# AiFace BS Communication Protocol Documentation

This document defines the communication specification between AiFace facial recognition devices and backend servers. The core feature is a unified JSON command set that operates over three transport layers: **WebSocket**, **MQTT**, and **HTTP**.

> ⚠️ **Protocol Role Note:** In the BS communication protocol, the **device acts as a client** that actively connects to the backend server; the server acts as the service endpoint, receiving and responding to the device's connections and requests. Therefore, the BS protocol **supports Wide Area Network (WAN) deployment** — devices can communicate with remote servers across network segments and geographic locations.

> 🔴 **Transport Recommendation: WebSocket (WS/WSS) is strongly recommended.** MQTT/MQTTS and HTTP/HTTPS are **not recommended** for the following reasons:
> - **WebSocket (WS/WSS)**: Long-lived connection, server can push commands in real time with lowest latency and minimal resource usage. **This is the preferred solution.**
> - **HTTP/HTTPS**: Short-lived connections with no push capability. Devices must rely on heartbeat polling to fetch commands, which severely consumes device memory and CPU resources, while also placing heavy load on the server. Real-time performance cannot be guaranteed.
> - **MQTT/MQTTS**: Requires deploying an additional MQTT Broker server, increasing deployment complexity and operational costs, with real-time performance inferior to direct WebSocket connections.

---

## 1. Transport Protocol Specifications

### 1.1 WebSocket / WSS (WebSocket Secure)

**Connection URL Format:**

```
ws://<server_ip>:<port>/<path>
wss://<server_ip>:<port>/<path>
```

**Default Configuration:**

- Default WebSocket port: **7788**
- Default WSS port: **433**
- Default path: **pub/chat** (full connection URL example: `ws://192.168.1.100:7788/pub/chat`)
- If port > 7500, defaults to WebSocket mode

**Authentication Flow:**

1. Device initiates TCP/TLS connection to the server
2. On `MG_EV_WS_OPEN` event, device immediately sends the `reg` command (Device Registration)
3. Server responds with `ret="reg"` containing authorization result and time sync

4. If authorization succeeds, `reg_success = TRUE`; device begins normal operations

**Data Format:**

- All messages are JSON text frames (`WEBSOCKET_OP_TEXT`)
- Encoding: UTF-8

**Heartbeat Mechanism:**

- Device sends a PING frame every **10 seconds** when connected
- If no PONG response is received within **30 seconds**, the connection is considered dead and is closed
- Reconnection is attempted automatically by the `timer_fn` loop

**Reconnection Logic:**

- For persistent connections (WS/WSS), reconnection is attempted whenever `s_conn == NULL`
- Network priority: Wired (highest) > WiFi > 4G USB
- If a higher-priority network becomes available, the current connection is closed and re-established

---

## 1.2 MQTT / MQTTS

> ⚠️ **Not Recommended** — Requires deploying an additional MQTT Broker server, increasing deployment complexity and operational costs. Real-time performance is inferior to direct WebSocket connections.

**Connection URL Format:**

```
mqtt://<server_ip>:<port>
mqtts://<server_ip>:<port>
```

**Default Configuration:**

- Default MQTT port: **1883**
- Default MQTTS port: **8883**

**Topic Naming Rules:**

The serial number (``) in topic names is dynamically replaced with the device's unique serial number.

| Direction | Topic Name | Description |
|---|---|---|
| Device subscribes to (receives commands) | `aiface//pub` | Server publishes commands here |
| Device publishes to (sends data) | `aiface//sub` | Device publishes logs, events here |

**Note:** If a custom topic configuration file exists at `/data/res/mqttpwd.json`, the topics defined in that file override the default topics. The file format supports variable substitution with ``:

```
{
  "user": "mqtt_username",
  "pass": "mqtt_password",
  "client_id": "device_client_id",
  "sub": "aiface/<sn>/pub",
  "pub": "aiface/<sn>/sub"
}
```

**QoS Configuration:**

- Default QoS level: **1** (at least once delivery)
- MQTT KeepAlive interval: **60 seconds**

**Authentication:**

- Username/password from `/data/res/mqttpwd.json`
- Optional client_id for broker identification
- MQTT `clean` session flag is set to `true`

**Connection Flow:**

1. Device connects to MQTT broker
2. On `MG_EV_MQTT_OPEN` event, device subscribes to its command topic
3. Device immediately sends `reg` command via publish
4. Messages received on the subscription topic are parsed by `procrecmessage()`

---

## 1.3 HTTP / HTTPS Polling

> ⚠️ **Not Recommended** — Short-lived connections with no push capability. Devices must rely on heartbeat polling to fetch commands, which severely consumes device memory and server resources. Real-time performance cannot be guaranteed.

**Request URL Format:**

```
http://<server_ip>:<port>/<path>
https://<server_ip>:<port>/<path>
```

**Default Configuration:**

- Default HTTP port: **80**
- Default HTTPS port: **443**

**HTTP Request Format:**

```
POST /<path> HTTP/1.1
Host: <server_ip>:<port>
Content-Type: application/json
Content-Length: <length>

<JSON body>
```

**HTTP Response Handling:**

- Device receives server response in the HTTP response body
- After processing, the connection is immediately closed (`mg_close_conn`)

**Polling Interval:**

- Controlled by `DI_SERVER_RESPONSE_TIME` parameter (in seconds)
- Device only initiates a connection when:
- Polling interval has elapsed, OR
- There is data to send (log records, user modifications, QR code scans, PIN input, GPS data, video intercom events)

**Difference from REST API:**

- The BS protocol over HTTP uses the **same JSON command format** as WebSocket and MQTT
- Unlike typical REST APIs, there is no HTTP method routing (GET/POST/PUT/DELETE)
- All operations are performed by sending JSON command objects in the POST body
- The server responds with JSON in the HTTP response body

**Connection Behavior:**

- HTTP connections are **short-lived**: opened, used for one request/response cycle, then closed
- After registration ( `reg_success = TRUE` ), device polls only when data is pending
- Registration is sent immediately after connection is established

**Polling Heartbeat — `checklive` Command:**

Since HTTP/HTTPS uses short-lived connections, the device cannot receive server-pushed commands in real time. Therefore, when idle, the device actively sends a `checklive` command as a polling heartbeat. Upon receiving it, if the server has pending commands to deliver, it returns the pending command directly; if there are no pending commands, it replies with `ret:"checklive"` along with `cloudtime` for time synchronization.

**Device → Server Request:**

```
{
  "cmd": "checklive",
  "sn": "LF00000001",
  "time": "2025-01-01 08: 00: 00"
}
```

**Server → Device Response (with pending command):**

The server returns the pending command directly, for example:

```
{
  "cmd": "setuserinfo",
  "sn": "LF00000001",
  "userid": 1,
  "name": "Zhang San"
}
```

**Server → Device Response (no pending command):**

```
{
  "ret": "checklive",
  "cloudtime": "2025-01-01 08: 00: 00"
}
```

Upon receiving the `checklive` response, the device synchronizes its clock using the `cloudtime` field.

