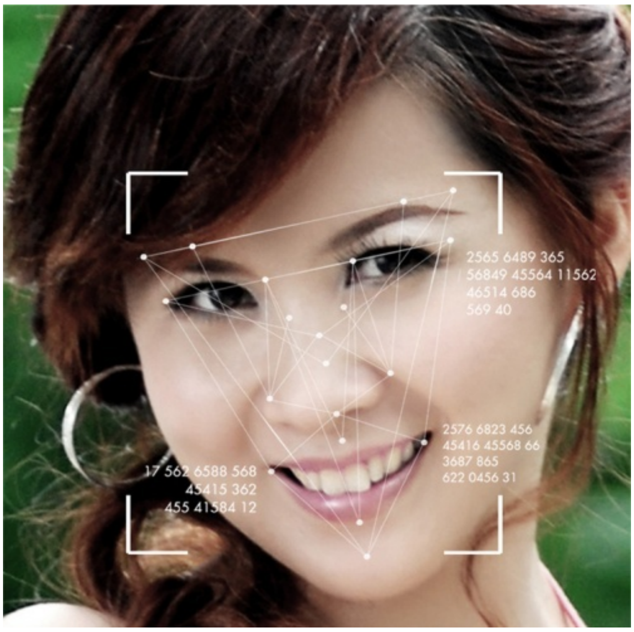
Hanvon FaceID Facial Recognition Device API Specification



Hanwang Technology Co.,Ltd

Version 3.3

July, 2012

About Hanwang Technology Co., Ltd.

Hanwang Technology Co., Ltd is the most important intelligent recognition solution provider in the world, and was founded in 1998. Nowadays, we have over 600 employees, and more than 300 R&D members. We have hosted in a 23,000m^2 building called Hanwang Building which locates at Zhongguancun Software Park in Beijing.

Vision: Enrich people’s life with good taste and efficiency by digital technology

Mission: Make customers/partners/shareholders/employees live easier and happier

Core Competence: Intelligence Interaction technologies & products, which used Pattern Recognition as the core

Statement

Thanks for your interest and cooperation with Hanwang Technology Co., Ltd. Please note below terms of use this Software Development Kits:

Follow up all instruction during your development, we do not be responsible for any data destroyed or lost which caused by human element or program you are developing or developed.

Table of Contents

1 SDK Overview 5

2 SDK Installation 6

2.1 How to Install 6

2.2 Quick Start 6

2.3 API 6

2.3.1 Client Side Interface 6

2.3.2 Server Side Interface 8

3 Overview of Communication Protocol 10

3.1 Communication with Facial Recognition Device 10

3.2 Grammar and Key Word 10

3.3 Naming Principles of Key Word 11

4 Key Word List and Definition 12

4.1 Command Word 12

4.2 Control Word 15

5 Description of Command Word 23

5.1 Employee Management Command 23

5.2 Record Management Command 27

5.3 Device Management Command 28

5.4 Image Management Command 45

Appendix - Device Function List 45

1 SDK Overview

This Software Development Kits (SDK) is targeted to provide flexible solution for Software or Solution integrators based on Hanwang Facial Recognition technology. This SDK is composed of development documentation, development Demo, development tools. Following functions could be realized:

- Employee Management
- Record Management
- Device Management
- Image Management

Consist of this SDK:

- DYNAMIC LINK LIBRARY: HwDevComm.dll and Hdcp_Utils.dll
- Documentation: SDK manual
- Development Demo: programming language C#, VC, Delphi and VB

This SDK could assort with Hanwang Face ID device:
FK603, FK605, FK610, FK628, F7, F710, C220, C230, C330, E350, E350A, E352, S7150A, F810.

Operating System compatible:
Windows 2000, Windows XP, Windows 2003, Windows Vista, Windows 7.

Hardware Platform compatible (at least):

CPU：PIII 1.6GHz；Memory：512M；Hard Disk：2GB

2 SDK Installation

2.1 How to Install

- Please copy this SDK to local disk, open Demo folder then find related folder;
- Copy HwDevComm.dll and Hdcp_Utils.dll to the related folder;
- Start related program tools to run.

Related parameter, such as connection method, device type, and IP address, could be defined with the interface of the Demo

4 kinds of demo source codes, C#, VC, DELPI and VB, are provided and kept in each separated folders in DEMO Directory.

2.2 Quick Start

The aim of this guide is to describe how to use dynamic link library HwDevComm.dll. Users can achieve different functions with different parameters via calling the interface.

Hdcp_Utils.dll is an accessory library file which used for the switch of JPG file and BASE64 code that help programmer to handle the image data.

It is no need to understand the details of the communication protocol as the protocol have been sealed in the SDK. The address of the API will be gained after loading the HwDevComm.dll, and then sends the parameters to related functions.

2.3 API

HwDevComm.dll provides 2 working mode: client side and server side. In client side mode, PCs with running dll regards as client to send request to FaceID devices, and wait for the responding; in server side mode, PCs with running dll regards as server to receive data from devices

2.3.1 Client Side Interface

In client side mode, HwDevComm.dll provides 2 interfaces, as the following frames shown:

Client Side API 1	Definition:
HwDev_Execute	Execute commands, such as management commands, record commands, and so on.

Declaration	Definition:
	Int HwDev_Execute(char * pDevInfoBuf, unsigned long nDevInfoLen, char * pSendBuf, unsigned long nSendLen, char ** pRecvBuf, unsigned long * pRecvLen, FuncTotalDoneTp pFuncTotalDone)
Parameter	**Definition:**
pDevInfoBuf	The information pointer of facial recognition device. The device information described as below: DeviceInfo(dev_id = "1" comm_type = "ip" ip_adress = "172.16.1.15" commukey = "")
nDevInfoLen	The length of device information buffer.
pSendBuf	The pointer of sending buffer. Please place one command in the butter area every time. If many commands are placed in the butter area at the same time, they will be invoked in order. If one command of them executes failed, the other following commands will not be affected.
nSendLen	The length of sending buffer. If nSendLen=0, the HwDevComm.dll will only receive, but not send anything.
pRecvBuf	The pointer of the receiving buffer. It places execution results from various commands. This buffer is applied by HwDev_Execute interface, and released by HwDev_Finish interface.
pRecvLen	The length of receiving buffer.
pFuncTotalDone	The pointer of callback functions; send the execution process back to the caller. Among of them, the definition of PFuncTotalDoneTp will be: typedef int (CALLBACK FuncTotalDoneTp) (unsigned long nTotal, unsigned long nDone).
Return Value	**Definition:**
0	Successful
-1	Failed
Others	Remain

Client Side API 2	**Definition:**
HwDev_Finish	Release data storage memory.
Declaration	**Definition:**
	void HwDev_Finish (char ** pRecvBuf)
Parameter	**Definition:**
pRecvBuf	The pointer of the receiving buffer. It places execution results from various commands; This buffer is applied by HwDev_Execute interface, and is released by HwDev_Finished interface.

