AiFace Device Web API Protocol Documentation (English)

Version: 1.0

Based on: webserver.cpp source code analysis

Protocol: HTTP + JSON

> ⚠️ **Protocol Role Note:** In the WebServer API, the **device acts as an HTTP server**, and browsers or other clients access the device as requesters. Therefore, this protocol is **restricted to Local Area Network (LAN) use only** — clients must be on the same network segment as the device to access it.

---

1. Overview

This document describes the HTTP API protocol of the embedded web server in AiFace facial recognition access control devices.

1.1 Server Architecture

Mode	Port	Description
HTTP	80 (default) or custom	Standard HTTP

- Maximum concurrent connections: **20**
- Idle connection timeout: **10 seconds**
- Timeout response: `HTTP 408 Timeout`
- Server busy response: `HTTP 503 Busy`

---

2. Connection & Authentication

2.1 Login Flow

Except for the `getlang` command, all `/api` commands require a `password` field in the request. The password is configured in the device's settings menu (mapped to `DI_WEBSERVER_PWD`).

Note: If no password has been set on the device (empty password), all password-required endpoints will return an error.

2.2 Request Endpoint

All JSON API requests are sent to:

```
POST http://<device_IP>:<port>/api
Content-Type: application/json
```

3. Common Request Format

```
{
  "cmd": "<command_name>",
  "password": "<device_password>",
  // ... command-specific parameters
}
```

Field	Type	Required	Description
`cmd`	string	Yes	Command name
`password`	string	Yes (except `getlang`)	Device management password

4. Common Response Format

```
{
  "ret": "<command_name>",
  "sn": "<device_serial_number>",
  "result": true/false,
  // ... command-specific response data
}
```

Field	Type	Description
`ret`	string	Echoes the request command name
`sn`	string	Device serial number
`result`	bool	Whether the operation succeeded
`reason`	number	Failure reason code (only on failure)
`msg`	string	Failure description text (only on failure)

5. Error Codes

reason	Description
1	User not found / Parameter format error
2	Password incorrect / Password is empty
3	Device busy (e.g., enrollment in progress)

reason	Description
5	No face detected in image
6	Multiple faces detected in image
7	Image resolution too large
8	Face area too large
9	Face area too small
10	Image quality too low
11	Face position too off-center
12	Duplicate face (already registered to another user)

6. HTTP Route Endpoints

Endpoint	Method	Description
`/api`	POST	Main JSON API entry point
`/api/firmware/upload`	POST	Chunked firmware upload
`/download/report`	GET	Download summary report (.xls)
`/download/rawrecord`	GET	Download raw attendance records (.xls)
`/photos/`	GET	Access user enrollment photos
`/logphotos/`	GET	Access attendance log snapshot photos
`/`	GET	Static file server (from `/usr/webserver` or `/data/res/webserver`)

7. JSON API Command Reference

7.1 Authentication & Device Info

7.1.1 `getlang` — Get Current Language

No password required

Request:

```
{ "cmd": "getlang" }
```

Response:

```
{
  "ret": "getlang",
  "sn": "SN123456",
  "result": true,
  "lang": "English"
}
```

Field	Type	Description
`lang`	string	Current language name

7.1.2 `getlangoption` — Get Available Language List

Request:

```
{ "cmd": "getlangoption", "password": "xxx" }
```

Response:

```
{
  "ret": "getlangoption",
  "sn": "SN123456",
  "result": true,
  "lang": "English",
  "langs": [
    { "lvalue": 0, "ltitle": "English" },
    { "lvalue": 1, "ltitle": "简体中文" }
  ]
}
```

Field	Type	Description
`lang`	string	Currently active language
`langs`	array	Available language list (only languages enabled in device settings)
`langs[].lvalue`	number	Language code
`langs[].ltitle`	string	Language display name

7.1.3 `login` — Login Verification

Request:

```
{
  "cmd": "login",
  "username": "admin",
  "password": "xxx"
}
```

Field	Type	Required	Description
`username`	string	Yes	Must be `"admin"`
`password`	string	Yes	Device password

Response (success):

```
{
  "ret": "login",
  "sn": "SN123456",
  "result": true
}
```

Response (failure):

```
{
  "ret": "login",
  "sn": "SN123456",
  "result": false,
  "reason": 1,
  "msg": "username error"
}
```

7.1.4 `reg` — Get Full Device Registration Info

Request:

```
{ "cmd": "reg", "password": "xxx" }
```

Response `devinfo` field breakdown:

Field	Type	Description
`modelname`	string	Device model name
`manufacturer`	string	Manufacturer
`fpalgo`	string	Fingerprint algorithm version (e.g., `"thbio3.0"`)
`firmware`	string	Firmware version
`time`	string	Current device time `YYYY-MM-DD HH:MM:SS`
`curip`	string	Current IP address
`mac`	string	MAC address
`timezone`	string	Timezone string
`bandwidth_up`	string	Upload bandwidth
`bandwidth_down`	string	Download bandwidth
`disksize`	string	Total disk capacity (MB)

Field	Type	Description
useddisk	string	Used disk space (MB)
deviceid	number	Device ID
usersize	number	Max user capacity
facesize	number	Max face enrollment capacity
avatarsize	number	Max avatar capacity
fpsize	number	Max fingerprint capacity
palmsize	number	Max palm print capacity
cardsize	number	Max card capacity
pwdsize	number	Max password capacity
logsize	number	Max log capacity
useduser	number	Current enrolled user count
usedface	number	Current enrolled face count
usedavatar	number	Current enrolled avatar count
usedfp	number	Current enrolled fingerprint count
usedpalm	number	Current enrolled palm print count
usedcard	number	Current enrolled card count
usedpwd	number	Current enrolled password count
usedlog	number	Total log count
usednewlog	number	New (unread) log count
usedrtlog	number	Real-time log count
netinuse	number	Active network: 0=none, 1=wired, 2=WiFi, 3=4G
usb4g	number	USB 4G dongle present: 0/1
battery	number	Battery level
cpu_temp	number	CPU temperature (°C)
intercom	number	Intercom supported: 0/1
elevator_control	number	Elevator control enabled: 0/1
floors	number	Total floor count
base_floor	number	Base floor number
have_g_floor	number	Has G (ground) floor: 0/1
acces_stimes	number	Access times
charid	number	Character ID
useosdp	number	OSDP protocol in use: 0/1

Field	Type	Description
`facetemplate`	number	Face template mode
`usequestion`	number	Questionnaire enabled: 0/1
`onlinedebug`	number	Online debug mode: 0/1
`dislanguage`	number	Disabled language bitmask
`ntp`	number	NTP enabled: 0/1
`station_mcc`	number	Station MCC
`station_mnc`	number	Station MNC
`station_lac`	number	Station LAC
`station_cid`	number	Station CID
`gprsip`	string	4G module IP
`gprsmac`	string	4G module MAC
`gprsversion`	string	4G module model
`gprsimei`	string	4G module IMEI

7.1.5 `getdevinfo` — Get Device Configuration Parameters

Request:

```
{ "cmd": "getdevinfo", "password": "xxx" }
```

Response: Returns all `DI_*` configuration key-value pairs.

Response Example:

```
{
  "ret": "getdevinfo",
  "sn": "SN123456",
  "result": true,
  "deviceid": 1,
  "language": 0,
  "managers": 1,
  "verifymode": 0,
  "volume": 80,
  "netipaddress": "192.168.1.100",
  "serverip": "192.168.1.200",
  "wlan_ssid": "MyWiFi",
  "door_opentime": 5,
  "live_detect": 1,
  "fcmatch_level": 60
}
```

Complete Configuration Parameter List:

General Settings

Field	Type	Description
`deviceid`	number	Device ID
`language`	number	Language
`managers`	number	Number of administrators
`verifymode`	number	Verification mode (see Appendix 10)
`volume`	number	Volume
`video_clarity`	number	Video clarity
`micvolume`	number	Microphone volume
`intercomvolume`	number	Intercom volume
`date_format`	number	Date format
`label_style`	number	Label style
`keyboard`	number	Keyboard settings
`usegps`	number	Use GPS
`time_format`	number	Time format
`use_dst`	number	Use daylight saving time
`dst_start`	number	DST start time
`dst_end`	number	DST end time
`show_avatar`	number	Show avatar
`show_battery`	number	Show battery
`use_qrcode`	number	Enable QR code
`screensavertime`	number	Screen saver timeout (seconds)
`showresulttime`	number	Verification result display time (seconds)
`sleeptime`	number	Sleep time (seconds)
`screensaver_wakeup`	number	Screen saver wake-up mode
`stranger_photo`	number	Stranger photo
`stranger_lock`	number	Stranger lock
`verifyfail_hint`	number	Verification failure hint
`userid_format`	number	User ID format
`rfid_free_reg`	number	RFID free registration
`rfid_auto_reg`	number	RFID auto registration
`shift_weekend_ot`	number	Shift weekend overtime
`rfid_module_type`	number	RFID module type
`filllight`	number	Fill light
`filllight_period`	number	Fill light period

Field	Type	Description
`accept_dose`	number	Accept dose
`face_double_check`	number	Face double check
`glog_warning`	number	Gate log warning threshold
`slog_warning`	number	System log warning threshold
`reverifytime`	number	Re-verification interval (seconds)
`use_logphoto`	number	Use log photo
`tts_voice`	number	TTS voice
`online_debug`	number	Online debug
`hide_ota`	number	Hide OTA upgrade
`multifaces`	number	Multi-face recognition
`disable_menu`	number	Disable menu
`server_keypad`	number	Server keypad
`keypad_pwd`	number	Keypad password
`reverify_nolock`	number	Re-verify without lock
`default_shift`	number	Default shift
`saturday_attend`	number	Saturday attendance
`sunday_attend`	number	Sunday attendance
`allow_latetime`	number	Allowed late time (minutes)
`allow_earlytime`	number	Allowed early leave time (minutes)
`auto_sign`	number	Auto sign-in
`shift_weekend`	number	Shift weekend
`excel_password`	string	Excel export password
`nosavelog`	number	Do not save logs
`hide_privacy`	number	Privacy protection
`reboottime1`	number	Scheduled reboot time 1
`reboottime2`	number	Scheduled reboot time 2
`reboottime3`	number	Scheduled reboot time 3
`reg_card_1`	number	Registration card 1
`reg_card_2`	number	Registration card 2
`reg_card_3`	number	Registration card 3
`selfchecklog`	number	Self-check log
`questionnaire`	number	Questionnaire
`questionnaire_card`	number	Questionnaire card