---

# 2. Protocol Differences Summary

| Feature | WebSocket / WSS | HTTP / HTTPS | MQTT / MQTTS |
|---|---|---|---|
| Connection Type | Long-lived persistent | Short-lived per request | Long-lived persistent |
| Heartbeat | PING/PONG frames (10s interval, 30s timeout) | `checklive` command polling | MQTT KeepAlive (60s) |

| Feature | WebSocket / WSS | HTTP / HTTPS | MQTT / MQTTS |
|---|---|---|---|
| Data Sending | `mg_ws_send` | HTTP POST body | `mg_mqtt_pub` |
| Data Receiving | `MG_EV_WS_MSG` event | HTTP response body | `MG_EV_MQTT_MSG` event |
| Subscribe Topic | N/A | N/A | `aiface//pub` |
| Publish Topic | N/A | N/A | `aiface//sub` |
| Registration | Sent on WS open | Sent on HTTP connect | Sent on MQTT open |
| QoS | N/A | N/A | Level 1 (default) |

# 3. JSON Command Format

## 3.1 General Request Format

```
{
  "cmd": "<command_name>",
  "sn": "<device_serial_number>",
  ...additional parameters...
}
```

## 3.2 General Response Format

**Success:**

```
{
  "ret": "<command_name>",
  "sn": "<device_serial_number>",
  "result": true,
  ...additional fields...
}
```

**Failure:**

```
{
  "ret": "<command_name>",
  "sn": "<device_serial_number>",
  "result": false,
  "reason": <error_code>,
  "msg": "<error_message>"
}
```

## 3.3 Error Codes (reason field)

Different commands use different error code systems. Common patterns:

| reason value | Meaning |
|---|---|
| 1 | Generic failure / parameter error |
| 2 | MD5/check error |

| reason value | Meaning |
|---|---|
| 3 | Operation failed |
| 4 | Invalid format |
| 5 | No data / not found |
| 6-12 | Specific validation failures |

# 4. backupnum Reference Table

The `backupnum` field identifies the type of biometric or credential data associated with a user.

| backupnum | Data Type | Encoding | Description |
|---|---|---|---|
| 0-9 | Fingerprint | hex (with `0` compression) | Fingerprint slot N (0=first finger, 1=second, etc.). Requires a fingerprint model. Each finger: ~810 bytes. |
| 10 | Password | number | User PIN/password |
| 11 | Card Number | number | RFID card number |
| 40 | Palm Vein (left) | base64 | Left palm vein. Requires a palm vein model |
| 41 | Palm Vein (right) | base64 | Right palm vein. Requires a palm vein model |
| 50 | Face Photo | base64 (JPG) | Face photo image. Used when `bFace_template = 0` |
| 51 | Face Feature | base64 (02 30 + feat) | Face feature template. Used when `bFace_template = 1` |

**Fingerprint 0-Compression Format:**

Fingerprint data uses a special compression scheme. Consecutive `0x00` bytes are encoded as `(N)` where N is the count:

- `ff(5)aabb` decodes to: `ff 00 00 00 00 00 aa bb`

# 5. Device Registration — `reg` Command

## 5.1 Device → Server (Registration Request)

Also known as `Active_send_reg`. The device sends this command automatically immediately after establishing a connection (WS open, HTTP connect, or MQTT open).

**Request Example:**

```
{
  "cmd": "reg",
  "sn": "LF00000001",
  "devinfo": {
    "modelname": "AiFace",
    "manufacturer": "AiFace Inc.",
    "usersize": 10000,
    "facesize": 10000,
    "avatarsize": 0,
    "fpsize": 10000,
    "palmsize": 10000,
    "cardsize": 10000,
    "pwdsize": 10000,
    "logsize": 300000,
    "useduser": 100,
    "usedface": 50,
    "usedavatar": 0,
    "usedfp": 20,
    "usedpalm": 0,
    "usedcard": 10,
    "usedpwd": 5,
    "usedlog": 1000,
    "usednewlog": 500,
    "usedrtlog": 3,
    "netinuse": 1,
    "usb4g": 0,
    "fpalgo": "thbio3.0",
    "firmware": "1.0.0",
    "time": "2025-01-01 08: 00: 00",
    "intercom": 0,
    "floors": 0,
    "acces_stimes": 0,
    "charid": 0,
    "useosdp": 0,
    "facetemplate": 0,
    "dislanguage": 0,
    "curip": "192.168.001.100",
    "ntp": 1,
    "timezone": "UTC+8",
    "cpu_temp": 45,
    "bandwidth_up": "100Kbps",
    "bandwidth_down": "1Mbps",
    "battery": 100,
    "disksize": 1024.0,
    "useddisk": 512.0,
    "mac": "AA-BB-CC-DD-EE-FF"
  }
}
```


### devinfo Field Descriptions:

| Field | Type | Description |
|---|---|---|
| modelname | string | Device model name (e.g., "AiFace") |
| manufacturer | string | Manufacturer name |
| usersize | number | Maximum number of users |
| facesize | number | Maximum face templates |
| avatarsize | number | Maximum avatar photos |
| fpsize | number | Maximum fingerprints |
| palmsize | number | Maximum palm veins |

| Field | Type | Description |
| --- | --- | --- |
| cardsize | number | Maximum card numbers |
| pwdsize | number | Maximum passwords |
| logsize | number | Maximum log records |
| useduser | number | Currently enrolled users |
| usedface | number | Used face templates |
| usedavatar | number | Used avatar photos |
| usedfp | number | Used fingerprints |
| usedpalm | number | Used palm veins |
| usedcard | number | Registered cards |
| usedpwd | number | Registered passwords |
| usedlog | number | Stored log records |
| usednewlog | number | New (unread) log records |
| usedrtlog | number | Real-time log records |
| netinuse | number | Network in use (1=wired, 2=wifi, 3=4g) |
| usb4g | number | USB 4G module status (0=not used, 1=used) |
| fpalgo | string | Fingerprint algorithm version |
| firmware | string | Firmware version |
| time | string | Device current time "YYYY-MM-DD HH:MM:SS" |
| intercom | number | Video intercom capability (0=none, 1=supported) |
| floors | number | Number of floors (for elevator control) |
| acces_stimes | number | Access times limit |
| charid | number | Character ID / encoding |
| useosdp | number | OSDP protocol support |
| facetemplate | number | Face template format (0=photo, 1=feature) |
| dislanguage | number | Display language setting |
| curip | string | Current IP address |
| ntp | number | NTP enabled flag |
| timezone | string | Timezone (e.g., "UTC+8") |
| cpu_temp | number | CPU temperature (Celsius) |
| bandwidth_up | string | Upload bandwidth |
| bandwidth_down | string | Download bandwidth |
| battery | number | Battery level (%) |
| disksize | number | Storage size (MB) |
| useddisk | number | Used storage (MB) |
| mac | string | MAC address |