2.3.2 Server Side Interface

In server side mode, HwDevComm.dll provides 1 interface, as the following frames shown:

Server Side API	**Definition:**
HwDev_Server	To startup server side or close specific monitoring service. It is to receive uploaded information from FaceID device initiatively.
Declaration	**Definition:**
	int HwDev_Server(int nSwith, char *pServerInfoBuf, unsigned long nServerInfoLen, FuncProcessData pFuncProcessData)
Parameter	**Definition:**
nSwith	It is to indicate functionality of executing function: 1 - to start a port monitoring service; 0 - to close a running monitoring service; Others - remain value.
pServerInfoBuf	String address of information for starting service in local server. The format is: ServerInfo(comm_type = "ip" ip_address = "173.16.4.253" ip_port = "9923") comm_type – connecting method, only support network connection currently;

	ip_address – IP address of local server; ip_port – specific binding port. If this port has been used already, then return false value.
nServerInfoLen	String length of server information.
pFuncProcessData	Reutrn function pointer. FuncProcessData has been defined as: typedef int (CALLBACK *FuncProcessDate)(char * pDevInfo , unsigned long devInfoLen, char *pRecvBuf, unsigned long RecvLen); In this function, user can define details method. Parameter: pDevInfo – a string type which contains device information (IP address). devInfoLen – string length of device information. pRecvBuf – information string uploaded by device, which contains complete information. It may be 1 or more strings. pRecvLen string length of uploaded information. This function can be rewritten.
Return Value	**Definition:**
0	To start or close specific port monitoring service.
-1	Repeat to open SERVER service for the same port.
-2	Function parameter inputting error.
-3	Fail to open monitoring communication link.
-4	Allocate memory error for opening monitoring service.
-5	Fail to close port monitoring socket.
-6	The connection is close or not exists.
Other	Remain value

3 Overview of Communication Protocol

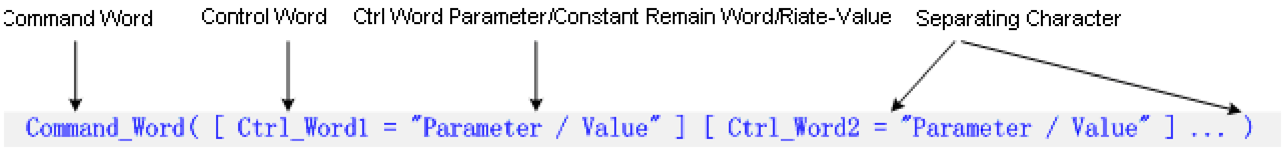
3.1 Communication with Facial Recognition Device

Preparation: connect the device to PC with a network cable.

Setup Socket connection: Setup Socket in PC and connect to Device (the default status of the device is “Accept”, and the monitor port is 9922).

Remove Socket connection: Close the Socket in PC.

3.2 Grammar and Key Word



```
Command_Word( [ Ctrl_Word1 = "Parameter / Value" ] [ Ctrl_Word2 = "Parameter / Value" ] ... )
```

Figure 3.1

The grammar of this protocol will be shown in Figure 3.1, it is comprised with 4 types of key word, as described in Frame 3.1:

Key words name	**Definition and purpose**
Command word	To express what this operation will do. (e.g. GetEmployeeID express return all Employee ID from the device).
Control word	As the parameter of the command word, to express what the parameter is. (e.g. name express the employee’s name)
Constant key word	As the parameter of the command word, it usually follows the control word, to express what the parameter value is. (e.g. language="chs" express the device language setting is Simplify Chinese).
Separating character	To separate key words.

Frame 3.1

Except the separator, the other key words are combined with number, letters and underline, they are 32 letters in maximum and case sensitive. Space, Tab, CR can be inserted between each key word, and ignore the procedure.

3.3 Naming Principles of Key Word

Key word name	Naming Principle
Command word	Pascal naming rule, means the name is combined with one or more words, the initial letter of each words will be capitalized, and the other letters will be lower case.
Control word	Uses “lower case letter and underline” method, all letters are lower case and the words are separated by “_”, e.g. face_data.
Constant key word	Include “lower case letter and underline” method.

4 Key Word List and Definition

4.1 Command Word

Name	**Definition**
Employee management command	
GetEmployeeID	Return all employees ID from the device.
GetEmployee	Return specified employee information from the device according to employee ID, only one employee each time.
SetEmployee	Set employee information to the device, only one employee each time.
DeleteEmployee	Delete a specified employee from the device according to employee ID, many employees can be deleted once.
DeleteAllEmployee	Delete all employees from the device.
SetNameTable	This command updates the “id-name” list in the device.
AddNameTable	Add “id-name” list
DetectEmployeeData	Detect the device whether contains user data.
GetTimeZone	Get time zone from device
SetTimeZone	Set the information for the device
DeleteTimeZone	Delete a time zone from the device
Record management command	
GetRecord	Return records from the device according to given time range.
DeleteAllRecord	Delete all records stored in the device.
Device management command	
InitDevice	Initialize device to default setting.
InitDeviceAdmin	Initialize administrator setting to default setting.
GetDeviceInfo	Return configuration and status information from the device.

SetDeviceInfo	Set configuration and status information to the device.
UpdateFirmware	Upgrade firmware
GetNetInfo	Get device network parameter
SetNetInfo	Set device network parameter
GetMAC_SN	Get MAC address and SN
DetectDevice	Detect device whether exist
RestartDevice	Restart device
SetDayLight	Set Daylight Saving Time (DST)
GetDayLight	Get Daylight Saving Time (DST) setting
SetWorkCode	Set work code information.
GetWorkCode	Return work code information.
ClearWorkCode	Clear work code information.
DeleteWorkCode	Delete specific work code information.
SetWorkStatus	Set work status information.
GetWorkStatus	Return work status information.
ClearWorkStatus	Clears work status information.
DeleteWorkStatus	Delete specific work status information.
GetManagerID	Get all admin. IDs
GetManager	Get all admin. information by admin. IDs
SetManager	Set admin. information
DeleteManager	Delete specific admin.
GetAttendanceInterval	Get device interval period
SetAttendanceInterval	Set device interval period
OpenDoor	Remote to open door
ConfirmPassword	Comfirm password
SetRelayTime	Set relay delay time
GetRemoveAlarm	Get status for removal alarm
SetRemoveAlarm	Set status for removal alarm
GetFRAlarm	Get status for false recognition alarm
SetFRAlarm	Set status for false recognition alarm

GetMagneAlarm	Get status for door magnet sensor
SetMagneAlarm	Set status for door magnet sensor
GetDateTime	Get device date
SetDateTime	Set device date
GetVolume	Get device volume status
SetVolume	Set device volume status
GetWiegandOut	Get Wiegand output parameter
SetWiegandOut	Set Wiegand output parameter
GetBell	Get time alarm bell status
SetBell	Set time alarm bell status
GetOnOffSchedule	Get power on/off schedule status
SetOnOffSchedule	Set power on/off schedule status
GetClientStatus	Get records auto uploading status
SetClientStatus	Set records auto uploading status
SetAutoResetTime	Reset the device at the certain time
SDFormat()	Format the SD card
Image management command	
GetPictureName	Return image file name list according to time of this file and whether the images were recognized successfully.
GetPicture	Return image files according to the name of image file. These image files are encoded by base-64, and uses JPG format.
Result command	
Return	Return value mark. It's an executing result of a command. The structure is: **Return(result="success / failed" [Ctrl_Word = "Parameter / Value"] ...)**
Wait	Wait mark. It means the device is running a command, and need to wait a moment (the time is given by the control word "wait_time", and uses "second" as a unit),to return the result. The structure is:

	Wait(wait_time = "10") //this means that need to wait for 10 seconds This command will be used in the following situation: Some actions need a long time to operate (e.g. initializes the device), at this moment, the other waiting side will know how much time the correct result will be received through “wait_time”. Usually, if the executing command can return the result within 5 seconds, then it is not necessary to use “Wait”.
	Caution: the default time of receiving data for this protocol is 5 seconds. If a command does not return any value within 5 seconds, and it does not indicate a waiting time by using “Wait” command as well, then the command will get an overtime exception and exit.