Field	Type	Description
`questionnaire_switch_card`	number	Questionnaire switch card
`event_schedule`	number	Event schedule 0
`event_schedule1`	number	Event schedule 1
`event_schedule2`	number	Event schedule 2
`event_schedule3`	number	Event schedule 3
`event_schedule4`	number	Event schedule 4
`event_schedule5`	number	Event schedule 5
`event_schedule6`	number	Event schedule 6
`event_schedule7`	number	Event schedule 7
`event_schedule8`	number	Event schedule 8
`bellcount`	number	Bell count
`belloutput`	number	Bell output
`ring_style`	number	Ring style

Biometric Settings

Field	Type	Description
`live_detect`	number	Liveness detection switch
`live_level`	number	Liveness detection level
`fcmatch_level`	number	Face matching threshold
`identify_distance`	number	Identification distance
`fpmatch_level`	number	Fingerprint matching threshold
`palm_vein_level`	number	Palm vein level
`palm_det_level`	number	Palm detection level
`fps_peruser`	number	Fingerprints per user
`disable_face`	number	Disable face
`enable_palm`	number	Enable palm

Serial & Network Settings

Field	Type	Description
`usedcom`	number	Serial port in use
`combaurate`	number	Serial baud rate
`netipaddress`	IP	Ethernet IP address
`netmask`	IP	Subnet mask
`netgateway`	IP	Gateway

Field	Type	Description
use_eth_dhcp	number	Use Ethernet DHCP
commpassword	number	Communication password
netportnum	number	Network port number
rs485_fun	number	RS485 function
use_wlan	number	Use WiFi
use_wlan_dhcp	number	Use WiFi DHCP
wlan_ip_address	IP	WiFi IP address
wlan_net_mask	IP	WiFi subnet mask
wlan_gateway	IP	WiFi gateway
wlan_ssid	string	WiFi SSID
wlan_pwd	string	WiFi password
gprs_apn	string	GPRS APN
gps_duration	number	GPS report interval
gprs_ipaddress	number	GPRS IP address
gprs_macaddress	number	GPRS MAC address

Server Settings

Field	Type	Description
serverip	IP	Server IP address
serverport	number	Server port
webserverport	number	Web server port
server_response_time	number	Server response timeout
user_add_enable	number	Allow adding users
use_bs	number	Use backend service
use_webserver	number	Use web server
bs_domain_name	string	Backend service domain name
use_domain_name	number	Use domain name
dns_server_ip	IP	DNS server IP
server_dns_name	string	Server domain name
server_verify	number	Server verification mode
server_verify_timeout	number	Server verification timeout
use_ntp	number	Use NTP
ntp_timezone	number	NTP timezone

Door Access Settings

Field	Type	Description
`door_opentime`	number	Door open time (seconds)
`door_sensor_type`	number	Door sensor type
`door_alarm_time`	number	Door alarm time
`door_firealarmin`	number	Fire alarm input
`door_interlock`	number	Door interlock
`door_mutiopen`	number	Multi-door open
`door_trycount`	number	Attempt count alarm
`access_times`	number	Access times
`door_antipass`	number	Anti-passback
`multiusers`	number	Multi-user verification
`door_password`	number	Door password
`timezone_punchs`	number	Timezone punch
`access_denied_nolog`	number	Access denied without log
`outof_notimzone_denied`	number	Deny out-of-timezone
`elevator_control`	number	Elevator control
`floor_number`	number	Floor number
`base_floor`	number	Base floor
`have_g_floor`	number	Has G floor
`elevator_keypad`	number	Elevator keypad
`maintenance_card`	number	Maintenance card

Wiegand Settings

Field	Type	Description
`wiegand_output_mode`	number	Wiegand output mode
`card_disp_format`	number	Card display format
`card_reversal`	number	Card reversal
`wg_reversal`	number	Wiegand reversal
`wiegand_output_bit`	number	Wiegand output bits
`wiegand_outside_input`	number	Wiegand outside input
`alarm_output`	number	Alarm output
`outwg_shot`	number	Wiegand output trigger
`wiegand_port`	number	Wiegand port

Card & Security Settings

Field	Type	Description
`bio_to_card`	number	Biometric to card number
`bio_verify_waiver`	number	Biometric verify waiver
`bio_deletion`	number	Biometric deletion
`bio_user_deletion`	number	User biometric deletion
`cpucard_key`	string	CPU card key
`qrcode_pwd`	string	QR code password
`webserver_pwd`	string	Web server password

Other Settings

Field	Type	Description
`threat`	number	Threat level
`hide_ip`	number	Hide IP display
`key_lock`	number	Key lock
`face_wakeup_seconds`	number	Face wake-up time (seconds)

> **Type Notes:** `number` = numeric, `IP` = IP address string (e.g. `"192.168.1.100"`), `string` = plain string.

7.1.6 `setdevinfo` — Set Device Configuration Parameters

Request:

```
{
  "cmd": "setdevinfo",
  "password": "xxx",
  "nowtime": 1700000000,
  "deviceid": 1,
  "volume": 80,
  "live_detect": 1,
  "fcmatch_level": 60
}
```

Field	Type	Required	Description
`nowtime`	number	No	Unix timestamp (seconds) for time sync
Other fields	-	No	Any `DI_*` configuration key-value pairs; only include items you want to modify

> **Type Rules:**
> - IP-type fields (e.g. `netipaddress`, `serverip`, etc.) accept IP address strings like `"192.168.1.100"`

- String-type fields (e.g. `wlan_ssid`, `excel_password`, `cpucard_key`, etc.) accept string values
- All other fields are numeric and accept numbers

7.1.7 `getdevcap` — Get Device Capacity Info

Request:

```
{ "cmd": "getdevcap", "password": "xxx" }
```

Response:

```
{
  "ret": "getdevcap",
  "sn": "SN123456",
  "result": true,
  "usersize": 10000,
  "facesize": 10000,
  "avatarsize": 10000,
  "fpsize": 30000,
  "palmsize": 30000,
  "cardsize": 10000,
  "pwdsize": 10000,
  "logsize": 100000,
  "useduser": 150,
  "usedface": 148,
  "usedavatar": 100,
  "usedfp": 10,
  "usedpalm": 0,
  "usedcard": 50,
  "usedpwd": 20,
  "usedlog": 5000,
  "usednewlog": 100,
  "usedrtlog": 5
}
```

Field	Type	Description
`usersize`	number	Max user capacity
`facesize`	number	Max face enrollment capacity
`avatarsize`	number	Max avatar capacity
`fpsize`	number	Max fingerprint capacity
`palmsize`	number	Max palm print capacity
`cardsize`	number	Max card capacity
`pwdsize`	number	Max password capacity
`logsize`	number	Max log capacity
`useduser`	number	Current enrolled user count
`usedface`	number	Current enrolled face count
`usedavatar`	number	Current enrolled avatar count

Field	Type	Description
`usedfp`	number	Current enrolled fingerprint count
`usedpalm`	number	Current enrolled palm print count
`usedcard`	number	Current enrolled card count
`usedpwd`	number	Current enrolled password count
`usedlog`	number	Total log count
`usednewlog`	number	New (unread) log count
`usedrtlog`	number	Real-time log count

7.2 User Management

7.2.1 `getuserlist` — Get User List (Paginated)

Request:

```
{
  "cmd": "getuserlist",
  "password": "xxx",
  "stn": 1
}
```

Field	Type	Description
`stn`	number	1 = start from beginning; 0 = continue to next page

Response:

```
{
  "ret": "getuserlist",
  "sn": "SN123456",
  "result": true,
  "count": 100,
  "to": 50,
  "record": [
    {
      "id": "1001",
      "name": "John Doe",
      "department": "Engineering",
      "shift": 0,
      "admin": 0,
      "fingerprint": 2,
      "palm": 0,
      "face": 1,
      "password": 0,
      "card": 12345678,
      "weekzone": 0,
      "group": 0,
      "access_times": 0,
      "verifymode": 0,
```

```
      "enable": 1,
      "birthday": "01-15",
      "starttime": "2024-01-01 00: 00",
      "endtime": "2025-12-31 23: 59",
      "photourl": "/photos/LF00001001.jpg"
    }
  ]
}
```

Field	Type	Description
`count`	number	Total user count
`to`	number	Number of records returned (0 if no more data)
`record[]`	array	Array of user objects (max 50 per page)
`record[].id`	string	User ID
`record[].name`	string	User name
`record[].department`	string	Department
`record[].shift`	number	Shift ID
`record[].admin`	number	Admin flag (0=user, 1=admin)
`record[].fingerprint`	number	Enrolled fingerprint count
`record[].palm`	number	Enrolled palm print count
`record[].face`	number	Enrolled face count
`record[].password`	number	Password set flag
`record[].card`	number	Card ID
`record[].weekzone`	number	Week zone ID
`record[].group`	number	Group ID
`record[].access_times`	number	Allowed access times
`record[].verifymode`	number	Verification mode (see Appendix 10)
`record[].enable`	number	Enable flag (1=enabled, 0=disabled)
`record[].birthday`	string	Birthday `"MM-DD"` (append `*` for lunar)
`record[].starttime`	string	Valid start time
`record[].endtime`	string	Valid end time
`record[].photourl`	string	Photo URL (if available)
`record[].userprofile`	string	User profile string (conditional)

> Returns up to 50 records per call. Call repeatedly with `stn=0` until `to >= count` or `to == 0`.