### 5.2 Server → Device (Registration Response)

**Success Response:**

```
{
  "ret": "reg",
  "tryseconds": 10,
  "result": true,
  "cloudtime": "2025-01-01 08: 00: 00",
  "nosenduser": false,
  "nosendlog": false,
  "nosendimage": false
}
```

| Field | Type | Description |
|---|---|---|
| tryseconds | number | Periodic re-registration interval (seconds) |
| result | boolean | Authorization result |
| cloudtime | string | Server time for synchronization |
| nosenduser | boolean | If true, device should not send user change notifications |
| nosendlog | boolean | If true, device should not send log records |
| nosendimage | boolean | If true, device should not send images in logs |

**Failure Response:**

```
{
  "ret": "reg",
  "tryseconds": 30,
  "result": false,
  "reason": "Device not authorized"
}
```

**Periodic Re-registration:**

- Device re-sends `reg` command every `tryseconds` seconds (default: 300s / 5 minutes) while `reg_success = TRUE`

---

# 6. User Management Commands

### 6.1 setuserinfo — Set User Information

Downloads and sets user profile information including biometrics.

**Request Parameters:**

| Parameter | Type | Description |
|---|---|---|
| enrollid | number/string | User ID |
| aliasid | string | User alias ID |
| birthday | string | Birthday "MM-DD" or "MM-DD*" |
| verifymode | number | Verification mode (values in Appendix B) |

| Parameter | Type | Description |
| --- | --- | --- |
| card | number/string | RFID card number |
| pwd | number | Password |
| shiftid | number | Shift ID |
| zoneid | number | Time zone ID |
| groupid | number | Group ID |
| access_times | number | Maximum access times (max 30000) |
| postid | number | Position/Post ID |
| department | string | Department name (auto-converted to postid) |
| enable | number | Enable flag (1=enabled, 0=disabled) |
| starttime | string | Validity start "YYYY-MM-DD HH:MM:SS" |
| endtime | string | Validity end "YYYY-MM-DD HH:MM:SS" |
| backupnum | number | Biometric data type (see backupnum table) |
| admin | number | Admin privilege level |
| name | string | User name |
| userprofile | string | Custom user profile (JSON string) |
| record | string/number | Biometric data (base64 or hex per backupnum type) |

### Request Example:

```
{
  "cmd": "setuserinfo",
  "sn": "LF00000001",
  "enrollid": 123,
  "aliasid": "USR001",
  "name": "John Doe",
  "birthday": "01-15",
  "verifymode": 0,
  "card": 12345,
  "pwd": 0,
  "shiftid": 0,
  "zoneid": 0,
  "groupid": 0,
  "access_times": 0,
  "postid": 1,
  "department": "Engineering",
  "enable": 1,
  "starttime": "2025-01-01 00: 00: 00",
  "endtime": "2025-12-31 23: 59: 59",
  "backupnum": 50,
  "admin": 0,
  "userprofile": "{}",
  "record": "base64_encoded_face_data"
}
```

### Success Response:

```
{
  "ret": "setuserinfo",
  "enrollid": 123,
```

```
  "sn": "LF00000001",
  "backupnum": 50,
  "result": true,
  "aliasid": "USR001"
}
```

**Failure Response:**

```
{
  "ret": "setuserinfo",
  "enrollid": 123,
  "sn": "LF00000001",
  "backupnum": 50,
  "result": false,
  "reason": 1,
  "msg": "space overflow",
  "aliasid": "USR001"
}
```

---

## 6.2 deleteuser — Delete User

Deletes a user or specific biometric data for a user.

**Request Parameters:**

| Parameter | Type | Description |
|---|---|---|
| enrollid | number/string | User ID |
| aliasid | string | User alias ID |
| backupnum | number | Data type to delete |

**backupnum Values for Deletion:**

- `12` or `13` : Delete entire user (all biometrics and photo)
- `10` : Delete password only
- `11` : Delete card only
- `50` : Delete face photo
- `51` : Delete face feature
- `0-9` : Delete specific fingerprint
- `40` : Delete left palm vein
- `41` : Delete right palm vein

**Request Example:**

```
{
  "cmd": "deleteuser",
  "sn": "LF00000001",
  "enrollid": 123,
  "aliasid": "USR001",
  "backupnum": 12
}
```

**Success Response:**

```
{
  "ret": "deleteuser",
  "enrollid": 123,
  "sn": "LF00000001",
  "backupnum": 12,
  "result": true,
  "aliasid": "USR001"
}
```

---

## 6.3 cleanuser — Clear All Users

Deletes all users from the device. No parameters.

**Request Example:**

```
{
  "cmd": "cleanuser",
  "sn": "LF00000001"
}
```

**Success Response:**

```
{
  "ret": "cleanuser",
  "sn": "LF00000001",
  "result": true
}
```

---

## 6.4 setusername — Batch Set User Names

**Request Parameters:**

| Parameter | Type | Description |
|---|---|---|
| count | number | Number of records |
| birthday | string | Birthday (applied globally) |
| record | array | Array of `{enrollid, aliasid, name, verifymode}` |

**Request Example:**

```
{
  "cmd": "setusername",
  "sn": "LF00000001",
  "count": 2,
  "birthday": "06-15",
  "record": [
    {"enrollid": 100, "aliasid": "U100", "name": "Alice", "verifymode": 0},
    {"enrollid": 101, "aliasid": "U101", "name": "Bob", "verifymode": 0}
  ]
}
```

**Success Response:**

```
{
  "ret": "setusername",
  "sn": "LF00000001",
  "result": true
}
```

### Failure Response:

```
{
  "ret": "setusername",
  "sn": "LF00000001",
  "result": false,
  "reason": 1,
  "msg": "set username fail"
}
```

---

## 6.5 getusername — Get User Name

### Request Parameters:

| Parameter | Type | Description |
|---|---|---|
| enrollid | number/string | User ID |
| aliasid | string | User alias ID |

### Request Example:

```
{
  "cmd": "getusername",
  "sn": "LF00000001",
  "enrollid": 123,
  "aliasid": "USR001"
}
```

### Success Response:

```
{
  "ret": "getusername",
  "sn": "LF00000001",
  "result": true,
  "record": "John Doe"
}
```

### Failure Response:

```
{
  "ret": "getusername",
  "sn": "LF00000001",
  "result": false,
  "reason": 1,
  "msg": "can not find this name"
}
```

---

## 6.6 getuserlist — Get User ID List (Paginated)

Retrieves a paginated list of user IDs. Use `stn=1` to reset pagination and start a new query.