4.2 Control Word

<table>
<tr><th>Name</th><th>Definition</th></tr>
<tr><td><b>The description of the command executive result</b></td><td></td></tr>
<tr><td>result</td><td>Mark the executive result of a command, the possible constant values will be:
<table>
<tr><th>Name</th><th>Definition</th></tr>
<tr><td>success</td><td>Successful</td></tr>
<tr><td>failed</td><td>Failed</td></tr>
</table></td></tr>
<tr><td>reason</td><td>The explanation of the executive result, the possible constant values will be:
<table>
<tr><th>Name</th><th>Definition</th></tr>
<tr><td>unknown command</td><td>Unrecognized command</td></tr>
<tr><td>bad parameter</td><td>Wrong parameter</td></tr>
</table></td></tr>
</table>

<table>
<tr><td></td><td><table>
<tr><td>device busy</td><td>Device is busy</td></tr>
<tr><td>employee overflow</td><td>For command SetEmployee, the number of employee exceeds the maximum number.</td></tr>
<tr><td>unknown id</td><td>For command DeleteEmployee, the certain employee ID doesn’t exist.</td></tr>
</table></td></tr>
<tr><td>notify</td><td>The notice message for the executive result, it’s usually caused by the unknown parameter in the command. The possible constant value will be:<table>
<tr><th>Name</th><th>Definition</th></tr>
<tr><td>unknown parameter</td><td>Unrecognized parameter</td></tr>
</table></td></tr>
<tr><td></td><td><b>Caution: The executive result must begin with result/reason/notify, and the order must be result/reason/notify.</b></td></tr>
<tr><td>wait_time</td><td>To indicate how much time need to wait for a device to finish its operation. Use “second” as a unit.</td></tr>
<tr><td></td><td></td></tr>
<tr><td><b>Public constant</b></td><td></td></tr>
<tr><td>Check method constant</td><td>To definite the value of control word for time attendance and access control, the possible constant values will be:<table>
<tr><th>Name</th><th>Definition</th></tr>
<tr><td>face</td><td>Face recognition</td></tr>
<tr><td>num&face</td><td>Pin and Face recognition</td></tr>
<tr><td>card</td><td>Card only recognition</td></tr>
<tr><td>card&photo</td><td>Card with photo taking recognition</td></tr>
<tr><td>card&face</td><td>Card and Face recognition</td></tr>
</table></td></tr>
<tr><td></td><td></td></tr>
<tr><td>Logic operation constant</td><td>To combine several check methods constant, the possible constant values will be:<table>
<tr><th>Name</th><th>Definition</th></tr>
</table></td></tr>
</table>

<table>
<tr><td></td><td><table><tr><td>&</td><td>AND</td></tr><tr><td>|</td><td>OR</td></tr></table></td></tr>
<tr><td></td><td></td></tr>
<tr><td><b>The description of employee information</b></td><td></td></tr>
<tr><td>id</td><td>User ID number (ASCII, i.e. id=“9997”)
Note: for each employee record, it must start with id parameter</td></tr>
<tr><td>name</td><td>Employee name (e.g. name=“Alice”)</td></tr>
<tr><td>card_num</td><td>Employee card number (e.g. card_num=“0Xffffffff”)</td></tr>
<tr><td>register_type</td><td>register_type = "face": represents face
register_type = "card": represents card with photo taking</td></tr>
<tr><td>enterType</td><td>enterType="0X01": represents face
enterType="0X02": represents card&photo
enterType="0X04": represents card&face
enterType="0X08": represents card</td></tr>
<tr><td>authority</td><td>authority="0X0": authority of attendance and access control;
authority="0X55": authority of attendance only;
authority="0XAA": authority of access control only;
authority="0XFF": authority for others.</td></tr>
<tr><td>check_type</td><td>Attendance method, the possible constant value is <b>Check method constant</b>. e.g. check_type="card&face". This is for Hanvon internal use. The value for this parameter is defined by “enter_type”.</td></tr>
<tr><td>opendoor_type</td><td>Open door method, the possible constant value is <b>Check method constant</b>. e.g. opendoor_type="card&face". This is for Hanvon internal use. The value for this parameter is defined by “enter_type”.</td></tr>
<tr><td>face_data</td><td>Face features data, binary data encoded by Base-64. “face_data” contains 876 characters</td></tr>
<tr><td>head_photo</td><td>ID photo, binary data encoded by Base-64</td></tr>
</table>

<table>
<tr><td></td><td>Use ““ to bracket data</td></tr>
<tr><td>Note:</td><td>For card user, it does not include face_data and head_photo items.</td></tr>
<tr><td></td><td></td></tr>
<tr><td><b>The description of the attendance records</b></td><td></td></tr>
<tr><td>time</td><td>Attendance record time, format: YYYY-MM-DD hh:mm:ss
Note: attendance record must begin with “time” parameter
For example: time=“2008-07-15 10:05:49”</td></tr>
<tr><td>id</td><td>User ID, for example: id=“9997”</td></tr>
<tr><td>Name</td><td>User name, for example: name=“Alice”</td></tr>
<tr><td>card_src</td><td>Source of clock time, the possible constant values will be:
<table>
<tr><th>Name</th><th>Definition</th></tr>
<tr><td>from_door</td><td>Valid logs for opening door</td></tr>
<tr><td>from_check</td><td>Valid logs for attendance</td></tr>
<tr><td></td><td>Note: if valid logs for both opening door and attendance, then:
card_src=”from_door&from_check”</td></tr>
</table></td></tr>
<tr><td>type</td><td>Attendance method, the possible constant value is <b>Check method constant.</b>
<table>
<tr><th>Name</th><th>Definition</th></tr>
<tr><td>Face</td><td>Face method of attendance</td></tr>
<tr><td>Photo</td><td>Photo method of attendance</td></tr>
</table></td></tr>
<tr><td>card_type</td><td>Attribute of clock time, the possible constant values will be:
<table>
<tr><th>Name</th><th>Definition</th></tr>
<tr><td>cardtype_normal</td><td>Normal card</td></tr>
<tr><td>cardtype_on</td><td>On duty</td></tr>
<tr><td>cardtype_off</td><td>Off duty</td></tr>
<tr><td>cardtype_addon</td><td>Work overtime sign in</td></tr>
<tr><td>cardtype_addoff</td><td>Work overtime sign off</td></tr>
</table></td></tr>
</table>