7.2.2 `getuserids` — Get All User IDs

Request:

```
{ "cmd": "getuserids", "password": "xxx" }
```

Response:

```
{
  "ret": "getuserids",
  "sn": "SN123456",
  "result": true,
  "count": 3,
  "record": ["1001", "1002", "1003"]
}
```

Field	Type	Description
`count`	number	Total user count
`record`	array	Array of user ID strings

7.2.3 `getunuserdid` — Get Next Available User ID

Request:

```
{ "cmd": "getunuserdid", "password": "xxx" }
```

Response:

```
{
  "ret": "getunuserdid",
  "sn": "SN123456",
  "result": true,
  "enrollid": "1004"
}
```

Field	Type	Description
`enrollid`	string	Next unused user ID

7.2.4 `checkuserid` — Check If User ID Exists

Request:

```
{
  "cmd": "checkuserid",
  "password": "xxx",
  "enrollid": "1001"
}
```

Field	Type	Description
`enrollid`	string/number	User ID to check

Response:

```
{
  "ret": "checkuserid",
  "sn": "SN123456",
  "result": true,
  "exists": true
}
```

Field	Type	Description
`exists`	bool	Whether the user ID exists

7.2.5 `getuserinfo` — Get Single User Details

Request:

```
{
  "cmd": "getuserinfo",
  "password": "xxx",
  "enrollid": "1001",
  "backupnum": 50
}
```

Field	Type	Description
`enrollid`	string/number	User ID or alias ID
`backupnum`	number	Optional — retrieve specific biometric data (see Appendix)

Response (with face photo, backupnum=50):

```
{
  "ret": "getuserinfo",
  "sn": "SN123456",
  "result": true,
  "enrollid": "1001",
  "name": "John Doe",
  "department": "Engineering",
  "admin": 0,
  "pwd": 0,
  "card": 12345678,
  "fpflag": 3,
  "fpcnt": 2,
  "palmflag": 1,
  "palmcnt": 1,
  "faceflag": 1,
  "birthday": "01-15",
  "starttime": "2024-01-01 00: 00: 00",
  "endtime": "2025-12-31 23: 59: 59",
  "photourl": "/photos/LF00001001.jpg",
  "backupnum": 50,
```

```
  "record": "<Base64-encoded face photo>"
}
```

Response (user not found):

```
{
  "ret": "getuserinfo",
  "sn": "SN123456",
  "result": false,
  "reason": 1,
  "msg": "can not find the user",
  "enrollid": "1001"
}
```

Field	Type	Description
`enrollid`	string	User ID
`name`	string	User name
`department`	string	Department
`admin`	number	Admin flag
`pwd`	number	Password flag
`card`	number	Card ID (conditional)
`fpflag`	number	Fingerprint flag bitmask (conditional)
`fpcnt`	number	Enrolled fingerprint count (conditional)
`palmflag`	number	Palm print flag (conditional)
`palmcnt`	number	Enrolled palm print count (conditional)
`faceflag`	number	Face flag (conditional)
`enable`	number	User enabled flag (conditional)
`birthday`	string	Birthday `"MM-DD"` (conditional)
`shiftid`	number	Shift ID (conditional)
`verifymode`	number	Verification mode (conditional; see Appendix 10)
`zoneid`	number	Time zone ID (conditional)
`groupid`	number	Group ID (conditional)
`access_times`	number	Allowed access times (conditional)
`postid`	number	Department/post ID (conditional)
`starttime`	string	Valid start time (conditional)
`endtime`	string	Valid end time (conditional)
`userprofile`	string	User profile string (conditional)
`backupnum`	number	Requested backupnum (conditional)
`record`	string	Base64 biometric data (conditional, varies by backupnum)

Field	Type	Description
`photourl`	string	Photo URL (conditional)

`record` format by `backupnum`:

backupnum	record format	Description
0–9	string	Fingerprint template — hex with zero run-length compression (`(N)` = N consecutive `0x00` bytes), 810 bytes after decoding. Omitted when the slot is empty.
10	number	User password
11	number	Card number
40 / 41	string	Palm vein template — Base64, 1548 bytes after decoding
50	string	Face JPEG photo — Base64
51	string	Face feature + key points — Base64, 1074 bytes after decoding (2-byte header + 1072-byte feature)

> An exported `record` can be pushed verbatim to another device with `setuserinfo` (using the same `backupnum`) to migrate biometric templates.

7.2.6 `setuserinfo` — Add or Update User Information

Request:

```
{
  "cmd": "setuserinfo",
  "password": "xxx",
  "enrollid": "1001",
  "name": "John Doe",
  "admin": 0,
  "pwd": 123456,
  "card": 12345678,
  "department": "Engineering",
  "birthday": "01-15",
  "starttime": "2024-01-01 00: 00: 00",
  "endtime": "2025-12-31 23: 59: 59",
  "verifymode": 0,
  "shiftid": 1,
  "zoneid": 1,
  "groupid": 0,
  "access_times": 0,
  "enable": 1,
  "face": "<Base64-encoded face photo>"
}
```

Field	Type	Description
`enrollid`	string/number	User ID (string or number)

Field	Type	Description
`name`	string	Full name
`admin`	number	Privilege: 0=regular user, 1=administrator
`pwd`	number	Password (numeric)
`card`	number/string	RFID card ID
`department`	string	Department name (auto-created if not found)
`postid`	number	Department/post ID
`birthday`	string	Birthday in `MM-DD` format (append `*` for lunar calendar)
`starttime`	string	Valid start datetime `YYYY-MM-DD HH:MM:SS`
`endtime`	string	Valid end datetime `YYYY-MM-DD HH:MM:SS`
`verifymode`	number	User verification mode (0 = any method; see Appendix 10)
`shiftid`	number	Shift ID
`zoneid`	number	Time zone ID
`groupid`	number	Group ID
`access_times`	number	Allowed pass count (0=unlimited, max 30000)
`enable`	number	User enabled: 0/1
`face`	string	Base64-encoded face image (JPEG, ≤512KB)
`backupnum`	number	Biometric slot index (see Appendix 9)
`record`	string	Biometric template data; encoding depends on `backupnum` (see below)
`userprofile`	string	User profile string

How each biometric type is pushed:

Biometric	`backupnum`	Data field	`record` encoding	Length after decoding
Face photo	omitted (or 50)	`face`	Base64 (JPEG)	≤ 512KB
Fingerprint	0–9 (finger index)	`record`	Hex + zero run-length compression (**not** Base64)	810 bytes
Palm vein / palm print	40 = left, 41 = right	`record`	Base64	1548 bytes
Face feature + key points	51	`record`	Base64	1074 bytes (2-byte header + 1072-byte feature)

⚠️ The fingerprint `record` is **not Base64** — it is a "hex + zero run-length" text string. Only palm vein and face feature use Base64. The decoded length must match the table exactly, otherwise the device replies `"xxx data error"`.

`record` encoding rules:

backupnum	Encoding rule
0–9 fingerprint	Each byte as 2 lowercase hex digits; runs of N `0x00` bytes are compressed to `(N)`. E.g. `00 00 00` → `(3)`, and bytes `0a 1f 00 00 2b` → `0a1f(2)2b`. Must decode to exactly 810 bytes.
40 / 41 palm vein	Standard Base64, must decode to exactly 1548 bytes (Base64 string is always 2064 characters).
51 face feature	Standard Base64, must decode to exactly 1074 bytes: 2-byte template header (`08 06` on regular platforms, `08 10` on BOARD_CORE_810) followed by the 1072-byte feature vector, which already contains 10 float face key points and the normalization parameters.
50 face photo	Returned as Base64 JPEG by `getuserinfo` only; push photos with the `face` field.

Recommended call order: create the user first with a `setuserinfo` call that carries no biometric data, then push each template by `backupnum`. Pushing `backupnum=51` for a non-existent user fails with `"face error"`.

Response (success):

```
{
  "ret": "setuserinfo",
  "sn": "SN123456",
  "result": true,
  "enrollid": "1001",
  "backupnum": 50
}
```

Response (failure):

```
{
  "ret": "setuserinfo",
  "sn": "SN123456",
  "result": false,
  "reason": 5,
  "msg": "no face",
  "enrollid": "1001"
}
```

Face enrollment failure reason codes:

reason	Description
1	Image file too large
5	No face detected
6	Multiple faces detected
7	Image dimensions too large
8	Face region too large
9	Face region too small
10	Image quality too low

reason	Description
11	Face position too off-center
12	Duplicate face (already registered to another user ID, e.g. `face is double,id=1002`)

Example 1 — push a face photo (`face` field, device extracts the feature on site):

```
{
  "cmd": "setuserinfo",
  "password": "xxx",
  "enrollid": "1001",
  "name": "John Doe",
  "face":
"/9j/4AAQSkZJRgABAQAAAQABAAD/2wBDAAgGBgcGBQgHBwcJCQgKDBQNDAsLDBkSEw8UHRofHh0aHBwgJC4nICIsIxwcKDcpLDAxNDQ0Hyc5PTgyPC
4zNDL..."
}
```

```
{
  "ret": "setuserinfo",
  "sn": "SN123456",
  "result": true,
  "enrollid": "1001"
}
```

Example 2 — push a fingerprint (slot 0, `backupnum` =0, `record` is the compressed hex string):

```
{
  "cmd": "setuserinfo",
  "password": "xxx",
  "enrollid": "1001",
  "backupnum": 0,
  "record": "0a1f2b(807)"
}
```

> The sample above is a complete, format-valid payload: 3 hex bytes followed by `(807)` (807 `0x00` bytes), which decodes to exactly 810 bytes. **Minimal testable payload:** `"record": "(810)"` , which decodes to 810 `0x00` bytes and passes format validation. To push a real template, export the same encoding from an enrolled device with `getuserinfo` (`backupnum` =0) or `bioread` .

Example 3 — push a palm vein template (left hand, `backupnum` =40; right hand is 41):

```
{
  "cmd": "setuserinfo",
  "password": "xxx",
  "enrollid": "1001",
  "backupnum": 40,
  "record": "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA..."
}
```

> `record` is standard Base64 and must decode to exactly 1548 bytes (the Base64 string is always 2064 characters). **Minimal testable payload:** Base64 of 1548 `0x00` bytes (2064 `A` characters).

Example 4 — push a face feature with key points (`backupnum` =51, `record` is Base64):

```
{
  "cmd": "setuserinfo",
  "password": "xxx",
  "enrollid": "1001",
  "backupnum": 51,
  "record": "CAYAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA..."
}
```

> `record` is standard Base64 and must decode to exactly 1074 bytes. **Minimal testable payload:** `08 06` + 1072 `0x00` bytes, i.e. a Base64 string starting with `CAYA` followed by 1428 `A` characters (1432 characters in total); on BOARD_CORE_810 use the `08 10` header instead (Base64 starts with `CBAA`). Export real features from an enrolled device with `getuserinfo` (`backupnum` =51) and feed them back to other devices unchanged.

Response (success):

```
{
  "ret": "setuserinfo",
  "sn": "SN123456",
  "result": true,
  "enrollid": "1001",
  "backupnum": 51
}
```

> The response echoes `backupnum` whenever it was present in the request.