**Request Parameters:**

| Parameter | Type | Description |
|---|---|---|
| stn | number | `1` = start new query (reset pagination index) |

**Pagination Behavior:**

- `stn=1` : Resets internal index to 0 and starts a new query
- Each response contains up to 100 records
- Internal index auto-increments after each response
- Repeat the request to get the next page

**Request Example:**

```
{
  "cmd": "getuserlist",
  "sn": "LF00000001",
  "stn": 1
}
```

**Success Response (with data):**

```
{
  "ret": "getuserlist",
  "sn": "LF00000001",
  "result": true,
  "count": 500,
  "from": 0,
  "to": 100,
  "record": [
    {"enrollid": 123, "admin": 0, "backupnum": 50, "enable": 1, "aliasid": "USR001"},
    {"enrollid": 124, "admin": 1, "backupnum": 51, "enable": 1, "aliasid": "USR002"}
  ]
}
```

**Success Response (empty):**

```
{
  "ret": "getuserlist",
  "sn": "LF00000001",
  "result": true,
  "count": 0,
  "from": 0,
  "to": 0,
  "record": []
}
```

---

## 6.7 getuserids — Get All User IDs

Returns all user IDs in a single response (no pagination).

**Request Parameters:**

None.

**Request Example:**

```
{
  "cmd": "getuserids",
  "sn": "LF00000001"
}
```

**Success Response:**

```
{
  "ret": "getuserids",
  "sn": "LF00000001",
  "result": true,
  "count": 100,
  "record": ["123", "456", "USR001", "USR002"]
}
```

> Returns user IDs as strings. Alias IDs are used when available, otherwise numeric ID strings.

---

## 6.8 getunuserdid — Get Next Available User ID

Returns the next available unused user ID.

**Request Parameters:**

None.

**Request Example:**

```
{
  "cmd": "getunuserdid",
  "sn": "LF00000001"
}
```

**Success Response:**

```
{
  "ret": "getunuserdid",
  "sn": "LF00000001",
  "result": true,
  "enrollid": "100"
}
```

---

## 6.9 checkuserid — Check If User ID Exists

**Request Parameters:**

| Parameter | Type | Description |
|---|---|---|
| enrollid | number/string | User ID or alias ID to check |

**Request Example:**

```
{
  "cmd": "checkuserid",
  "sn": "LF00000001",
  "enrollid": 123
}
```

**Success Response (exists):**

```
{
  "ret": "checkuserid",
  "sn": "LF00000001",
  "result": true,
  "exists": true
}
```

**Success Response (not found):**

```
{
  "ret": "checkuserid",
  "sn": "LF00000001",
  "result": true,
  "exists": false
}
```

---

## 6.10 getuserinfo — Get User Detailed Information

**Request Parameters:**

| Parameter | Type | Description |
|---|---|---|
| enrollid | number/string | User ID |
| aliasid | string | User alias ID |
| backupnum | number | (Optional) Specific biometric type to retrieve |

**Without backupnum** — returns full user profile (without raw biometric data):

**Request Example:**

```
{
  "cmd": "getuserinfo",
  "sn": "LF00000001",
  "enrollid": 123,
  "aliasid": "USR001"
}
```

**Success Response:**

```
{
  "ret": "getuserinfo",
  "result": true,
  "sn": "LF00000001",
  "enrollid": 123,
  "name": "John Doe",
  "department": "Engineering",
  "admin": 0,
  "aliasid": "USR001",
```

```
  "photourl": "/photos/LF00000123.jpg",
  "card": 12345,
  "pwd": 0,
  "fpflag": 3,
  "fpcnt": 2,
  "palmflag": 0,
  "palmcnt": 0,
  "faceflag": 1,
  "enable": 1,
  "birthday": "01-15",
  "shiftid": 0,
  "verifymode": 0,
  "zoneid": 0,
  "groupid": 0,
  "access_times": 0,
  "postid": 1,
  "starttime": "2025-01-01 00: 00: 00",
  "endtime": "2025-12-31 23: 59: 59",
  "userprofile": "{}"
}
```

**With backupnum** — returns raw biometric data:

| backupnum | Returns |
|---|---|
| 0-9 | Fingerprint data (hex) |
| 10 | Password (number) |
| 11 | Card number (number) |
| 50 | Face photo (base64 JPG) |
| 51 | Face feature (base64) |

### Request Example (face photo):

```
{
  "cmd": "getuserinfo",
  "sn": "LF00000001",
  "enrollid": 123,
  "aliasid": "USR001",
  "backupnum": 50
}
```

### Success Response:

```
{
  "ret": "getuserinfo",
  "result": true,
  "sn": "LF00000001",
  "enrollid": 123,
  "name": "John Doe",
  "backupnum": 50,
  "record": "base64_encoded_face_photo_data"
}
```

### Failure Response:

```
{
  "ret": "getuserinfo",
  "result": false,
  "sn": "LF00000001",
```

```
  "enrollid": 123,
  "reason": 1,
  "msg": "can not find this user",
  "aliasid": "USR001"
}
```

---

## 6.11 getallusers — Get All User Data (Paginated, Multi-Record)

Returns all user data in paginated format. Each user may return multiple records (one per biometric type). Use `stn=true` to start a new query.

**Request Parameters:**

| Parameter | Type | Description |
|---|---|---|
| stn | boolean | `true` = start new query (reset pagination) |

**Request Example:**

```
{
  "cmd": "getallusers",
  "sn": "LF00000001",
  "stn": true
}
```

**Response (password record, backupnum=10):**

```
{
  "ret": "getallusers",
  "sn": "LF00000001",
  "result": true,
  "count": 500,
  "index": 0,
  "enrollid": 123,
  "backupnum": 10,
  "admin": 0,
  "enable": 1,
  "record": 12345,
  "aliasid": "USR001"
}
```

**Response (card record, backupnum=11):**

```
{
  "ret": "getallusers",
  "sn": "LF00000001",
  "result": true,
  "count": 500,
  "index": 1,
  "enrollid": 123,
  "backupnum": 11,
  "admin": 0,
  "enable": 1,
  "record": 1234567890,
  "aliasid": "USR001"
}
```

**Response (face photo, backupnum=50):**

```
{
  "ret": "getallusers",
  "sn": "LF00000001",
  "result": true,
  "count": 500,
  "index": 2,
  "enrollid": 123,
  "backupnum": 50,
  "admin": 0,
  "enable": 1,
  "record": "base64_jpg_data",
  "aliasid": "USR001"
}
```


**Response (face feature, backupnum=51):**


```
{
  "ret": "getallusers",
  "sn": "LF00000001",
  "result": true,
  "count": 500,
  "index": 3,
  "enrollid": 123,
  "backupnum": 51,
  "admin": 0,
  "enable": 1,
  "record": "base64_feat_data",
  "aliasid": "USR001"
}
```


**Response (fingerprint, backupnum=0):**


```
{
  "ret": "getallusers",
  "sn": "LF00000001",
  "result": true,
  "count": 500,
  "index": 5,
  "enrollid": 123,
  "backupnum": 0,
  "admin": 0,
  "enable": 1,
  "record": "hex_fp_data",
  "aliasid": "USR001"
}
```


**Response (palm vein, backupnum=40):**


```
{
  "ret": "getallusers",
  "sn": "LF00000001",
  "result": true,
  "count": 500,
  "index": 10,
  "enrollid": 123,
  "backupnum": 40,
  "admin": 0,
  "enable": 1,
  "record": "base64_palm_data",
  "aliasid": "USR001"
}
```

**Failure Response:**

```
{
  "ret": "getallusers",
  "sn": "LF00000001",
  "backupnum": 50,
  "result": false,
  "reason": 1,
  "msg": "this photo is too big"
}
```

---

## 6.12 adduser — Add User (Device-Side Registration)

Initiates user enrollment on the device. The device processes biometric capture and reports progress.