<table>
<tr><td></td><td><table>
<tr><td>cardtype_out</td><td>leave office</td></tr>
<tr><td>cardtype_back</td><td>back to office</td></tr>
</table></td></tr>
<tr><td>photo</td><td>For photo method of attendance, this control word is followed by a photo which the format is jpeg and encoded by base-64.
For example: photo="…"</td></tr>
<tr><td></td><td></td></tr>
<tr><td><b>The description of the configuration of facial recognition device</b></td><td></td></tr>
<tr><td>time</td><td>Time in the device</td></tr>
<tr><td>language</td><td>Language in the device, the possible constant values will be:
<table>
<tr><th>Name</th><th>Definition</th></tr>
<tr><td>chs</td><td>Simplified Chinese</td></tr>
<tr><td>cht</td><td>Traditional Chinese</td></tr>
<tr><td>enu</td><td>American English</td></tr>
<tr><td>jpn</td><td>Japanese</td></tr>
<tr><td>kor</td><td>Korean</td></tr>
</table></td></tr>
<tr><td>volume</td><td>Volume of the device, the possible constant values will be:
<table>
<tr><th>Name</th><th>Definition</th></tr>
<tr><td>0</td><td>Mute</td></tr>
<tr><td>1</td><td>Low</td></tr>
<tr><td>2</td><td>Middle</td></tr>
<tr><td>3</td><td>High</td></tr>
</table></td></tr>
<tr><td>dev_id</td><td>Device ID</td></tr>
<tr><td>edition</td><td>The software version of device</td></tr>
<tr><td>wiegen</td><td>Wiegend protocol.</td></tr>
<tr><td>ip\gateway\netmask\mac\hostip\commukey</td><td>Network information</td></tr>
<tr><td>mac</td><td>MAC address</td></tr>
<tr><td>sn</td><td>Serial number</td></tr>
</table>

<table>
<tr><td>erasepage</td><td>Flash erasing scope</td></tr>
<tr><td>writepage</td><td>Flash writing scope</td></tr>
<tr><td>holdpage</td><td>Flash remain scope</td></tr>
<tr><td>md5check</td><td>MD5 secret key</td></tr>
<tr><td>style</td><td>Date format, constant values are:
<table>
<tr><th>Name</th><th>Definition</th></tr>
<tr><td>1</td><td>YYYY-MM-DD</td></tr>
<tr><td>2</td><td>YYYY/MM/DD</td></tr>
<tr><td>3</td><td>YYYY.MM.DD</td></tr>
<tr><td>4</td><td>MM-DD-YYYY</td></tr>
<tr><td>5</td><td>MM/DD/YYYY</td></tr>
<tr><td>6</td><td>MM.DD.YYYY</td></tr>
<tr><td>7</td><td>DD-MM-YYYY</td></tr>
<tr><td>8</td><td>DD/MM/YYYY</td></tr>
<tr><td>9</td><td>DD.MM.YYYY</td></tr>
</table></td></tr>
<tr><td>pattern</td><td>Wiegand format, constant values are:
<table>
<tr><th>Name</th><th>Definition</th></tr>
<tr><td>w26</td><td>26-bit without site code</td></tr>
<tr><td>w26_site</td><td>26-bit with site code</td></tr>
<tr><td>w34</td><td>34-bit without site code</td></tr>
<tr><td>w34_site</td><td>34-bit with site code</td></tr>
<tr><td>customize</td><td>Customized Wiegand format</td></tr>
</table></td></tr>
<tr><td>pulse_width</td><td>Pulse signal width. Range is within 20-800μs, default value is 100μs</td></tr>
<tr><td>interval</td><td>Pulse signal cycle. Range is within 200-20000μs, default value is 1000μs.</td></tr>
<tr><td>content</td><td>Wiegand output content, constant values are:
<table>
<tr><th>Name</th><th>Definition</th></tr>
<tr><td>id</td><td>Output content is User ID</td></tr>
<tr><td>card</td><td>Output content is Card Number</td></tr>
</table></td></tr>
</table>

site_code	Site code
oem_bit	OEM bit
oem_value	OEM value
area_bit	Site code bit
area_value	Site code value
card_bit	Card number bit
even_start	Start value of domain for even parity bit
even_end	End value of domain for even parity bit
odd_start	Start value of domain for odd parity bit
odd_end	End value of domain for odd parity bit
bell_numbe	Bell number
bell_sound	Bell sound
alarm_time	Bell duration
bell_times	Bell times
turn_on_status	Device power on status
turn_on_time	Device power on time
turn_off_status	Device power off status
turn_off_time	Device power off time
fr_times	Times of false recognition
fr_period	Alarm period of false recognition
magne_delay	Delay time for door sensor alarm
magne _period	Alarm period for door sensor
max_managernum	Maximum number of administrators
managernum	Current number of administrators
max_faceregist	Maximum number of face users
real_faceregist	Current number of face users
max_facerecord	Maximum record number of face users
real_facerecord	Current record number of face users
max_cardregist	Maximum number of card users

<table>
<tr><td>real_cardregist</td><td>Current number of card users</td></tr>
<tr><td>max_photorecord</td><td>Maximum number of photos</td></tr>
<tr><td>real_photorecord</td><td>Current number of photos</td></tr>
<tr><td>format</td><td>Daylight Saving Time format, constant values are:
<table>
<tr><th>Name</th><th>Definition</th></tr>
<tr><td>month_date</td><td>Month-Date format</td></tr>
<tr><td>month_week</td><td>Month-Week-Day format</td></tr>
</table></td></tr>
<tr><td></td><td></td></tr>
<tr><td><b>Other control information</b></td><td></td></tr>
<tr><td>total</td><td>Total number. Usually it is used to mark number of return results, when there are multiple return results. This key word must appear before multiple return results described.</td></tr>
<tr><td>start_time</td><td>Start time, format: YYYY-MM-DD hh:mm:ss</td></tr>
<tr><td>end_time</td><td>End time, format: YYYY-MM-DD hh:mm:ss</td></tr>
</table>

If there are not mentioned field names appear in data transmission, they are for internal use only, and user can ignore these field names.

5 Description of Command Word

5.1 Employee Management Command

Command	**GetEmployeeID()**
Successful reply	Return(result="success" total="100" id="11" id="109" ...)
Failed reply	Return(result="failed")
Caution	For S7150A model, the return result of this function contains register_type, like: Return(result="success" total="100" id="11" register_type="face" id="109" register_type="face"...)

Command	**GetEmployee(id="120")**

Successful reply	Return(result="success"
	id="120"
	name="John"
	calid=""
	card_num=" 0X4c2c3801"
	authority="0X0"
	check_type="face&card"
	opendoor_type="face&card"
	face_data="ADFASERQERERTYSDFGHSDFGADSF..."
	......
	face_data="HASRTTYHRTAEFASDFQEQAFf...")
	There are total 18 sets of BASE64 encoding face templates data (face_data); "face_data" contains 876 characters.

Failed reply	Return(result="failed" reason="unknown id")
Caution	Can only get one ID each time. The value of string "calid" is calculated according to some rules, user can ignore it. The same explanation uses for the following cases.