Response (failures):

```
{
  "ret": "setuserinfo",
  "sn": "SN123456",
  "result": false,
  "reason": 12,
  "msg": "finger already used,id=1002",
  "finger": 0,
  "enrollid": "1001",
  "backupnum": 0
}
```

```
{
  "ret": "setuserinfo",
  "sn": "SN123456",
  "result": false,
  "reason": 1,
  "msg": "palm data error",
  "palm": 0,
  "enrollid": "1001",
  "backupnum": 40
}
```

reason	msg	Description
1	space overflow	Face or fingerprint database is full
1	base64 wrong	`face` field failed Base64 decoding
3	file size=0	Image byte count is 0

reason	msg	Description
1	record can not empty	`backupnum` 0–9 but `record` is missing
1	finger data error	Fingerprint string is malformed or does not decode to 810 bytes
1	palm data error	Palm vein Base64 is malformed or does not decode to 1548 bytes
1	palm save failed	Palm vein could not be written to the device
1	face error	Face feature could not be written (user missing or feature DB full)
12	finger already used,id=xxx	Fingerprint duplicated with another user (response carries `finger`)
12	palm already used,id=xxx	Palm vein duplicated with another user (response carries `palm`)
12	face is double,id=xxx	Face duplicated with another user
12	card already used,id=xxx	Card number already owned by another user (response carries `card`)
5/6/7/8/9/10/11	see table above	Face photo extraction failed on device

> The `reason` =12 duplicate checks (face / fingerprint / palm vein / card) are controlled by the device duplicate-check switch; when it is off, no check is made and the data overwrites the existing entry.

7.2.7 `deleteuser` — Delete User or Specific Biometric Data

Request:

```
{
  "cmd": "deleteuser",
  "password": "xxx",
  "enrollid": "1001",
  "backupnum": 12
}
```

Field	Type	Description
`enrollid`	string/number	User ID or alias ID
`backupnum`	number	Action type (see table below)

backupnum	Action
12 or 13	Delete entire user
0-9	Delete fingerprint at specified index
10	Clear password
11	Clear card
40/41	Delete left/right palm print

backupnum	Action
50 or 51	Delete face

Response:

```
{
  "ret": "deleteuser",
  "sn": "SN123456",
  "result": true,
  "backupnum": 12,
  "enrollid": "1001"
}
```

Field	Type	Description
`backupnum`	number	Echoed backupnum value
`enrollid`	string	User ID

7.2.8 `deleteuserface` — Delete User Face

Request:

```
{
  "cmd": "deleteuserface",
  "password": "xxx",
  "enrollid": "1001"
}
```

Field	Type	Description
`enrollid`	string/number	User ID or alias ID

Response:

```
{
  "ret": "deleteuserface",
  "sn": "SN123456",
  "result": true,
  "enrollid": "1001"
}
```

7.2.9 `deleteuserfp` — Delete All User Fingerprints

Request:

```
{
  "cmd": "deleteuserfp",
  "password": "xxx",
  "enrollid": "1001"
}
```

Field	Type	Description
`enrollid`	string/number	User ID or alias ID

Response:

```
{
  "ret": "deleteuserfp",
  "sn": "SN123456",
  "result": true,
  "enrollid": "1001"
}
```

7.2.10 `deleteuserpalm` — Delete All User Palm Prints

Request:

```
{
  "cmd": "deleteuserpalm",
  "password": "xxx",
  "enrollid": "1001"
}
```

Field	Type	Description
`enrollid`	string/number	User ID or alias ID

Response:

```
{
  "ret": "deleteuserpalm",
  "sn": "SN123456",
  "result": true,
  "enrollid": "1001"
}
```

7.2.11 `deleteusers` — Batch Delete Users

Request:

```
{
  "cmd": "deleteusers",
  "password": "xxx",
  "list": ["1001", "1002", "1003"]
}
```

Field	Type	Description
`list`	array	Array of user IDs (string or number) to delete

Response:

```
{
  "ret": "deleteusers",
  "sn": "SN123456",
  "result": true
}
```

7.2.12 `cleanuser` — Delete All Users

Request:

```
{ "cmd": "cleanuser", "password": "xxx" }
```

> ⚠️ **DESTRUCTIVE**: Removes all enrolled users from the device.

Response:

```
{
  "ret": "cleanuser",
  "sn": "SN123456",
  "result": true
}
```

7.2.13 `enableuser` — Enable or Disable a User

Request:

```
{
  "cmd": "enableuser",
  "password": "xxx",
  "enrollid": "1001",
  "enflag": 1
}
```

Field	Type	Description
`enrollid`	string/number	User ID or alias ID
`enflag`	number	1=enable, 0=disable (default: 1)

Response:

```
{
  "ret": "enableuser",
  "sn": "SN123456",
  "result": true,
  "enrollid": "1001"
}
```

7.3 Biometric Enrollment (Device-Side)

7.3.1 `adduser` — Trigger On-Device Biometric Enrollment

Triggers the device's local screen to perform fingerprint/palm/face enrollment.

Request:

```
{
  "cmd": "adduser",
  "password": "xxx",
  "enrollid": "1001",
  "name": "John Doe",
  "admin": 0,
  "backupnum": 50,
  "flag": 0
}
```

Field	Type	Description
`cancel`	bool	Set to `true` to cancel current enrollment
`enrollid`	string/number	User ID
`backupnum`	number	Enrollment type (see Appendix)
`admin`	number	Whether to enroll as administrator
`flag`	number	Flag bits
`name`	string	User name

Response (success):

```
{
  "ret": "adduser",
  "sn": "SN123456",
  "result": true,
  "backupnum": 50,
  "enrollid": "1001"
}
```

Response (cancel):

```
{
  "ret": "adduser",
  "result": true,
  "cancel": true
}
```

Response (failure):

```
{
  "ret": "adduser",
  "sn": "SN123456",
  "result": false,
  "reason": 3,
  "msg": "device busy",
```

```
  "backupnum": 50,
  "enrollid": "1001"
}
```

Failure reason codes:

reason	Description
1	Empty user ID
2	Invalid backupnum
3	Device busy (enrollment already in progress)

7.3.2 `checkregstatus` — Query Enrollment Status

Request:

```
{ "cmd": "checkregstatus", "password": "xxx" }
```

Response:

```
{
  "ret": "checkregstatus",
  "sn": "SN123456",
  "result": true,
  "status": 1,
  "msg": "success",
  "image": "<Base64-encoded preview image>"
}
```

Field	Type	Description
`status`	number	Enrollment status code (see table below)
`msg`	string	One-time status message (cleared after read)
`image`	string	Base64 preview image (conditional)

status	Description
-1	Idle / cancelled
0	Enrollment in progress
1	Enrollment succeeded
other	Enrollment failed

7.4 Log Management

7.4.1 `getlog` — Get Historical Logs (Paginated)

Request:

```
{
  "cmd": "getlog",
  "password": "xxx",
  "index": 0,
  "enrollid": "1001",
  "from": "2024-06-01",
  "to": "2024-12-31"
}
```

Field	Type	Description
`index`	number	Start position (0 = begin new query)
`enrollid`	string/number	Optional — filter by specific user (only when index=0)
`from`	string	Optional — start date `YYYY-MM-DD` format
`to`	string	Optional — end date `YYYY-MM-DD` format

Response (up to 50 records per page):

```
{
  "ret": "getlog",
  "sn": "SN123456",
  "result": true,
  "count": 500,
  "from": 0,
  "to": 50,
  "record": [
    {
      "enrollid": "1001",
      "name": "John Doe",
      "time": "2024-06-01 09: 00: 00",
      "mode": 1,
      "inout": 1,
      "event": 0,
      "note": "remark",
      "photourl": "/logphotos/logimage123.jpg"
    }
  ]
}
```

Field	Type	Description
`count`	number	Total matching log count
`from`	number	Start index of returned records
`to`	number	End index of returned records
`record[]`	array	Log record array (max 50 per page)
`record[].enrollid`	string	User ID
`record[].name`	string	User name
`record[].time`	string	Timestamp

Field	Type	Description
`record[].mode`	number	Verification method (1=face, 2=fingerprint, 3=card, 4=password, etc.)
`record[].inout`	number	Direction: 1=entry, 0=exit
`record[].event`	number	Event type code
`record[].note`	string/object	Remark or additional note (conditional)
`record[].photourl`	string	Snapshot photo URL (conditional)

> Call repeatedly with incrementing `index` until `to >= count`.

7.4.2 `getrtlog` — Get Real-Time Logs (Consumed on Read)

Request:

```
{ "cmd": "getrtlog", "password": "xxx", "index": 0 }
```

Field	Type	Description
`index`	number	Start position (0 = begin new query)

Response:

```
{
  "ret": "getrtlog",
  "sn": "SN123456",
  "result": true,
  "count": 25,
  "from": 0,
  "to": 20,
  "record": [
    {
      "enrollid": "1001",
      "name": "John Doe",
      "time": "2024-06-01 09: 00: 00",
      "mode": 1,
      "inout": 1,
      "event": 0,
      "note": "remark"
    }
  ]
}
```

Field	Type	Description
`count`	number	Total real-time log count in queue
`from`	number	Start index
`to`	number	End index
`record[]`	array	Log record array (max 20 per page), same fields as `getlog`

> Real-time logs are **removed from the queue once read**. Up to 20 records per page. Call repeatedly with incrementing `index` until `to >= count`.

7.4.3 `setlogs` — Mark Logs as Synced

Request:

```
{
  "cmd": "setlogs",
  "password": "xxx",
  "records": [
    { "id": "1001", "event": 0, "mode": 1, "time": "2024-06-01 09: 00: 00" }
  ]
}
```

Field	Type	Description
`records`	array	Array of log records to mark as synced
`records[].id`	string/number	User ID
`records[].event`	number	Event type
`records[].mode`	number	Verification mode
`records[].time`	string	Log timestamp

Response:

```
{
  "ret": "setlogs",
  "sn": "SN123456",
  "result": true
}
```

7.4.4 `cleanlog` — Clear All Logs

Request:

```
{ "cmd": "cleanlog", "password": "xxx" }
```

> ⚠️ **DESTRUCTIVE**: Removes all historical attendance logs.