**Request Parameters:**

| Parameter | Type | Description |
|---|---|---|
| cancel | boolean | `true` = cancel the enrollment process |
| enrollid | number/string | User ID |
| aliasid | string | User alias ID |
| backupnum | number | Enrollment type: `0-9` (fingerprint), `10` (password), `11` (card), `40/41` (palm), `50` (face) |
| admin | number | Admin privilege level |
| flag | number | Flags |
| name | string | User name |

**Request Example (start enrollment):**

```
{
  "cmd": "adduser",
  "sn": "LF00000001",
  "enrollid": 123,
  "aliasid": "USR001",
  "backupnum": 50,
  "admin": 0,
  "flag": 0,
  "name": "John Doe"
}
```

**Success Response:**

```
{
  "ret": "adduser",
  "enrollid": 123,
  "sn": "LF00000001",
  "backupnum": 50,
  "result": true,
  "aliasid": "USR001"
}
```

**Failure Response:**

```
{
  "ret": "adduser",
  "enrollid": 123,
  "sn": "LF00000001",
  "backupnum": 50,
  "result": false,
  "reason": 1,
  "msg": "userid can not empty"
}
```

**Cancel Enrollment:**

```
{
  "cmd": "adduser",
  "sn": "LF00000001",
  "cancel": true
}
```

**Cancel Response:**

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

---

## 6.13 checkregstatus — Check Device-Side Registration Status

Queries the current enrollment progress on the device.

**Request Parameters:**

None.

**Request Example:**

```
{
  "cmd": "checkregstatus",
  "sn": "LF00000001"
}
```

**Success Response:**

```
{
  "ret": "checkregstatus",
  "sn": "LF00000001",
  "result": true,
  "status": 1,
  "msg": "Enrollment progress message",
  "image": "base64..."
}
```

> `status` : `0` = waiting, `>=0` = success (value corresponds to backupnum), `<0` = failure

---

## 6.14 canceladduser — Cancel User Enrollment

Cancels the current user enrollment process. No parameters.

**Request Example:**

```
{
  "cmd": "canceladduser",
  "sn": "LF00000001"
}
```

**Success Response:**

```
{
  "ret": "canceladduser",
  "result": true
}
```

---

## 6.15 enableuser — Enable/Disable User

**Request Parameters:**

| Parameter | Type | Description |
|---|---|---|
| enrollid | number/string | User ID |
| aliasid | string | User alias ID |
| enflag | boolean | `true` = enable, `false` = disable |

**Request Example:**

```
{
  "cmd": "enableuser",
  "sn": "LF00000001",
  "enrollid": 123,
  "aliasid": "USR001",
  "enflag": false
}
```

**Success Response:**

```
{
  "ret": "enableuser",
  "enrollid": 123,
  "sn": "LF00000001",
  "result": true,
  "aliasid": "USR001"
}
```

---

## 6.16 getuserprofile — Get User Custom Profile

**Request Parameters:**

| Parameter | Type | Description |
| --- | --- | --- |
| enrollid | number/string | User ID |
| aliasid | string | User alias ID |

### Request Example:

```
{
  "cmd": "getuserprofile",
  "sn": "LF00000001",
  "enrollid": 123,
  "aliasid": "USR001"
}
```

### Success Response:

```
{
  "ret": "getuserprofile",
  "sn": "LF00000001",
  "result": true,
  "record": "{...json...}"
}
```

### Failure Response:

```
{
  "ret": "getuserprofile",
  "sn": "LF00000001",
  "result": false,
  "reason": 1,
  "msg": "userprofile can not empty"
}
```

---

## 6.17 setuserprofile — Set User Custom Profile

### Request Parameters:

| Parameter | Type | Description |
| --- | --- | --- |
| enrollid | number/string | User ID |
| aliasid | string | User alias ID |
| profile | string | Configuration JSON string |
| duration | number | Display duration (only when nID=0) |
| color | number | Color (only when nID=0) |

### Request Example:

```
{
  "cmd": "setuserprofile",
  "sn": "LF00000001",
  "enrollid": 123,
  "aliasid": "USR001",
  "profile": "{\"key\": \"value\"}",
  "duration": 5,
```

```
  "color": 1
}
```

**Success Response:**

```
{
  "ret": "setuserprofile",
  "enrollid": 123,
  "sn": "LF00000001",
  "result": true,
  "aliasid": "USR001"
}
```

**Failure Response:**

```
{
  "ret": "setuserprofile",
  "enrollid": 123,
  "sn": "LF00000001",
  "result": false,
  "reason": 1,
  "msg": "setuserprofile fail"
}
```

---

# 7. Log Management Commands

## 7.1 getnewlog — Get New Records

Retrieves new (unread) attendance/access log records with pagination.

**Request Parameters:**

| Parameter | Type | Description |
|---|---|---|
| stn | number | `1` = start new query (reset index) |
| logphoto | number | `1` = include photo data in logs |

**Request Example:**

```
{
  "cmd": "getnewlog",
  "sn": "LF00000001",
  "stn": 1,
  "logphoto": 1
}
```

**Success Response:**

```
{
  "ret": "getnewlog",
  "sn": "LF00000001",
  "result": true,
  "count": 500,
  "from": 0,
  "to": 100,
```

```
  "record": [
    {
      "enrollid": 123,
      "name": "John Doe",
      "time": "2025-01-01 08: 00: 00",
      "mode": 1,
      "inout": 0,
      "event": 0,
      "aliasid": "USR001",
      "image": "base64...",
      "note": ""
    }
  ]
}
```

**Empty Response:**

```
{
  "ret": "getnewlog",
  "sn": "LF00000001",
  "result": true,
  "count": 0,
  "from": 0,
  "to": 0,
  "record": []
}
```

**Record Fields:**

| Field | Type | Description |
|---|---|---|
| enrollid | number | User ID |
| name | string | User name |
| time | string | Event time "YYYY-MM-DD HH:MM:SS" |
| mode | number | Verification mode (1=face, 2=fingerprint, 4=card, 8=password, etc.) |
| inout | number | Direction (0=in, 1=out) |
| event | number | Event type |
| aliasid | string | User alias ID |
| image | string | Photo base64 (if logphoto=1) |
| note | string | Note/comment |

---

## 7.2 getalllog — Get All Records

Retrieves all log records with optional filtering and pagination.