	For S7150A model, the return result of this function contains register_type and enterType

Command	**SetEmployee(id="1009"** **name="John"** **calid=""** **card_num=" 0X4c2c3801"** **authority="0X0"** **check_type="face"** **opendoor_type="face"** **face_data="ADFASERQERERTYSDFGHSDFGADSF..."** **......** **face_data="HASRTTYHRTAEFASDFQEQAFf...")** (base-64 encode)
Successful reply	Return(result="success")
Failed reply	Return(result="failed" reason="employee overflow")
Caution	For S7150A model, this function needs to set "register_type" and "photo" parameters; For FK628 model, this function needs to set "photo" parameter; For F810 model, this function needs to set "enter_type" and parameter. Values "face", "card&face", "card" and "card&photo" in "enter_type" parameter are not interchangeable to each other.

Command	**DeleteEmployee(id="100")**
Successful reply	Return(result="success")
Failed reply	Return(result="failed" reason="unknown id")

Command	**DeleteAllEmployee()**
Successful reply	Return(result="success")
Failed reply	Return(result="failed" reason="FAILED REASON")

Command	**SetNameTable(120="John" 88="Tom" 192="Mary" 1290="Alice"...)**
Successful reply	Return(result="success")
Failed reply	Return(result="failed" reason="FAILED REASON")
Caution	This command will update the “id-name” user list, and will overwrite the existing names in the list.
	The old list will be overwritten by the new list. Note: this command will not modify id and name in employee data.

Command	**AddNameTable (120="John" 88="Tom" 192="Mary" 1290="Alice"...)**
Successful reply	Return(result="success")
Failed reply	Return(result="failed" reason="FAILED REASON")
Caution	This command will update the “id-name” user list, and new id and name will be appended in the list without overwrite whole user list.

Command	**DetectEmployeeData ()**
Successful reply	Return(result="success")
Failed reply	Return(result="failed" reason="FAILED REASON")
Caution	If return “success”, then represents the device contains users’ data; if return “failed”, then represents the device does not contain users’ data, and return failed reason.

Command	GetTimeZone( )
Meaning	Get all the time zone information of the device

Successful reply	Return(result="success"
	id="1" name="Time Zone" Sun.="00:00-23:59" Mon.="00:00-23:59" ... Sat.:= "00:00-23:59"
	id="2" name=" Time Zone" Sun.="00:00-23:59" Mon.="00:00-23:59" ... Sat.:= "00:00-23:59"
	...

	id="50" name=" Time Zone" Sun.="00:00-23:59" Mon.="00:00-23:59" ...
	Sat.:= "00:00-23:59"
	)

Failed reply	Return(result="failed" reason="FAILED REASON")
Parameter instruction	Id: time zone code, range:[1-50] Name: time zone name, all are named as Time Zone, and it could not be modified. Sun.: time zone setting for Sunday 00:00: beginning time 23:59: ending time Mon.: time zone setting for Monday … Sat …
Caution	When the id=1,the time zone items could not be modified, from Monday to Sunday it is a constant value:00:00 to 23:59. Id equals from 2 to 50,users can modified the items

Command	SetTimeZone(id="2" name="Time Zone" Sun.="02:00-08:00" Mon.="02:00-17:00" Tues.="02:00-17:00"...)
Meanding	Set the information for the time zone
Successful reply	Return(result="success")
Failed reply	Return(result="failed" reason="FAILED REASON")
Parameter instruction	The meaning of the parameter is in accordance with the GetTimeZone()'
Caution	When the id=1,the time zone items could not be modified, from Monday to Sunday it is a constant value:00:00 to 23:59. Id equals from2 to 50,users can modified the items If user use the command as :SetTimeZone(id="1" …) , Then it will

	"Return(result=\"failed\" reason=\"bad parameter\")； Users could only set the items from id=2 to id=50.

Command	DeleteTimeZone(id="2")
Meaning	Delete the information of the time zone item.
Successful reply	Return(result="success")
Failed reply	Return(result="failed" reason="unknown id")
Parameter instruction	Id : time zone code Could delete id from id=2 to id=50, id=1 could not be deleted. If DeleteTimeZone(id=" 1"), then it will return failed reply.
Caution	If id is from 2-50.then it will return successful reply，means it delete the time zone successfully; If id is out of 2-50，then it will return failed reply，means delete failed. Id=1,defined by the system，and it could not be deleted.

5.2 Record Management Command

Command	**GetRecord(start_time="2009-1-1 0:0:0" end_time="2009-11-30 24:00:00")**

Successful reply	Return(result="success" dev_id="1" total="100"
	time="2009-1-10 15:40:13" id="120" name="John" workcode="Engineer" status="1" card_src="from_door" card_src="from_check"
	…
	time="2009-1-10 19:45:00" id="128" name="Roger" workcode="Staff" status="2" card_src="from_door" card_src="from_check"

	photo="SDFQWERASDFAESRASDF...") (Base64 encode)
Failed reply	Return(result="failed")
Caution	If there isn’t start_time, all records before end_time will be read; If there isn’t end_time, all records after start_time will be read; If there aren’t start_time and end_time, all records will be read.
	start_time <= valid time period <= end_time
	Use the control word “time” as a separator to separate each record when receiving records.
	Note: The output result may have a slight difference for each model of device;
	For S7150A model, this function includes enterType parameter in return value

Command	**DeleteAllRecord()**
Successful reply	Return(result="success")
Failed reply	Return(result="failed")
Caution	This function will delete all records.
	For S7150A model, the format of this function is: DeleteAllRecord(time="2008-08-01 0:0:0"), it allows to delete all records before indicated date.

5.3 Device Management Command

Command	**InitDevice()**
Successful reply	Return(result="success")
Failed reply	Return(result="failed")
Caution	It will clear all user data and record data, and device will recover to factory default setting status (include: users, administrators, records, and user setting data)

	After all data are deleted, the device will reset.

Command	**InitDeviceAdmin()**
Successful reply	Return(result="success")
Failed reply	Return(result="failed")
Caution	It will delete all administrators' data from device.

Command	**SDFormat()**
Successful reply	Return(result="success")
Failed reply	Return(result="failed")
Caution	It will format the SD card from device.

Command	**SetDeviceInfo(time="2009-7-1 12:39:40" week="3")**
Successful reply	Return(result="success")
Failed reply	Return(result="failed")
Caution	Value of "week" is from 1 to 7. 1-Monday, 2-Tuesday, 3-Wednesday, 4-Thursday, 5-Friday, 6-Saturday 7-Sunday

Command	**GetDeviceInfo()**

Successful reply	Return(result="success"
	dev_id="1"
	time="2006-1-10 10:12:20"
	volume="1"
	edition="2.031.001"
	weigen= 0
	ip="172.16.2.209"
	gateway="172.16.1.251"
	netmask="255.255.252.0"
	mac= ""

	)
Failed reply	Return(result="failed")
Caution	For S7150 model, the return value also includes: max_managernum, managernum, max_faceregist, real_faceregist, max_facerecord, real_facerecord, max_cardregist, real_cardregist, max_photorecord, real_photorecord. For F810 model, the return value also includes: Return(result="success" dev_id="1" edition="2.000.056" max_managernum="8" managernum="0" max_faceregist="500" real_faceregist="499" max_facerecord="150000" real_facerecord="150000" max_cardregist="5000" real_cardregist="5000" max_photorecord="70000" real_photorecord="8430426") For C220, C230, E350 and E350A model, the return value also includes: max_managernum, managernum, max_faceregist, real_faceregist, max_facerecord, real_facerecord.