Response:

```
{
  "ret": "cleanlog",
  "sn": "SN123456",
  "result": true
}
```

7.4.5 `cleanlogphoto` — Clear All Log Photos

Request:

```
{ "cmd": "cleanlogphoto", "password": "xxx" }
```

Response:

```
{
  "ret": "cleanlogphoto",
  "sn": "SN123456",
  "result": true
}
```

7.5 System Management

7.5.1 `gettime` — Get Device Time

Request:

```
{ "cmd": "gettime", "password": "xxx" }
```

Response:

```
{
  "ret": "gettime",
  "sn": "SN123456",
  "time": "2024-06-01 10: 00: 00"
}
```

Field	Type	Description
`time`	string	Current device time

7.5.2 `settime` — Set Device Time

Method 1 (string format):

```
{
  "cmd": "settime",
  "password": "xxx",
  "cloudtime": "2024-06-01 10: 00: 00"
}
```

Method 2 (Unix timestamp in seconds):

```
{
  "cmd": "settime",
  "password": "xxx",
  "nowtime": 1717200000
}
```

Field	Type	Description
`cloudtime`	string	Time string in `YYYY-MM-DD HH:MM:SS` format (use this or `nowtime`)
`nowtime`	number	Unix timestamp in seconds (use this or `cloudtime`)

Response (success):

```
{
  "ret": "settime",
  "sn": "SN123456",
  "result": true
}
```

Response (failure):

```
{
  "ret": "settime",
  "sn": "SN123456",
  "result": false,
  "reason": 1
}
```

> Validates year >= 2021, month 1-12, day, hour, minute, second ranges. Returns reason 1 on failure.

7.5.3 `reboot` — Reboot Device

Request:

```
{ "cmd": "reboot", "password": "xxx" }
```

> ⚠️ **No response** — the device reboots immediately. The HTTP connection will be dropped.

7.5.4 `disabledevice` — Disable Device (Pause Recognition)

Request:

```
{ "cmd": "disabledevice", "password": "xxx" }
```

Response:

```
{
  "ret": "disabledevice",
  "sn": "SN123456",
  "result": true
}
```

> The response is sent first, then the device disables face/fingerprint/palm/ID card recognition.

7.5.5 `enabledevice` — Enable Device (Resume Recognition)

Request:

```
{ "cmd": "enabledevice", "password": "xxx" }
```

Response:

```
{
  "ret": "enabledevice",
  "sn": "SN123456",
  "result": true
}
```

7.5.6 `initsys` — Full System Initialization (Clear All Data)

> ⚠️ **DESTRUCTIVE**: Clears all users, logs, and settings (language setting is preserved). Also removes bell schedules, post data, attendance settings, and lock data. Backs up WiFi configuration.

Request:

```
{ "cmd": "initsys", "password": "xxx" }
```

Response:

```
{
  "ret": "initsys",
  "sn": "SN123456",
  "result": true
}
```

7.5.7 `initmenu` — Reset Menu Settings (Preserve User Data)

Request:

```
{ "cmd": "initmenu", "password": "xxx" }
```

Response:

```
{
  "ret": "initmenu",
  "sn": "SN123456",
  "result": true
}
```

> Same as `initsys` but does **NOT** clear user database or log database.

7.5.8 `cleandatebase` — Clear Log Database

Request:

```
{ "cmd": "cleandatebase", "password": "xxx" }
```

> ⚠️ **DESTRUCTIVE**: Clears entire database and sets a reboot flag. Device will reboot after this operation.

Response:

```
{
  "ret": "cleandatebase",
  "sn": "SN123456",
  "result": true
}
```

7.5.9 `cleanadmin` — Remove All Administrator Privileges

Request:

```
{ "cmd": "cleanadmin", "password": "xxx" }
```

Response:

```
{
  "ret": "cleanadmin",
  "sn": "SN123456",
  "result": true
}
```

7.5.10 `cleaninactiveuser` — Remove Users Without Biometric Data

Request:

```
{ "cmd": "cleaninactiveuser", "password": "xxx" }
```

Response:

```
{
  "ret": "cleaninactiveuser",
  "sn": "SN123456",
  "result": true
}
```

7.5.11 `setscreensaver` — Set Screensaver Parameters

Request:

```
{
  "cmd": "setscreensaver",
  "password": "xxx",
  "filename": "logo.jpg",
  "photo": "<Base64-encoded image>",
  "duration": 60,
  "remove": false,
  "removeall": false
}
```

Field	Type	Description
`filename`	string	Screensaver image filename
`photo`	string	Base64-encoded image content
`duration`	number	Display duration (seconds)
`remove`	bool	Remove specific screensaver image
`removeall`	bool	Remove all screensaver images

Response:

```
{
  "ret": "setscreensaver",
  "sn": "SN123456",
  "result": true
}
```

7.5.12 `upgrade` — Trigger OTA Firmware Upgrade via JSON API

Request:

```
{
  "cmd": "upgrade",
  "password": "xxx",
  "filename": "ai230_product.zip",
  "md5": "d41d8cd98f00b204e9800998ecf8427e",
  "allcount": 100,
  "index": 0,
  "count": 50,
```

```
  "record": "<Base64-encoded firmware chunk>"
}
```

Field	Type	Description
`filename`	string	Firmware filename (must match device model, `.zip` or `.pkg`)
`md5`	string	Full file MD5 checksum (verified on last chunk)
`allcount`	number	Total number of chunks
`index`	number	Current chunk index (0-based)
`count`	number	Number of chunks in this request
`record`	string	Base64-encoded firmware data chunk

Response (success):

```
{
  "ret": "upgrade",
  "sn": "SN123456",
  "result": true
}
```

Response (failure):

```
{
  "ret": "upgrade",
  "sn": "SN123456",
  "result": false,
  "reason": 1,
  "msg": "file mismatch or insufficient space"
}
```

Failure reason codes:

reason	Description
1	File mismatch or insufficient disk space
2	MD5 checksum verification failed
3	File move error
4	Wrong file extension (must be .zip or .pkg)

7.5.13 `setvoice` — Configure Voice Prompts

Request:

```
{
  "cmd": "setvoice",
  "password": "xxx",
  "index": 0,
  "voice": "<Base64-encoded WAV audio>",
```

```
  "play": true
}
```

Field	Type	Description
`index`	number	Voice prompt index
`voice`	string	Base64-encoded WAV audio data (max 512KB)
`play`	bool	Play the voice prompt immediately after setting

Response:

```
{
  "ret": "setvoice",
  "sn": "SN123456",
  "result": true
}
```

7.5.14 `keypad` — Activate/Deactivate Keypad Mode

Request:

```
{
  "cmd": "keypad",
  "password": "xxx",
  "password": true,
  "keep": false,
  "timeout": 30,
  "on": true
}
```

Field	Type	Description
`password`	bool	Enable keypad password input
`keep`	bool	Keep keypad active after use
`timeout`	number	Keypad timeout (seconds)
`on`	bool	Turn keypad on/off

Response:

```
{
  "ret": "keypad",
  "sn": "SN123456",
  "result": true
}
```

7.6 Door Access Control

7.6.1 `opendoor` — Remote Door Open

Request:

```
{
  "cmd": "opendoor",
  "password": "xxx",
  "enrollid": "1001",
  "floor": 3,
  "msg": "Welcome"
}
```

Field	Type	Description
`enrollid`	string/number	Optional — associated user ID (for Wiegand output)
`floor`	number	Optional — elevator control floor number
`msg`	string	Optional — message to display on screen (for 3 seconds)

Response:

```
{
  "ret": "opendoor",
  "sn": "SN123456",
  "result": true
}
```

7.6.2 `lockctrl` — Door Lock State Control

Request:

```
{
  "cmd": "lockctrl",
  "password": "xxx",
  "fuc": 1
}
```

Field	Type	Description
`fuc`	number	Lock control function code

fuc	Description
1	Normally Open (NO)
2	Normally Closed (NC)
0	Restore normal operation

Response:

```
{
  "ret": "lockctrl",
```

```
  "sn": "SN123456",
  "result": true
}
```

7.6.3 `getdoorstatus` — Get Door Status

Request:

```
{ "cmd": "getdoorstatus", "password": "xxx" }
```

Response:

```
{
  "ret": "getdoorstatus",
  "sn": "SN123456",
  "result": true,
  "doorstatus": 1
}
```

Field	Type	Description
`doorstatus`	number	Current door status code

7.6.4 `setdevlock` — Configure Door Access Parameters

Request (all fields optional):

```
{
  "cmd": "setdevlock",
  "password": "xxx",
  "opendelay": 5,
  "doorsensor": 1,
  "alarmdelay": 30,
  "InputAlarm": 0,
  "antpass": 0,
  "interlock": 0,
  "mutiopen": 0,
  "tryalarm": 5,
  "access_times": 0,
  "punchtimes": 0,
  "tamper": 0,
  "wgformat": 26,
  "wgoutput": 0,
  "cardformat": 0,
  "dayzone": [],
  "weekzone": [],
  "lockgroup": [],
  "nopentime": [],
  "visitortime": []
}
```

Field	Type	Description
`opendelay`	number	Door open delay (seconds)

Field	Type	Description
`doorsensor`	number	Door sensor type
`alarmdelay`	number	Alarm delay (seconds)
`InputAlarm`	number	Input alarm setting
`antpass`	number	Anti-passback mode
`interlock`	number	Interlock function
`mutiopen`	number	Multi-user door open
`tryalarm`	number	Failed verification alarm threshold
`access_times`	number	Access times limit
`punchtimes`	number	Punch times limit
`tamper`	number	Tamper alarm setting
`wgformat`	number	Wiegand format: 0=26bit, 1=34bit, 2=42bit, 3=50bit, 4=58bit, 5=66bit, 6=37bit
`wgoutput`	number	Wiegand output mode
`cardformat`	number	Card format
`dayzone`	array	Day time zone schedule array
`weekzone`	array	Week time zone schedule array
`lockgroup`	array	Lock group array
`nopentime`	array	No-open time array
`visitortime`	array	Visitor time array

Response:

```
{
  "ret": "setdevlock",
  "sn": "SN123456",
  "result": true
}
```

7.6.5 `getdevlock` — Get Door Access Parameters

Request:

```
{ "cmd": "getdevlock", "password": "xxx" }
```

Response:

```
{
  "ret": "getdevlock",
  "sn": "SN123456",
  "result": true,
  "opendelay": 5,
```

```
  "doorsensor": 1,
  "alarmdelay": 30,
  "InputAlarm": 0,
  "antpass": 0,
  "interlock": 0,
  "mutiopen": 0,
  "tryalarm": 5,
  "access_times": 0,
  "tamper": 0,
  "wgformat": 26,
  "wgoutput": 0,
  "cardformat": 0,
  "dayzone": [],
  "weekzone": [],
  "lockgroup": [],
  "nopentime": [],
  "visitortime": []
}
```