**Request Parameters:**

| Parameter | Type | Description |
|---|---|---|
| stn | number | `1` = start new query |
| enrollid | number/string | (Optional) Filter: number = by user ID; string = by alias ID (aliasid) |

| Parameter | Type | Description |
|---|---|---|
| from | string | Start date "YYYY-MM-DD" |
| to | string | End date "YYYY-MM-DD" |
| logphoto | number | `1` = include photo data |

**Request Example:**

```
{
  "cmd": "getalllog",
  "sn": "LF00000001",
  "stn": 1,
  "enrollid": 123,
  "from": "2025-01-01",
  "to": "2025-12-31",
  "logphoto": 0
}
```

**Success Response:**

```
{
  "ret": "getalllog",
  "sn": "LF00000001",
  "result": true,
  "count": 1000,
  "from": 0,
  "to": 100,
  "record": [
    {
      "enrollid": 123,
      "name": "John Doe",
      "time": "2025-01-01 08: 00: 00",
      "mode": 1,
      "inout": 0,
      "event": 0,
      "aliasid": "USR001",
      "image": "base64...",
      "note": ""
    }
  ]
}
```

---

## 7.3 cleanlog — Clear All Logs

Deletes all log records from the device. No parameters.

**Request Example:**

```
{
  "cmd": "cleanlog",
  "sn": "LF00000001"
}
```

**Success Response:**

```
{
  "ret": "cleanlog",
```

```
  "sn": "LF00000001",
  "result": true
}
```

---

### 7.4 cleanlogphoto — Clear Log Photos

Deletes all photos associated with log records. No parameters.

**Request Example:**

```
{
  "cmd": "cleanlogphoto",
  "sn": "LF00000001"
}
```

**Success Response:**

```
{
  "ret": "cleanlogphoto",
  "sn": "LF00000001",
  "result": true
}
```

---

## 8. Device Management Commands

### 8.1 getdevinfo — Get Device System Parameters

Reads all system parameters. No parameters.

**Request Example:**

```
{
  "cmd": "getdevinfo",
  "sn": "LF00000001"
}
```

**Success Response:**

```
{
  "ret": "getdevinfo",
  "sn": "LF00000001",
  "result": true,
  "param1": "value1",
  "param2": "value2"
}
```

> Returns all parameters from DI_FIRST to DI_END enumeration as key-value pairs.

---

## 8.2 setdevinfo — Set Device System Parameters

Sets one or more system parameters.

**Request Parameters:**

| Parameter | Type | Description |
|---|---|---|
| nowtime | number | Current Unix timestamp (seconds) |
| (dynamic) | varies | Any parameter from DI_FIRST~DI_END enumeration as key-value pairs |

**Request Example:**

```
{
  "cmd": "setdevinfo",
  "sn": "LF00000001",
  "nowtime": 1735689600,
  "DI_SERVERIP": "192.168.1.100",
  "DI_SERVERPORT": 7788
}
```

**Success Response:**

```
{
  "ret": "setdevinfo",
  "sn": "LF00000001",
  "result": true
}
```

> **Note:** Modifying domain name, IP, or port parameters triggers a connection re-establishment ( `need_close_connection = TRUE` ).

---

## 8.3 getdevcap — Get Device Capacity

Returns device storage capacity information. No parameters.

**Request Example:**

```
{
  "cmd": "getdevcap",
  "sn": "LF00000001"
}
```

**Success Response:**

```
{
  "ret": "getdevcap",
  "sn": "LF00000001",
  "result": true,
  "usersize": 10000,
  "facesize": 10000,
  "avatarsize": 0,
  "fpsize": 10000,
  "palmsize": 10000,
  "cardsize": 10000,
```

```
  "pwdsize": 10000,
  "logsize": 300000,
  "useduser": 100,
  "usedface": 50,
  "usedavatar": 0,
  "usedfp": 20,
  "usedpalm": 0,
  "usedcard": 10,
  "usedpwd": 5,
  "usedlog": 1000,
  "usednewlog": 500,
  "usedrtlog": 3
}
```

| Field | Description |
| --- | --- |
| usersize | Maximum number of users |
| facesize | Maximum face templates |
| avatarsize | Maximum avatar photos |
| fpsize | Maximum fingerprints |
| palmsize | Maximum palm veins |
| cardsize | Maximum card numbers |
| pwdsize | Maximum passwords |
| logsize | Maximum log records |
| used* | Currently used count for each type |
| usednewlog | Number of unread new logs |
| usedrtlog | Real-time log queue size |

## 8.4 gettime — Get Device Time

Gets the current device time. No parameters.

**Request Example:**

```
{
  "cmd": "gettime",
  "sn": "LF00000001"
}
```

**Success Response:**

```
{
  "ret": "gettime",
  "sn": "LF00000001",
  "time": "2025-01-01 08: 00: 00"
}
```

## 8.5 settime — Set Device Time

**Request Parameters:**

| Parameter | Type | Description |
|---|---|---|
| cloudtime | string | Cloud/server time "YYYY-MM-DD HH:MM:SS" |
| nowtime | number | Unix timestamp in seconds |

**Request Example:**

```
{
  "cmd": "settime",
  "sn": "LF00000001",
  "cloudtime": "2025-01-01 08: 00: 00",
  "nowtime": 1735689600
}
```

**Success Response:**

```
{
  "ret": "settime",
  "sn": "LF00000001",
  "result": true
}
```

**Failure Response:**

```
{
  "ret": "settime",
  "sn": "LF00000001",
  "result": false,
  "reason": 1,
  "msg": "time format wrong"
}
```

---

## 8.6 initsys — Factory Reset

Performs a factory reset: clears all users and logs, restores default settings, backs up WiFi configuration. No parameters.

**Request Example:**

```
{
  "cmd": "initsys",
  "sn": "LF00000001"
}
```

**Success Response:**

```
{
  "ret": "initsys",
  "sn": "LF00000001",
  "result": true
}
```

## 8.7 initmenu — Reset Menu Settings

Restores default menu settings. Preserves language and backs up WiFi configuration. No parameters.

**Request Example:**

```
{
  "cmd": "initmenu",
  "sn": "LF00000001"
}
```

**Success Response:**

```
{
  "ret": "initmenu",
  "sn": "LF00000001",
  "result": true
}
```

## 8.8 cleandatebase — Clear Database

Clears the entire database. Device must be rebooted after this operation. No parameters.

**Request Example:**

```
{
  "cmd": "cleandatebase",
  "sn": "LF00000001"
}
```

**Success Response:**

```
{
  "ret": "cleandatebase",
  "sn": "LF00000001",
  "result": true
}
```

## 8.9 cleanadmin — Clear All Administrators

Removes all administrator privileges. No parameters.

**Request Example:**

```
{
  "cmd": "cleanadmin",
  "sn": "LF00000001"
}
```

**Success Response:**

```
{
  "ret": "cleanadmin",
  "sn": "LF00000001",
  "result": true
}
```

---

## 8.10 cleaninactiveuser — Clear Inactive Users

Removes users who have not been active (no attendance records). No parameters.