Command	**UpdateFirmware(erasepage="128" writepage="52" md5="f1f86766b307f4f7289dace37a91428d" SN="6710710030000170")**
Successful reply	Return(result="success")
Failed reply	Return(result="failed")
Caution	If upgrade binding with SN, then SN=”Device SN Value”; otherwise, SN value is null.

Command	**GetNetInfo()**
Successful reply	Return(result="success" ip="172.16.2.209" netmask="255.255.252.0" gateway="172.16.1.251" commukey=”123456”)

Failed reply	Return(result="failed" reason="FAILED REASON")
Caution	Device default value are: IP: 192.168.0.2; MASK: 255.255.255.0; Gateway IP: 192.168.0.1

Command	**SetNetInfo(ip=“173016001001” mask=“255255248000” gateway=“173016001251” commukey="123456")**
Successful reply	Return(result="success")
Failed reply	Return(result="failed" reason="FAILED REASON")
Caution	Return value is “success”, represents success to set network; Return value is “failed”, represents fail to set network, and returns failed reason in return value.
	For C220, C230, E350 and E350A model, this function needs set hostip parameter.

Command	**GetMAC_SN()**
Successful reply	Return(result="success" mac="90-EA-BA-DC-E9-F0" sn="31233")
Failed reply	Return(result="failed" reason="FAILED REASON")

Command	**DetectDevice()**
Successful reply	Return(result="success")
Failed reply	Return(result="failed" reason="FAILED REASON")
Caution	Return value is “success”, represents the device exists; Return value is “failed”, represents device does not exist, and return failed reason in return value.

Command	**RestartDevice()**
Successful reply	Return(result="success")
Failed reply	Return(result="failed")

Command	**SetDayLight(support="1" start_time="5-10 0" end_time="11-10 0")**
Successful reply	Return(result="success")
Failed reply	Return(result="failed" reason="FAILED REASON")
Caution	Parameter support=“1” means Daylight Saving Time (DST) function is open, and support=“0” means DST function is closed. The format of “start_time” and “end_time” is “Month-Day hour”.
	Besides this type of DST format, F810 support another DST format: ***SetDayLight(status="enable" start_time="5-10 0" end_time="11-10 0" format="month_date”)*** Parameter “status” is to indicate status of DST, “enable” is turn on the DST, “disable” is to turn off DST. Parameter “format” is to choose different DST formats, if the value is “month_date”, which means it uses “Month-Date hour” format; if the value is “month_week”, which means it uses “Month-Week-Day hour” format. For example, “Month-Week-Day hour” format represents “March-2nd-Friday 12”. Month: 1-12; Week:1-4 or L (Last week); Day: 1-Monday; 2- Tuesday; 3- Wednesday; 4- Thursday; 5- Friday; 6-Saturday; 7-Sunday Hour: 0-23

Command	**GetDayLight()**
Successful reply	Return(result="success" start_time=“5-10 0” end_time=“11-10 0”)
Failed reply	Return(result="failed" reason="FAILED REASON")
Caution	Return(result="failed" reason="FAILED REASON").

	For F810 model, the return value of this function also contains “status” and “format” parameters. Parameter “status” is to indicate status of DST, “enable” is turn on the DST, “disable” is to turn off DST. Parameter “format” is to choose different DST formats, if the value is “month_date”, which means it uses “Month-Date hour” format; if the value is “month_week”, which means it uses “Month-Week-Day hour” format. The result value shown as below: ***Return(result="success" status="enable" start_time="5-10 0" end_time="11-10 0" format="month_date")*** For example, “Month-Week-Day hour” format represents “March-2nd-Friday 12”. Month: 1-12; Week:1-4 or L (Last week); Day: 1-Monday; 2- Tuesday; 3- Wednesday; 4- Thursday; 5- Friday; 6-Saturday; 7-Sunday Hour: 0-23

Command	**GetWorkCode()**
Successful reply	Return(result="success" id=“1” name=“manager”)
Failed reply	Return(result="failed" reason="FAILED REASON")
Caution	This function has two formats: ● GetWorkCode(), returns all work codes and the corresponding contents of these work codes, when the command runs successfully. ● GetWorkCode(id=“1”), returns specific work code and its corresponding content, when the command runs successfully.
	For F810 model, the return value of this function contains parameter “status”. If status is “enable”, it means “Work Code” function turns on; if

	status is “enable”, it means “Work Code” function turns off. When this function runs successfully, but status is “disable”, then return value will be: Return(result="failed" reason="WorkCode Close"); If the device does not support this function, then "FAILED REASON" will be "unknown command".

Command	**SetWorkCode(1=“manager” 2=“employee”)**
Successful reply	Return(result="success")
Failed reply	Return(result="failed" reason="FAILED REASON")
Caution	Assume workcode=1 and its corresponding content is “manager”; workcode=2, its corresponding content is “employee”. User can define work codes and contents according the needs. If return value is “success”, then succeed to set work code; if return value is “failed”, then fail to set work code, and the failed reason will be return; If the device does not support this function, then "FAILED REASON" will be "unknown command".
	For F810 model, it needs to set parameter “status” in this function.

Command	**ClearWorkCode()**
Successful reply	Return(result="success")
Failed reply	Return(result="failed" reason="FAILED REASON")

Command	**DeleteWorkCode(id="1")**
Successful reply	Return(result="success")。
Failed reply	Return(result="failed" reason="unknown id")

Caution	Valid ID range of work code is 1-9999. If return “success”, then it means the specified work code has been deleted; if return “failed”, then it means the specified work word does not exist.

Command	**GetWorkStatus()**
Successful reply	Return(result="success" id=“1” name=“on duty”)
Failed reply	Return(result="failed" reason="FAILED REASON")
Caution	This command has two formats: ● GetWorkStatus(), returns all work status and the corresponding contents of these work status, when the command runs successfully. ● GetWorkStatus(id=“1”), returns specific work status and its corresponding content, when the command runs successfully.
	For F810 model, the return value of this function contains parameters “status”, “autostatus”, “ondutytime” and “offdutytime”. If parameter “workstatus” is “enable”, it means work status function turns on; if “workstatus” is “disable”, it means work status function turns off. If parameter “autostatus” is “enable”, it means status auto switch function turns on; If “autostatus” is “disable”, it means status auto switch function turns off. Parameter “ondutytime” represents on duty time; parameter “offdutytime” represents off duty time. If the function runs successfully, but “autostatus” is “disable”, then return value will be: Return(result="failed" reason="AutoStatus Close"); If the device does not support this function, then "FAILED REASON" will be "unknown command”.

Command	**SetWorkStatus(3="on weekend duty" 4="off weekend duty")**
Successful reply	Return(result="success")
Failed reply	Return(result="failed" reason="FAILED REASON")

Caution	Assume id="3", its corresponding content is "on weekend duty"; id="4", its corresponding content is "off weekend duty". User can define work status and contents according the needs. ID is 1 and 2 are two system default status, which the corresponding contents are "on duty" and "off duty". The default status cannot be modified or deleted.
	If return "success", it means succeed to set work status; if return "failed", then return failed reason. If the device does not support this function, then "FAILED REASON" will be "unknown command".
	For F810 model, this function needs to set parameters "workstatus", "autostatus", "ondutytime" and "offdutytime". The format: ***SetWorkStatus(workstatus="enable" autostatus="enable" ondutytime="09:00" offdutytime="18:00" 3="on weekend duty" 4="off weekend duty")***

Command	**ClearWorkStatus()**
Successful reply	Return(result="success")
Failed reply	Return(result="failed" reason="FAILED REASON")

Command	**DeleteWorkStatus(id="3")**
Successful reply	Return(result="success")。
Failed reply	Return(result="failed" reason="unknown id")
Caution	Range of ID for work status is 1-255. ID 1 and ID 2 is used by system, they can not be deleted. If return "success", it means succeed to delete work status by the specified ID; if return "failed", it means work status of the specified ID

	does not exist.