Field	Type	Description
`opendelay`	number	Door open delay (seconds)
`doorsensor`	number	Door sensor type
`alarmdelay`	number	Alarm delay (seconds)
`InputAlarm`	number	Input alarm setting
`antpass`	number	Anti-passback mode
`interlock`	number	Interlock function
`mutiopen`	number	Multi-user door open
`tryalarm`	number	Failed verification alarm threshold
`access_times`	number	Access times limit
`tamper`	number	Tamper alarm setting
`wgformat`	number	Wiegand format: 0=26bit, 1=34bit, 2=42bit, 3=50bit, 4=58bit, 5=66bit, 6=37bit
`wgoutput`	number	Wiegand output mode
`cardformat`	number	Card format
`dayzone`	array	Day time zone schedule
`weekzone`	array	Week time zone schedule
`lockgroup`	array	Lock group settings
`nopentime`	array	No-open time schedule
`visitortime`	array	Visitor time schedule

7.6.6 `getuserlock` — Get User-Level Access Settings

Request:

```
{
  "cmd": "getuserlock",
  "password": "xxx",
  "enrollid": "1001"
}
```

Field	Type	Description
`enrollid`	string/number	User ID or alias ID

Response (found):

```
{
  "ret": "getuserlock",
  "sn": "SN123456",
  "result": true,
  "enrollid": "1001",
  "birthday": "01-15",
  "verifymode": 0,
  "enable": 1,
  "weekzone": 0,
  "group": 0,
  "access_times": 0,
  "starttime": "2024-01-01 00: 00: 00",
  "endtime": "2025-12-31 23: 59: 59"
}
```

Response (not found):

```
{
  "ret": "getuserlock",
  "sn": "SN123456",
  "result": false,
  "reason": 1,
  "msg": "can not find the user",
  "enrollid": "1001"
}
```

Field	Type	Description
`enrollid`	string	User ID
`birthday`	string	Birthday `"MM-DD"` format
`verifymode`	number	Verification mode (see Appendix 10)
`enable`	number	Enable flag (1=enabled, 0=disabled)
`weekzone`	number	Week zone ID
`group`	number	Group ID
`access_times`	number	Allowed access times
`starttime`	string	Valid start time
`endtime`	string	Valid end time

7.6.7 `setuserlock` — Set User-Level Access Parameters (Batch)

Request:

```
{
  "cmd": "setuserlock",
  "password": "xxx",
  "count": 2,
  "record": [
    {
      "enrollid": "1001",
      "birthday": "01-15",
      "verifymode": 0,
      "weekzone": 0,
      "group": 0,
      "access_times": 0,
      "starttime": "2024-01-01 00: 00: 00",
      "endtime": "2025-12-31 23: 59: 59"
    },
    {
      "enrollid": "1002",
      "birthday": "06-20",
      "verifymode": 1,
      "weekzone": 1,
      "group": 0,
      "access_times": 5,
      "starttime": "2024-01-01 00: 00: 00",
      "endtime": "2025-12-31 23: 59: 59"
    }
  ]
}
```

Field	Type	Description
`count`	number	Number of records in the array
`record`	array	Array of user access settings
`record[].enrollid`	string/number	User ID
`record[].birthday`	string	Birthday `"MM-DD"` format
`record[].verifymode`	number	Verification mode (see Appendix 10)
`record[].weekzone`	number	Week zone ID
`record[].group`	number	Group ID
`record[].access_times`	number	Allowed access times
`record[].starttime`	string	Valid start time
`record[].endtime`	string	Valid end time

Response (success):

```
{
  "ret": "setuserlock",
  "sn": "SN123456",
  "result": true
}
```

Response (failure):

```
{
  "ret": "setuserlock",
  "sn": "SN123456",
  "result": false,
  "reason": 1,
  "msg": "setuserlock fail"
}
```

7.6.8 `deleteuserlock` — Delete User Access Settings

Request:

```
{
  "cmd": "deleteuserlock",
  "password": "xxx",
  "enrollid": "1001"
}
```

Field	Type	Description
`enrollid`	string/number	User ID or alias ID

Response:

```
{
  "ret": "deleteuserlock",
  "sn": "SN123456",
  "result": true,
  "enrollid": "1001"
}
```

7.6.9 `cleanuserlock` — Clear All User Access Settings

Request:

```
{ "cmd": "cleanuserlock", "password": "xxx" }
```

Response:

```
{
  "ret": "cleanuserlock",
  "sn": "SN123456",
  "result": true
}
```

7.6.10 `openallfloors` — Open All Floors

Request:

```
{ "cmd": "openallfloors", "password": "xxx" }
```

Response:

```
{
  "ret": "openallfloors",
  "sn": "SN123456",
  "result": true
}
```

7.6.11 `closeallfloors` — Close All Floors

Request:

```
{ "cmd": "closeallfloors", "password": "xxx" }
```

Response:

```
{
  "ret": "closeallfloors",
  "sn": "SN123456",
  "result": true
}
```

7.6.12 `openselectfloors` — Open Selected Floors

Request:

```
{
  "cmd": "openselectfloors",
  "password": "xxx",
  "profile": "1,2,5"
}
```

Field	Type	Description
`profile`	string	Comma-separated floor numbers to open (optional)

Response:

```
{
  "ret": "openselectfloors",
  "sn": "SN123456",
  "result": true
}
```

7.6.13 `closeselectfloors` — Close Selected Floors

Request:

```
{
  "cmd": "closeselectfloors",
  "password": "xxx",
  "profile": "3,4"
}
```

Field	Type	Description
`profile`	string	Comma-separated floor numbers to close (optional)

Response:

```
{
  "ret": "closeselectfloors",
  "sn": "SN123456",
  "result": true
}
```

7.7 Shifts & Attendance

7.7.1 `getshift` — Get Shift Configuration

Request:

```
{ "cmd": "getshift", "password": "xxx" }
```

Response:

```
{
  "ret": "getshift",
  "sn": "SN123456",
  "result": true,
  "shifts": [
    {
      "name": "Shift 1",
      "cutoftime": 10,
      "sections": [
        { "start": "09: 00", "end": "12: 00", "type": 0 },
        { "start": "13: 00", "end": "18: 00", "type": 0 },
        { "start": "", "end": "", "type": 0 }
      ]
    }
  ]
}
```

Field	Type	Description
`shifts`	array	Array of up to 8 shift definitions
`shifts[].name`	string	Shift name
`shifts[].cutoftime`	number	Cutoff time (minutes) for late arrival
`shifts[].sections`	array	Array of 3 time sections per shift
`shifts[].sections[].start`	string	Section start time `"HH:MM"`
`shifts[].sections[].end`	string	Section end time `"HH:MM"`
`shifts[].sections[].type`	number	Section type

7.7.2 `setshift` — Set Shift Configuration

Request:

```
{
  "cmd": "setshift",
  "password": "xxx",
  "shifts": [
    {
      "name": "Shift 1",
      "cutoftime": 10,
      "sections": [
        { "start": "09: 00", "end": "12: 00", "type": 0 },
        { "start": "13: 00", "end": "18: 00", "type": 0 },
        { "start": "", "end": "", "type": 0 }
      ]
    }
  ]
}
```

Field	Type	Description
`shifts`	array	Array of up to 8 shift definitions
`shifts[].name`	string	Shift name
`shifts[].cutoftime`	number	Cutoff time (minutes)
`shifts[].sections`	array	Array of 3 time sections per shift
`shifts[].sections[].start`	string	Section start time `"HH:MM"`
`shifts[].sections[].end`	string	Section end time `"HH:MM"`
`shifts[].sections[].type`	number	Section type

Response:

```
{
  "ret": "setshift",
  "sn": "SN123456",
  "result": true
}
```

7.7.3 `getbelltime` — Get Bell Schedule

Request:

```
{ "cmd": "getbelltime", "password": "xxx" }
```

Response:

```
{
  "ret": "getbelltime",
  "sn": "SN123456",
  "result": true,
  "belltimes": [
    { "time": "09: 00", "flag": 1 },
    { "time": "12: 00", "flag": 0 },
    { "time": "13: 00", "flag": 1 },
    { "time": "18: 00", "flag": 0 }
  ]
}
```

Field	Type	Description
`belltimes`	array	Bell schedule array
`belltimes[].time`	string	Bell time `"HH:MM"`
`belltimes[].flag`	number	Bell type flag

7.7.4 `setbelltime` — Set Bell Schedule

Request:

```
{
  "cmd": "setbelltime",
  "password": "xxx",
  "belltimes": [
    { "time": "09: 00", "flag": 1 },
    { "time": "12: 00", "flag": 0 },
    { "time": "13: 00", "flag": 1 },
    { "time": "18: 00", "flag": 0 }
  ]
}
```

Field	Type	Description
`belltimes`	array	Bell schedule array
`belltimes[].time`	string	Bell time `"HH:MM"`
`belltimes[].flag`	number	Bell type flag

Response:

```
{
  "ret": "setbelltime",
```

```
  "sn": "SN123456",
  "result": true
}
```

7.7.5 `getholiday` — Get Holiday Settings

Request:

```
{ "cmd": "getholiday", "password": "xxx" }
```

Response:

```
{
  "ret": "getholiday",
  "sn": "SN123456",
  "result": true,
  "holidays": [
    {
      "name": "New Year",
      "startday": "01-01",
      "endday": "01-03",
      "shift": 0,
      "dayzone": 0
    }
  ]
}
```

Field	Type	Description
`holidays`	array	Array of up to 30 holiday definitions
`holidays[].name`	string	Holiday name
`holidays[].startday`	string	Start date `"MM-DD"`
`holidays[].endday`	string	End date `"MM-DD"`
`holidays[].shift`	number	Shift ID to apply
`holidays[].dayzone`	number	Day zone ID

7.7.6 `setholiday` — Set Holiday Settings

Request:

```
{
  "cmd": "setholiday",
  "password": "xxx",
  "holidays": [
    {
      "name": "New Year",
      "startday": "01-01",
      "endday": "01-03",
      "shift": 0,
      "dayzone": 0
```

```
    }
  ]
}
```

Field	Type	Description
`holidays`	array	Array of up to 30 holiday definitions
`holidays[].name`	string	Holiday name
`holidays[].shift`	number	Shift ID to apply
`holidays[].dayzone`	number	Day zone ID
`holidays[].startday`	string	Start date `"MM-DD"`
`holidays[].endday`	string	End date `"MM-DD"`

Response:

```
{
  "ret": "setholiday",
  "sn": "SN123456",
  "result": true
}
```

7.8 Reports

7.8.1 `getreport` — Get Attendance Report Data

Request:

```
{
  "cmd": "getreport",
  "password": "xxx",
  "enrollid": "1001",
  "from": "2024-01-01",
  "to": "2024-12-31"
}
```

Field	Type	Description
`enrollid`	string/number	Optional — filter by specific user
`from`	string	Start date `"YYYY-MM-DD"`
`to`	string	End date `"YYYY-MM-DD"`

Response (success):

```
{
  "ret": "getreport",
  "sn": "SN123456",
  "result": true
}
```

Response (no logs found):

```
{
  "ret": "getreport",
  "sn": "SN123456",
  "result": false,
  "reason": 1,
  "msg": "can not find logs"
}
```

> Response contains dynamic report data fields depending on the report type and date range.