**Request Example:**

```
{
  "cmd": "cleaninactiveuser",
  "sn": "LF00000001"
}
```

**Success Response:**

```
{
  "ret": "cleaninactiveuser",
  "sn": "LF00000001",
  "result": true
}
```

---

## 8.11 reboot — Reboot Device

Triggers an immediate device reboot. No response is sent (device restarts).

**Request Example:**

```
{
  "cmd": "reboot",
  "sn": "LF00000001"
}
```

**Behavior:** No response. Device reboots immediately.

---

## 8.12 disabledevice — Disable Device Recognition

Pauses all face, card, fingerprint, and palm detection. No parameters.

**Request Example:**

```
{
  "cmd": "disabledevice",
  "sn": "LF00000001"
}
```

**Success Response:**

```
{
  "ret": "disabledevice",
  "sn": "LF00000001",
  "result": true
}
```

---

## 8.13 enabledevice — Enable Device Recognition

Resumes all face, card, fingerprint, and palm detection. No parameters.

**Request Example:**

```
{
  "cmd": "enabledevice",
  "sn": "LF00000001"
}
```

**Success Response:**

```
{
  "ret": "enabledevice",
  "sn": "LF00000001",
  "result": true
}
```

---

## 8.14 getreg — Get Device Registration Information

Retrieves full device registration and capability information. No parameters.

**Request Example:**

```
{
  "cmd": "getreg",
  "sn": "LF00000001"
}
```

**Success Response:**

```
{
  "ret": "getreg",
  "sn": "LF00000001",
  "result": true,
  "devinfo": {
    "modelname": "AiFace",
    "manufacturer": "AiFace Inc.",
    "usersize": 10000,
    "facesize": 10000,
    "avatarsize": 0,
    "fpsize": 10000,
    "palmsize": 10000,
    "cardsize": 10000,
    "pwdsize": 10000,
    "logsize": 300000,
    "useduser": 100,
    "usedface": 50,
    "usedavatar": 0,
```

```
    "usedfp": 20,
    "usedpalm": 0,
    "usedcard": 10,
    "usedpwd": 5,
    "usedlog": 1000,
    "usednewlog": 500,
    "usedrtlog": 3,
    "netinuse": 1,
    "usb4g": 0,
    "fpalgo": "thbio3.0",
    "firmware": "1.0.0",
    "time": "2025-01-01 08: 00: 00",
    "intercom": 0,
    "floors": 0,
    "acces_stimes": 0,
    "charid": 0,
    "useosdp": 0,
    "facetemplate": 0,
    "dislanguage": 0,
    "curip": "192.168.001.100",
    "ntp": 1,
    "timezone": "UTC+8",
    "cpu_temp": 45,
    "bandwidth_up": "100Kbps",
    "bandwidth_down": "1Mbps",
    "battery": 100,
    "disksize": 1024.0,
    "useddisk": 512.0,
    "mac": "AA-BB-CC-DD-EE-FF"
  }
}
```


---

## 8.15 checklive — Heartbeat Check (Server-Issued)

A server-initiated heartbeat command to verify device connectivity. Device responds to confirm it is alive and provide server time for synchronization.

**Server → Device Request:**

```
{
  "cmd": "checklive",
  "sn": "LF00000001"
}
```

**Device → Server Response:**

```
{
  "ret": "checklive",
  "sn": "LF00000001",
  "result": true
}
```

---

# 9. Access Control Commands

## 9.1 opendoor — Remote Door Open

Remotely unlocks the door. Optionally sends user ID via Wiegand and displays a message.

**Request Parameters:**

| Parameter | Type | Description |
|---|---|---|
| enrollid | number | User ID (sent via Wiegand output) |
| floor | number | Floor number (requires elevator control enabled) |
| msg | string | Display message on device screen |

**Request Example:**

```
{
  "cmd": "opendoor",
  "sn": "LF00000001",
  "enrollid": 123,
  "floor": 0,
  "msg": "Welcome"
}
```

**Success Response:**

```
{
  "ret": "opendoor",
  "sn": "LF00000001",
  "result": true
}
```

## 9.2 lockctrl — Door Lock Control

Controls the door lock state directly.

**Request Parameters:**

| Parameter | Type | Description |
|---|---|---|
| fuc | number | `1` = unlock/open, `2` = lock/close |

**Request Example:**

```
{
  "cmd": "lockctrl",
  "sn": "LF00000001",
  "fuc": 1
}
```

**Success Response:**

```
{
  "ret": "lockctrl",
  "sn": "LF00000001",
  "result": true
}
```

## 9.3 getdoorstatus — Get Door Status

Returns the current door sensor status. No parameters.

**Request Example:**

```
{
  "cmd": "getdoorstatus",
  "sn": "LF00000001"
}
```

**Success Response:**

```
{
  "ret": "getdoorstatus",
  "doorstatus": 0,
  "sn": "LF00000001",
  "result": true
}
```

> `doorstatus`: `0` = closed, `1` = open (depends on door sensor configuration)

## 9.4 setdevlock — Set Door Access Parameters

Configures door access control parameters including timing, zones, and Wiegand settings.

**Request Parameters:**

| Parameter | Type | Description |
|---|---|---|
| opendelay | number | Door open delay (seconds) |
| doorsensor | number | Door sensor type |
| alarmdelay | number | Alarm trigger delay |
| InputAlarm | number | Fire alarm input |
| antpass | number | Anti-passback enabled |
| interlock | number | Interlock mode |
| mutiopen | number | Multi-person unlock requirement |
| tryalarm | number | Failed attempt alarm threshold |
| punchtimes | number | Punch/clock times |
| access_times | number | Maximum access times |
| serververifytimeout | number | Server verification timeout (seconds) |

| Parameter | Type | Description |
|---|---|---|
| tamper | number | Tamper detection enabled |
| wgformat | number | Wiegand output format: 0=26bit, 1=34bit, 2=42bit, 3=50bit, 4=58bit, 5=66bit, 6=37bit |
| wgoutput | number | Wiegand output mode |
| cardformat | number | Card display format |
| wgport | number | Wiegand port |
| dayzone | array | Date time zones `[{"day": [{"section": "HH:MM~HH:MM"}]}]` |
| weekzone | array | Weekly time zones `[{"week": [{"day": 1}]}]` |
| lockgroup | array | Lock group matching `[{"group": 1}]` |
| nopentime | array | No-pass time periods `[{"day": 0}]` |
| visitortime | array | Visitor time periods `[{"day": 0}]` |

**Request Example:**

```
{
  "cmd": "setdevlock",
  "sn": "LF00000001",
  "opendelay": 5,
  "doorsensor": 0,
  "alarmdelay": 30,
  "InputAlarm": 0,
  "antpass": 0,
  "interlock": 0,
  "mutiopen": 0,
  "tryalarm": 5,
  "access_times": 0,
  "punchtimes": 0,
  "tamper": 0,
  "wgformat": 0,
  "wgoutput": 0,
  "cardformat": 0,
  "wgport": 0
}
```

**Success Response:**

```
{
  "ret": "setdevlock",
  "sn": "LF00000001",
  "result": true
}
```

---

## 9.5 getdevlock — Get Door Access Parameters

Reads the current door access control configuration. No parameters.