Command	**GetManagerID()**
Successful reply	Return(result="success" total="100" id="11" id="109" ...)。
Failed reply	Return(result="failed")。

Command	**GetManager(id="120")**

Successful reply	Return(result="success"
	id="120"
	name="Bob"
	card_num="0Xffffffff"
	pass_word="123456"
	face_data="ADFASERQERERTYSDFGHSDFGADSF..."
	......
	face_data="HASRTTYHRTAEFASDFQEQAFf...")
	There are total 18 sets of BASE64 encoding face templates data (face_data);

Failed reply	Return(result="failed" reason="FAILED REASON")。
Caution	Can only get one ID each time. For C220, C230, F810 model, the return value of this function also contains "enter_type"; For E350,E350A model, the result value also contains "enter_type, manager_type".

Command	**SetManager(id="1009"** **name="Bob"** **valid=""** **card_num="0Xffffffff"** **pass_word="123456"** **face_data="ADFASERQERERTYSDFGHSDFGADSF..."** **......** **face_data="HASRTTYHRTAEFASDFQEQAFf...")**

	There are total 18 sets of BASE64 encoding face templates data (face_data);
Successful reply	Return(result="success")。
Failed reply	Return(result="failed" reason="FAILED REASON")。
Caution	For C220, C230, E350 and E350A model, this function needs to set enter_type parameter.

Command	**DeleteManager(id="100")**
Successful reply	Return(result="success")。
Failed reply	Return(result="failed" reason="unknown id")
Caution	If return “success”, it means succeed to delete the specified administrator; if return “failed”, it means the specified administrator does not exist.

Command	**GetAttendanceInterval()**
Successful reply	Return(result="success" interval="2")。
Failed reply	Return(result="failed")。
Caution	Parameter “interval” means recording interval for each attendance record. Default value is 1 minutes, the range is 0 – 255 mins

Command	**SetAttendanceInterval (interval="2")**
Successful reply	Return(result="success")
Failed reply	Return(result="failed" reason="FAILED REASON")
Caution	Default value is 1 minutes, the range is 0 – 255 mins

Command	**OpenDoor()**
Successful reply	Return(result="success")
Failed reply	Return(result="failed")

Command	**ConfirmPassword()**

Successful reply	Return(result="success")
Failed reply	Return(result="failed" reason="FAILED REASON")
Caution	FAILED REASON will indicate the reason.

Command	**SetRelayTime(relay_time="20")**
Successful reply	Return(result="success")。
Failed reply	Return(result="failed" reason="FAILED REASON")。
Caution	To set relay time. Its unit is second. The range is 1-99s If return “success”, it means succeed to relay time; if return “failed”, then return failed reason. If the device does not support this function, then "FAILED REASON" will be "unknown command".

Command	**GetRemoveAlarm()**
Successful reply	Return(result="success" status="enable")
Failed reply	Return(result="failed" reason="FAILED REASON")。
Caution	If parameter “status” is “enable”, it means removal alarm function turns on; if “status” is “disable”, it means removal alarm function turns off.

Command	**SetRemoveAlarm(status="enable")**
Successful reply	Return(result="success")
Failed reply	Return(result="failed" reason="FAILED REASON")
Caution	If return “success”, it means succeed to removal alarm function; if return “failed”, then return failed reason. If the device does not support this function, then "FAILED REASON" will be "unknown command".

Command	**GetFRAlarm()**
Successful reply	Return(result="success" status="enable" fr_times="3" fr_period="10")
Failed reply	Return(result="failed" reason="FAILED REASON")
Caution	If parameter “status” is “enable”, it means recognition rejection alarm function turns on; if “status” is “disable”, it means recognition rejection alarm function turns off. Parameter fr_times means rejection recognition times, the range is 1-9 times; Parameter fr_period means alarm time for rejection recognition, the range is 1-99 seconds.

Command	**SetFRAlarm(status="enable" fr_times="3" fr_period="10"))**
Successful reply	Return(result="success")
Failed reply	Return(result="failed" reason="FAILED REASON")
Caution	If return “success”, it means succeed to recognition rejection alarm function; if return “failed”, then return failed reason. If the device does not support this function, then "FAILED REASON" will be "unknown command".

Command	**GetMagneAlarm()**
Successful reply	Return(result="success" status="enable" magne_delay="30" magne _period="20")
Failed reply	Return(result="failed" reason="FAILED REASON")。
Caution	If parameter “status” is “enable”, it means door sensor alarm function turns on; if “status” is “disable”, it means door sensor alarm function turns off. Parameter “magne_delay” means alarm delay time for door sensor alarm, the range is 1-99 seconds; Parameter “magne_period” means alarm period for door sensor alarm, the range is 1-99 seconds.

Command	**SetMagneAlarm(status="enable" "magne_delay="30"**

	magne _period="20")
Successful reply	Return(result="success")
Failed reply	Return(result="failed" reason="FAILED REASON")
Caution	If return “success”, it means succeed to set alarm of door sensor; if return “failed”, then return failed reason. If the device does not support this function, then "FAILED REASON" will be "unknown command".

Command	**GetDateTime()**
Successful reply	Return(result="success" date="2000-08-31" time="23:59:59" style="1")
Failed reply	Return(result="failed" reason="FAILED REASON")。

Command	**SetDateTime(date="2000-08-31" time="23:59:59")**
Successful reply	Return(result="success")
Failed reply	Return(result="failed" reason="FAILED REASON")
Caution	If return “success”, it means succeed to set date and time; if return “failed”, then return failed reason. If the device does not support this function, then "FAILED REASON" will be "unknown command".

Command	**GetVolume()**
Successful reply	Return(result="success" volume="1")
Failed reply	Return(result="failed" reason="FAILED REASON")。

Command	**SetVolume(volume="1")**
Successful reply	Return(result="success")
Failed reply	Return(result="failed" reason="FAILED REASON")
Caution	If return “success”, it means succeed to set volume status; if return

	“failed”, then return failed reason. If the device does not support this function, then "FAILED REASON" will be "unknown command".