7.8.2 `generatereport` — Generate Report Files

Request:

```
{
  "cmd": "generatereport",
  "password": "xxx",
  "enrollid": "1001",
  "from": "2024-01-01",
  "to": "2024-12-31"
}
```

Field	Type	Description
`enrollid`	string/number	Optional — filter by specific user
`from`	string	Start date `"YYYY-MM-DD"`
`to`	string	End date `"YYYY-MM-DD"`

Response (success):

```
{
  "ret": "generatereport",
  "sn": "SN123456",
  "result": true,
  "msg": "Report generation started."
}
```

Response (busy):

```
{
  "ret": "generatereport",
  "sn": "SN123456",
  "result": false,
  "reason": 3,
  "msg": "A report is already being generated. Please wait."
}
```

> After generation completes, files can be downloaded via `/download/report` and `/download/rawrecord`.

7.8.3 `checkreportprogress` — Check Report Generation Progress

Request:

```
{ "cmd": "checkreportprogress", "password": "xxx" }
```

Response (in progress):

```
{
  "ret": "checkreportprogress",
  "sn": "SN123456",
  "result": true,
  "progress": 75,
  "status": "generating",
  "finished": false
}
```

Response (completed):

```
{
  "ret": "checkreportprogress",
  "sn": "SN123456",
  "result": true,
  "progress": 100,
  "status": "completed",
  "finished": true,
  "file_exists": true
}
```

Field	Type	Description
`progress`	number	Generation progress percentage (0-100)
`status`	string	Current status description
`finished`	bool	Whether generation is complete
`file_exists`	bool	Whether report file is available for download (only when finished)

7.9 Questionnaire & Notices

7.9.1 `setquestionnaire` — Configure Questionnaire

Request:

```
{
  "cmd": "setquestionnaire",
  "password": "xxx",
  "title": "Health Declaration",
  "voice": "Please complete health declaration",
  "errmsg": "Declaration failed",
  "radio": true,
  "optionflag": 3,
```

```
  "items": ["Question 1", "Question 2"],
  "schedules": ["08: 00-09: 00", "12: 00-13: 00"],
  "usequestion": true,
  "useschedule": false,
  "card": 0,
  "switch_card": 0,
  "enrollid": "1001",
  "open": true
}
```

Field	Type	Description
`title`	string	Questionnaire title
`voice`	string	Voice prompt text
`errmsg`	string	Error message text
`radio`	bool	Use radio button style (single select)
`optionflag`	number	Option flag
`items`	array	Array of up to 16 questionnaire items (strings)
`schedules`	array	Array of up to 8 schedule strings
`usequestion`	bool	Enable questionnaire feature
`useschedule`	bool	Enable schedule-based questionnaire
`card`	number/string	Associated card ID
`switch_card`	number/string	Switch card ID
`enrollid`	string/number	Associated user ID
`open`	bool	Open questionnaire for responses

Response:

```
{
  "ret": "setquestionnaire",
  "sn": "SN123456",
  "result": true
}
```

7.9.2 `getquestionnaire` — Get Questionnaire Configuration

Request:

```
{ "cmd": "getquestionnaire", "password": "xxx" }
```

Response (configured):

```
{
  "ret": "getquestionnaire",
  "sn": "SN123456",
  "result": true,
  "title": "Health Declaration",
```

```
  "voice": "Please complete health declaration",
  "errmsg": "Declaration failed",
  "radio": true,
  "optionflag": 3,
  "usequestion": true,
  "useschedule": false,
  "card": 0,
  "switch_card": 0,
  "items": ["Question 1", "Question 2"],
  "schedules": ["08: 00-09: 00", "12: 00-13: 00"]
}
```

Response (not configured):

```
{
  "ret": "getquestionnaire",
  "sn": "SN123456",
  "result": false
}
```

Field	Type	Description
`title`	string	Questionnaire title
`voice`	string	Voice prompt text
`errmsg`	string	Error message text
`radio`	bool	Radio button mode
`optionflag`	number	Option flag
`usequestion`	bool	Questionnaire enabled
`useschedule`	bool	Schedule mode enabled
`card`	number	Associated card ID
`switch_card`	number	Switch card ID
`items`	array	Questionnaire items
`schedules`	array	Schedule strings

7.9.3 `cleanquestionnaire` — Clear Questionnaire Settings

Request:

```
{ "cmd": "cleanquestionnaire", "password": "xxx" }
```

Response:

```
{
  "ret": "cleanquestionnaire",
  "sn": "SN123456",
  "result": true
}
```

7.9.4 `setcompanyname` — Set Company Name

Request:

```
{
  "cmd": "setcompanyname",
  "password": "xxx",
  "name": "ACME Corporation"
}
```

Field	Type	Description
`name`	string	Company name (max 21 characters)

Response:

```
{
  "ret": "setcompanyname",
  "sn": "SN123456",
  "result": true
}
```

7.9.5 `getcompanyname` — Get Company Name

Request:

```
{ "cmd": "getcompanyname", "password": "xxx" }
```

Response:

```
{
  "ret": "getcompanyname",
  "sn": "SN123456",
  "result": true,
  "name": "ACME Corporation"
}
```

Field	Type	Description
`name`	string	Current company name

7.10 Video Intercom (WebRTC)

7.10.1 `enablewebrtc` — Initiate Video Call

Request:

```
{
  "cmd": "enablewebrtc",
  "password": "xxx",
  "intercom": true,
  "monitor": true,
  "inputmode": 0,
  "waitseconds": 30,
  "stunserver": "stun:stun.l.google.com: 19302",
  "turnserver": "",
  "turnusername": "",
  "turnpassword": ""
}
```

Field	Type	Description
`intercom`	bool	Enable intercom mode
`monitor`	bool	Enable monitor mode
`inputmode`	number	Input mode: 0=direct call (no input dialog), 1=enroll/ID number, 2=phone number, 3=room number, 4=visual selection menu
`audioonly`	number / bool	Audio-only intercom: `1` /true = audio only, `0` /false = audio + video (also controls video on/off)
`waitseconds`	number	Wait timeout (seconds)
`stunserver`	string	STUN server URL
`turnserver`	string	TURN server URL
`turnusername`	string	TURN server username
`turnpassword`	string	TURN server password

Response (success):

```
{
  "ret": "enablewebrtc",
  "sn": "SN123456",
  "result": true,
  "lanscape": 0
}
```

Response (hardware not supported):

```
{
  "ret": "enablewebrtc",
  "sn": "SN123456",
  "result": false,
  "reason": 1,
  "msg": "this device have not this function"
}
```

Field	Type	Description
`lanscape`	number	Screen orientation (landscape flag)

7.10.2 `getwebrtc` — Get WebRTC Signaling Status

Request:

```
{ "cmd": "getwebrtc", "password": "xxx" }
```

Response (success):

```
{
  "ret": "getwebrtc",
  "sn": "SN123456",
  "result": true,
  "intercom": true,
  "monitor": false,
  "inputmode": 0,
  "waitseconds": 30,
  "stunserver": "stun:stun.l.google.com: 19302",
  "turnserver": "",
  "turnusername": "",
  "turnpassword": ""
}
```

Response (hardware not supported):

```
{
  "ret": "getwebrtc",
  "sn": "SN123456",
  "result": false,
  "reason": 1,
  "msg": "this device have not this function"
}
```

Field	Type	Description
`intercom`	bool	Intercom enabled
`monitor`	bool	Monitor enabled
`inputmode`	number	Audio input mode
`waitseconds`	number	Wait timeout (seconds)
`stunserver`	string	STUN server URL
`turnserver`	string	TURN server URL
`turnusername`	string	TURN server username
`turnpassword`	string	TURN server password

7.10.3 `checklive` — Check Live Stream Status

Request:

```
{ "cmd": "checklive", "password": "xxx" }
```

Response (ICE candidate available):

```
{
  "sn": "SN123456",
  "cmd": "icecandidate",
  "candidate": "<ICE candidate string>"
}
```

Response (call cancelled on device):

```
{
  "sn": "SN123456",
  "cmd": "talkcancel"
}
```

Response (no activity):

```
(null — no response)
```

> This command has non-standard response format. Returns ICE candidate data when available, cancellation notice if call was ended on the device side, or no response at all when idle.

7.10.4 `answer` — SDP Answer

Request:

```
{
  "cmd": "answer",
  "password": "xxx",
  "sdp": "<SDP string>"
}
```

Field	Type	Description
`sdp`	string	SDP answer string

Response (success):

```
{
  "ret": "answer",
  "sn": "SN123456",
  "result": true
}
```

Response (failure):

```
{
  "ret": "answer",
  "sn": "SN123456",
  "result": false,
  "reason": 1,
  "msg": "device error"
}
```

Failure reason codes:

reason	Description
1	Device error
2	SDP parameter missing

7.10.5 `icecandidate` — ICE Candidate Exchange

Request:

```
{
  "cmd": "icecandidate",
  "password": "xxx",
  "candidate": "<ICE candidate string>"
}
```

Field	Type	Description
`candidate`	string	ICE candidate string

Response:

```
{
  "ret": "icecandidate",
  "sn": "SN123456",
  "result": true
}
```

7.10.6 `callcancel` — Cancel Call

Request:

```
{ "cmd": "callcancel", "password": "xxx" }
```

Response:

```
{
  "ret": "callcancel",
  "sn": "SN123456",
  "result": true
}
```

7.10.7 `monitorcall` — Monitor Mode Call

Request:

```
{ "cmd": "monitorcall", "password": "xxx" }
```

Response (offer SDP):

```
{
  "cmd": "offer",
  "sn": "SN123456",
  "type": "offer",
  "sdp": "<SDP offer string>"
}
```

Response (failure):

```
{
  "ret": "monitorcall",
  "sn": "SN123456",
  "result": false,
  "reason": 1,
  "msg": "device busy"
}
```

Failure reason codes:

reason	Description
1	Device busy (another call in progress)
2	Offer generation failed

7.11 Biometric Verification

7.11.1 `verify` — Remote Biometric Verification

Performs remote biometric verification on the device using provided biometric data.

