0 means invalid record.
       U16 SMSID;              // SMS ID
}TUData, *PUData;
```

u **Management data structure**

```
typedef struct _OPLog_{
              U16 Admin; //administrator number
              BYTE OP;        // operation type
              time_t time_second; // time, complete timeDecode it according to the above user-defined method.
              U16 Users[4]; //User[0], operated user PINUser[1], operating result, 1—succeed, 0--fail
                                   //User[2], User[3] is useless.
}TOPLog, *POPLog;
```

Operating type is as the following:

Value	description
0	power on
1	power off
2	verification fail
3	dismantle machine
4	enter menu
5	modify setting
6	register fingerprint backup
7	add password
8	enroll HID card
9	delete user.
10	delete fingerprint
11	delete password
12	delete RF card
13	clear data
14	create MF card
15	enroll MF card
16	register MF card
17	delete MFcard registration
18	clear MF card content
19	move the registered data to the card
20	copy the data in card to offline fingerprint sensor
21	set offline fingerprint sensor time
22	reset factory
23	delete attendance log (out and in)
24	clear administrator privilege
25	modify access control group setting
26	modify user access control setting
27	modify access control time zone
28	modify unlocking combination setting
29	unlocking
30	enroll user

4 Real-time event

When the event is registered in the machine, the machine will send related information to the connector at the real time. When CMD_REG_EVENT from machine is received, the related information can be analyzed according to different data type. In the packet, SessionID indicate event type. Refer to the following: related information will be input in data part of the received packet. Session ID will be changed after real-time news is received. CMD_ACK_OK

- **EF_ATTLOG**

  The following is the meaning of data part:

  1—2 byte: user PIN, the high 4 bits of the fourth byte: whether it is valid, low 4 bits: attendance state, the third byte: verifying method, the following 6 bytes is: second/minute/hour/day/monthyear. Year is misregistration value based on year 2000.

- **EF_ENROLLUSER**

  When a user is enrolled, the machine returns the data at the real time. 2 bytes user PIN can be got by data part of packet.

- **EF_BUTTON**

  Return pressed button value.

- **EF_VERIFY**

  Return user PIN.

- **EF_FPFTR**

  Return fraction upon fingerprint verification.

- **EF_ALARM**

  The length of returned data part is 4. the first byte is 55:dismantle alarm, the first byte is 53:out-go button.

  The length of returned data part is 8. the first byte is 54: open and close the door.

  The length of returned data part is 12. the front 2 bytes is 0Xffff: duress alarm. The seventh and the eighth bytes are alarm type. The fifth and sixth bytes are duress fingerprint number. Byte 9—12 are verification method.