Command	**GetWiegandOut()**

Successful reply	If user selects Wiegand format is 26-bit with site code or 34-bit with site code, the return result is:
	Return(result="success" pattern="w26" pulse_width="100" interval="1600" content="card" site_code="002")
	
	If user selects customized Wiegand format, then return result will also contain other parameters, the return result is:
	Return(result="success" pattern="w26" pulse_width="100" interval="1600" content="card" oem_bit="32" oem_value="4294927695" area_bit="32" area_value="294927695" card_bit="32" even_start="1" even_end="9" odd_start="16" odd_end="56")

Failed reply	Return(result="failed" reason="FAILED REASON")。

Command	**SetWiegandOut(pattern="w26" pulse_width="100" interval="1600" content="card")**
Successful reply	Return(result="success")
Failed reply	Return(result="failed" reason="FAILED REASON")
Caution	If selects Wiegand format is 26-bit with site code or 34-bit with site code, user needs to set “site_code” parameter, the function format is: ***SetWiegandOut(pattern="w26" pulse_width="100" interval="1600" content="card" site_code="002")*** If selects customized Wiegand format, then user needs to set other

	parameters, the function format is: ***SetWiegandOut(pattern="customize" pulse_width="100" interval="1600" content="card" oem_bit="32" oem_value="4294927695" area_bit="32" area_value="294927695" card_bit="32" even_start="1" even_end="9" odd_start="16" odd_end="56")*** If return “success”, it means succeed to set Wiegand output function; if return “failed”, then return failed reason. If the device does not support this function, then "FAILED REASON" will be "unknown command".

Command	**GetBell()**
Successful reply	Return(result="success" bell_number="1" bell_sound="1" alarm_time="09:00" bell_times= "3" status="enable" bell_number="2" bell_sound="5" alarm_time="12:00" bell_times= "3" status="enable")
Failed reply	Return(result="failed" reason="FAILED REASON")。
Caution	If parameter “status” is “enable”, it means bell function turns on; If “status” is “disable”, it means bell function turns off;

Command	**SetBell(bell_number="2" bell_sound="5" alarm_time="12:00" bell_times= "3" status="enable")**
Successful reply	Return(result="success")
Failed reply	Return(result="failed" reason="FAILED REASON")
Caution	If return “success”, it means succeed to set bell; if return “failed”, then return failed reason. If the device does not support this function, then "FAILED REASON" will be "unknown command".

Command	**GetOnOffSchedule()**
Successful reply	Return(result="success" turn_on_status="enable" turn_on_time="07:00"

	turn_off_status="enable" turn_off_time="23:00")
Failed reply	Return(result="failed" reason="FAILED REASON")。
Caution	If parameter “turn_on_status” is “enable”, it means power on timer function turns on; If “status” is “disable”, it means power on timer function turns off; If parameter “turn_off_status” is “enable”, it means power off timer function turns on; If “status” is “disable”, it means power off timer function turns off;

Command	**SetOnOffSchedule(turn_on_status="enable" turn_on_time="07:00" turn_off_status="enable" turn_off_time="23:00")**
Successful reply	Return(result="success")
Failed reply	Return(result="failed" reason="FAILED REASON")
Caution	If return “success”, it means succeed to set timer of power on/off; if return “failed”, then return failed reason. If the device does not support this function, then "FAILED REASON" will be "unknown command".

Command	**GetClientStatus()**
Successful reply	Return(result="success" status="enable" hostip="172016002201")
Failed reply	Return(result="failed" reason="FAILED REASON")。
Caution	If parameter “status” is “enable”, it means device client side function turns on; If “status” is “disable”, it means device client side function turns off;

Command	**SetClientStatus(status="enable" hostip="172016002201")**
Successful reply	Return(result="success")

Failed reply	Return(result="failed" reason="FAILED REASON")
Caution	If return “success”, it means succeed to set client side status; if return “failed”, it means, fail to set client side status. If the device does not support this function, then "FAILED REASON" will be "unknown command".

Command	SetAutoResetTime**(date="2000-08-31" time="23:59:59")**
Successful Reply	Return(result="success")
Failed Reply	Return(result="failed" reason="FAILED REASON")
Caution	If it returns “success”, Then it means succeed to set the autoreset time; If it returns “failed”, it means fail, and return with fail reason。If the device could not support the command, "FAILED REASON":"unknown command"

5.4 Image Management Command

Command	**GetPictureName(time="2009-11-1 0:0:0" type="face\photo\card ")**
Successful reply	Return(result=“success” name=“080816/OK/110530.JPG” name=“080816/OK/113554.JPG” ...) Return(result=“success” name=“080816/SORRY/090530.JPG” name=“080816/OK/133554.JPG” ...) Return(result=“success” name=“080816/CARD/140530.JPG” name=“080816/CARD/173554.JPG” ...

	)
Failed reply	Return(result="failed")
Caution	1. According to parameter “time” and “type”, it will generate 2 sub-directories: ● “time” is for the first sub-directory. Uses “YYMMDD” format as name, such as “100115”. ● “type” is for the second sub-directory. Definition: face – OK photo – SORRY card – CARD 2. The function can receive security images within a specified day, if search by day.

Command	**GetPicture(name="/090729/OK/093617.JPG")**
Successful reply	Return(result="success" photo="SDFQWERASDFAESRASDF...") (base-64 encoded)
Failed reply	Return(result="failed")
Caution	The value of parameter "name" is a unique mark which combines date, type and photo name.

Thanks for your interest and cooperation with Hanwang Technology Co., Ltd. Please contact below information if any question with this SDK.

Address：Hanvon Tower, Building No.5,
Zhongguancun Software Park,
Haidian District, Beijing,
P.R.China 100193

Phone：+86 (0)10 82786699 ext. 8204
Email：FaceIDSdk@hanwang.com.cn

Appendix - Device Function List

	FK605	F710	FA007	F810
GetEmployeeID	■	■	■	■
GetEmployee	■	■	■	■
SetEmployee	■	■	■	■
DeleteEmployee	■	■	■	■
DeleteAllEmployee	■	■	■	■
SetNameTable	■	■	■	■
AddNameTable	■	■	■	■
GetRecord	■	■	■	■
DeleteAllRecord	■	■	■	■
InitDevice	■	■	■	■
InitDeviceAdmin	■	■	■	■
GetDeviceInfo	■	■	■	■
SetDeviceInfo	■	■	■	■
GetPictureName	■	■	■	■
GetPicture	■	■	■	■
SetMAC_SN	■	■	■	■
OpenDoor		■	■	■
SetWorkCode	■	■	■	■
SetWorkStatus	■	■	■	■
GetWorkCode	■	■	■	■
GetWorkStatus	■	■	■	■
ClearWorkStatus	■	■	■	■
ClearWorkCode	■	■	■	■
SetDayLight	■	■	■	■
GetDayLight	■	■	■	■

	FK605	**F710**	**FA007**	**F810**
SetNetInfo	■	■	■	■
GetManagerID				■
GetManager				■
SetManager				■
UpdateFirmware				■
SetAttendanceInterval	■	■	■	■
SetVolume				■
GetVolume				■
GetWiegandOut				■
SetWiegandOut				■
GetAttendanceInterval				■
GetRemoveAlarm				■
SetRemoveAlarm				■
GetFRAlarm				■
SetFRAlarm				■
GetMagneAlarm				■
SetMagneAlarm				■
GetDateTime				■
SetDateTime				■
DetectEmployeeData				■
DeleteManager				■
GetNetInfo				■
DeleteWorkCode				■
DeleteWorkStatus				■
GetBell				■
SetBell				■
GetOnOffSchedule				■
SetOnOffSchedule				■

	FK605	**F710**	**FA007**	**F810**
SetRelayTime				■
GetClientStatus				■
SetClientStatus				■
GetMAC_SN				■
Server (Record Auto-Upload)				■