Request:

```
{
  "cmd": "verify",
  "password": "xxx",
  "enrollid": "1001",
  "card": "12345678",
  "pwd": 123456,
  "name": "John Doe",
  "face": "<Base64-encoded face JPEG>",
  "finger": "<Hex fingerprint template>",
  "finger2": "<Hex second fingerprint template>",
  "profile": "default"
}
```

Field	Type	Description
`enrollid`	string/number	Optional — expected user ID for matching
`card`	string/number	Optional — card number for verification
`pwd`	number	Optional — numeric password for verification

Field	Type	Description
`name`	string	Optional — user name
`face`	string	Optional — Base64-encoded face image (JPEG)
`finger`	string	Optional — Hex-encoded fingerprint template
`finger2`	string	Optional — Hex-encoded second fingerprint template
`profile`	string	Optional — verification profile string

Response (success):

```
{
  "ret": "verify",
  "sn": "SN123456",
  "result": true,
  "enrollid": "1001"
}
```

Response (failure):

```
{
  "ret": "verify",
  "sn": "SN123456",
  "result": false,
  "reason": 1,
  "msg": "verification failed",
  "enrollid": "1001"
}
```

Failure reason codes:

reason	Description
1	User not found
2	Password incorrect
3	Card not found
4	Fingerprint mismatch
5	No face detected
6	Multiple faces detected
7	Image too large
8	Face too large
9	Face too small
10	Image quality too low
11	Face position off-center
12	Face not matched

7.11.2 `biomatch` — Biometric Matching (OSDP Test)

Request:

```
{ "cmd": "biomatch", "password": "xxx" }
```

> Requires `config_OSDP_TEST` configuration.

Response:

```
{
  "ret": "biomatch",
  "sn": "SN123456",
  "result": true
}
```

7.11.3 `bioread` — Read Raw Biometric Data (OSDP Test)

Request:

```
{ "cmd": "bioread", "password": "xxx" }
```

> Requires `config_OSDP_TEST` configuration.

Response:

```
{
  "ret": "bioread",
  "sn": "SN123456",
  "result": true
}
```

7.12 File Operations

7.12.1 `getdir` — List Directory Contents

Request:

```
{
  "cmd": "getdir",
  "password": "xxx",
  "path": "/data"
}
```

Field	Type	Description
`path`	string	Directory path to list

Response (success):

```
{
  "ret": "getdir",
  "sn": "SN123456",
  "result": true,
  "files": ["config.txt", "photo.jpg", "logs/"]
}
```

Response (failure):

```
{
  "ret": "getdir",
  "sn": "SN123456",
  "result": false
}
```

Field	Type	Description
`files`	array	Array of file/directory name strings

7.12.2 `getfile` — Get File Content (Base64 encoded)

Request:

```
{
  "cmd": "getfile",
  "password": "xxx",
  "path": "/data/config.txt"
}
```

Field	Type	Description
`path`	string	File path to read

Response (success):

```
{
  "ret": "getfile",
  "sn": "SN123456",
  "result": true,
  "base64": "<Base64-encoded file content>"
}
```

Response (failure):

```
{
  "ret": "getfile",
  "sn": "SN123456",
  "result": false
}
```

Field	Type	Description
`base64`	string	Base64-encoded file content

7.12.3 `writefile` — Write File

Request:

```
{
  "cmd": "writefile",
  "password": "xxx",
  "path": "/data/notice.txt",
  "base64": "<Base64-encoded content>"
}
```

Field	Type	Description
`path`	string	File path to write
`base64`	string	Base64-encoded file content

Response (success):

```
{
  "ret": "writefile",
  "sn": "SN123456",
  "result": true
}
```

Response (failure):

```
{
  "ret": "writefile",
  "sn": "SN123456",
  "result": false
}
```

7.13 Firmware Upgrade

Firmware upgrades use a dedicated HTTP endpoint (not JSON API).

Endpoint: `POST /api/firmware/upload`

Query Parameters:

Parameter	Description
`name`	Firmware filename (must match device model, e.g., `ai230_.zip` or `.pkg`)
`offset`	Current chunk start offset (bytes)
`total`	Total file size (bytes)

Parameter	Description
`password`	Device password
`md5`	File MD5 checksum (verified on last chunk)

Chunked Transfer Logic:

1. Each POST carries a file chunk as the request body
2. `offset=0` marks the first chunk — device clears `/data/ota` directory
3. When `data.len == 0` and `offset >= total`, MD5 is verified and processing completes
4. Supported formats: `.zip` (system update) and `.pkg` (firmware package)

Success Response: `"true"` (text/plain)

Error Response (HTTP 500):

Message	Description
`error password`	Wrong password
`this file not this machine firmware`	Firmware model mismatch
`Insufficient disk space`	Not enough disk space
`md5 check error`	MD5 verification failed
`need zip or pkg`	Unsupported file format

8. Static Resources & File Downloads

Path	Description
`/photos/LF.jpg`	User enrollment photo (by internal numeric ID)
`/photos/CF_.jpg`	User enrollment photo (by alias ID)
`/logphotos/logimage.jpg`	Attendance log snapshot photo
`/download/report`	Summary attendance report (.xls, valid for 20s after generation)
`/download/rawrecord`	Raw attendance records (.xls, valid for 20s after generation)

> Static files are served from `/data/res/webserver` first; if not found, fallback to `/usr/webserver`.

9. Appendix: backupnum Reference

The `backupnum` field is used in `getuserinfo`, `setuserinfo`, `deleteuser`, `adduser`, `deleteuserfp`, `deleteuserpalm` and related commands to identify which biometric slot to operate on. Push commands (`setuserinfo`) additionally carry the template payload in `record`.

backupnum	Type	Description	record encoding (push / export)	Length after decoding
0-9	Fingerprint	Fingerprint at index 0-9 (up to 10 per user)	Hex + zero run-length compression `(N)`, **not** Base64	810 bytes
10	Password	User password	Number	—
11	Card	RFID card	Number	—
12	Full delete	Delete entire user record	—	—
13	Full delete	Same as 12	—	—
40	Left palm	Left hand palm vein template	Base64	1548 bytes
41	Right palm	Right hand palm vein template	Base64	1548 bytes
50	Face photo	Face JPEG image	Base64 (JPEG)	≤ 512KB
51	Face feature	Face feature vector + 10 face key points	Base64 (2-byte header + 1072-byte feature)	1074 bytes

record quick reference:

- **Fingerprint (0-9):** each byte as 2 lowercase hex digits, with runs of N `0x00` bytes compressed to `(N)`; e.g. `0a1f(2)2b` decodes to `0a 1f 00 00 2b`. The decoded result must be exactly 810 bytes. Minimal testable payload: `(810)`.
- **Palm vein (40/41):** standard Base64 that must decode to exactly 1548 bytes (2064 Base64 characters); minimal testable payload is the Base64 of 1548 `0x00` bytes (2064 `A` characters).
- **Face feature (51):** standard Base64 that must decode to exactly 1074 bytes; the first 2 bytes are the template header (`08 06` on regular platforms, `08 10` on BOARD_CORE_810). Minimal testable payload is `08 06` + 1072 `0x00` bytes (Base64 starting with `CAYA`, 1432 characters).
- **Face photo (50):** export/read only; push photos with the `face` field of `setuserinfo` (Base64 JPEG).

Generating minimal testable payloads:

```
# Fingerprint (810 zero bytes) -> use the literal string (810)
# Palm vein (1548 zero bytes) -> Base64 string of 2064 characters
python3 -c "import base64;print(base64.b64encode(bytes(1548)).decode())"
# Face feature (08 06 + 1072 zero bytes) -> Base64 starting with CAYA
python3 -c "import base64;print(base64.b64encode(b'\x08\x06'+bytes(1072)).decode())"
```

> Fingerprint and palm vein templates are only available on models with the matching hardware (fingerprint module, or palm vein module); on unsupported models the corresponding `backupnum` is ignored.

---

10. Appendix: verifymode Reference

`verifymode` selects the verification method(s) a user may use (the device default uses the same values). Every method — including combinations such as "face + card" — has a fixed value; use one of the values below.

verifymode	Verification method	Notes
0	Any method	Default; no restriction
1	Fingerprint	Requires fingerprint model
2	Password	—
3	Card	—
4	Fingerprint + card	Requires fingerprint model
5	Fingerprint + password	Requires fingerprint model
6	Password + fingerprint	Requires fingerprint model
7	Card + fingerprint	Requires fingerprint model
8	Face	Requires camera model
9	Face + password	Requires camera model
10	Face + card	Requires camera model
11	Card + password	—
12	Face + fingerprint	Requires camera and fingerprint model
13	QR code	Requires QR code support
14	Face + (card or password)	Requires camera model
15	ID card (SFZ)	Requires ID card reader
16	Palm vein	Requires palm vein model
19	1:N automatic match	Set by the device itself; normally not sent by the platform
20	Any biometric of the person	Any biometric or combination of the person
21	Palm vein + face	Requires palm vein model
22	Fingerprint + palm vein	Requires fingerprint and palm vein model
23	Card + palm vein	Requires palm vein model
24	Palm vein + password	Requires palm vein model
25	Card + face + password	Requires camera model

Notes:

- A value of `0` means no restriction and is the factory default; a user without its own setting follows the device default.
- Only use methods the device supports. A method the device does not support has no effect: the user keeps the previous setting and no error is returned.
- Use one of the values listed above (values are independent and cannot be added together); do not use values that are not listed.
- The `mode` field of log and attendance records uses the same values.

Document generated: 2026-05-06

Last updated: 2026-09-22 — added fingerprint / palm vein / face feature (with key points) push examples, record encoding rules and duplicate-check failure codes; corrected the verifymode value reference (see Appendix 10)

Based on webserver.cpp source code analysis