**Request Example:**

```
{
  "cmd": "getdevlock",
  "sn": "LF00000001"
}
```

**Success Response:**

```
{
  "ret": "getdevlock",
  "sn": "LF00000001",
  "result": true,
  "opendelay": 5,
  "doorsensor": 0,
  "alarmdelay": 30,
  "InputAlarm": 0,
  "antpass": 0,
  "interlock": 0,
  "mutiopen": 0,
  "tryalarm": 5,
  "access_times": 0,
  "punchtimes": 0,
  "tamper": 0,
  "wgformat": 0,
  "wgoutput": 0,
  "cardformat": 0,
  "wgport": 0,
  "dayzone": [],
  "weekzone": [],
  "lockgroup": [],
  "nopentime": [],
  "visitortime": []
}
```

---

## 9.6 getuserlock — Get User Door Access Parameters

**Request Parameters:**

| Parameter | Type | Description |
|---|---|---|
| enrollid | number/string | User ID |
| aliasid | string | User alias ID |

**Request Example:**

```
{
  "cmd": "getuserlock",
  "sn": "LF00000001",
  "enrollid": 123,
  "aliasid": "USR001"
}
```

**Success Response:**

```
{
  "ret": "getuserlock",
  "sn": "LF00000001",
  "result": true,
  "enrollid": 123,
```

```
  "birthday": "01-15",
  "verifymode": 0,
  "enable": 1,
  "weekzone": 0,
  "group": 0,
  "access_times": 0,
  "starttime": "2025-01-01 00: 00: 00",
  "endtime": "2025-12-31 23: 59: 59",
  "aliasid": "USR001"
}
```

**Failure Response:**

```
{
  "ret": "getuserlock",
  "enrollid": 123,
  "sn": "LF00000001",
  "result": false,
  "reason": 1,
  "msg": "can not find the user"
}
```

---

## 9.7 setuserlock — Batch Set User Door Access Parameters

**Request Parameters:**

| Parameter | Type | Description |
|---|---|---|
| count | number | Number of records |
| birthday | string | Birthday (applied globally) |
| record | array | Array of `{enrollid, aliasid, verifymode, weekzone, group, access_times, starttime, endtime}` |

**Request Example:**

```
{
  "cmd": "setuserlock",
  "sn": "LF00000001",
  "count": 2,
  "birthday": "06-15",
  "record": [
    {"enrollid": 100, "aliasid": "U100", "verifymode": 0, "weekzone": 0, "group": 0, "access_times": 0,
"starttime": "2025-01-01 00: 00: 00", "endtime": "2025-12-31 23: 59: 59"},
    {"enrollid": 101, "aliasid": "U101", "verifymode": 0, "weekzone": 1, "group": 1, "access_times": 10,
"starttime": "2025-01-01 00: 00: 00", "endtime": "2025-12-31 23: 59: 59"}
  ]
}
```

**Success Response:**

```
{
  "ret": "setuserlock",
  "sn": "LF00000001",
```

```
  "result": true
}
```

**Failure Response:**

```
{
  "ret": "setuserlock",
  "sn": "LF00000001",
  "result": false,
  "reason": 1,
  "msg": "setuserlock fail"
}
```

---

## 9.8 deleteuserlock — Delete User Door Access Parameters

**Request Parameters:**

| Parameter | Type | Description |
|---|---|---|
| enrollid | number/string | User ID |
| aliasid | string | User alias ID |

**Request Example:**

```
{
  "cmd": "deleteuserlock",
  "sn": "LF00000001",
  "enrollid": 123,
  "aliasid": "USR001"
}
```

**Success Response:**

```
{
  "ret": "deleteuserlock",
  "enrollid": 123,
  "sn": "LF00000001",
  "result": true,
  "aliasid": "USR001"
}
```

---

## 9.9 cleanuserlock — Clear All User Door Access Parameters

Removes door access time zones and restrictions for all users. No parameters.

**Request Example:**

```
{
  "cmd": "cleanuserlock",
  "sn": "LF00000001"
}
```

**Success Response:**

```
{
  "ret": "cleanuserlock",
  "sn": "LF00000001",
  "result": true
}
```

# 10. Attendance Management Commands

## 10.1 getshift — Get Shift Schedule

Reads the configured shift/roster schedule. No parameters.

**Request Example:**

```
{
  "cmd": "getshift",
  "sn": "LF00000001"
}
```

**Success Response:**

```
{
  "ret": "getshift",
  "sn": "LF00000001",
  "result": true,
  "shifts": [
    {
      "name": "Day Shift",
      "cutoftime": "09: 00",
      "sections": [
        {"start": "08: 00", "end": "12: 00", "type": 0},
        {"start": "13: 00", "end": "17: 00", "type": 1},
        {"start": "17: 00", "end": "18: 00", "type": 2}
      ]
    }
  ]
}
```

> Maximum 8 shifts, each with up to 3 time sections.
> Section type: `0` =work, `1` =rest, `2` =overtime

## 10.2 setshift — Set Shift Schedule

**Request Parameters:**

| Parameter | Type | Description |
|---|---|---|
| shifts | array | Array of shift objects, each containing `name`, `cutoftime`, `sections[]` |

**Request Example:**

```
{
  "cmd": "setshift",
  "sn": "LF00000001",
  "shifts": [
    {
      "name": "Day Shift",
      "cutoftime": "09: 00",
      "sections": [
        {"start": "08: 00", "end": "12: 00", "type": 0},
        {"start": "13: 00", "end": "17: 00", "type": 1}
      ]
    },
    {
      "name": "Night Shift",
      "cutoftime": "21: 00",
      "sections": [
        {"start": "20: 00", "end": "23: 59", "type": 0},
        {"start": "00: 00", "end": "06: 00", "type": 0}
      ]
    }
  ]
}
```

**Success Response:**

```
{
  "ret": "setshift",
  "sn": "LF00000001",
  "result": true
}
```

---

## 10.3 getbelltime — Get Bell Schedule

Reads configured bell/doorbell time schedules. No parameters.

**Request Example:**

```
{
  "cmd": "getbelltime",
  "sn": "LF00000001"
}
```

**Success Response:**

```
{
  "ret": "getbelltime",
  "sn": "LF00000001",
  "result": true,
  "belltimes": [
    {"time": "08: 00", "flag": 127},
    {"time": "12: 00", "flag": 127}
  ]
}
```

> `flag` is a bitmask where each bit represents a day of the week (127 = all days)

---

## 10.4 setbelltime — Set Bell Schedule

### Request Parameters:

| Parameter | Type | Description |
| --- | --- | --- |
| belltimes | array | Array of `{time: "HH:MM", flag: number}` objects |

### Request Example:

```
{
  "cmd": "setbelltime",
  "sn": "LF00000001",
  "belltimes": [
    {"time": "08: 00", "flag": 127},
    {"time": "12: 00", "flag": 127},
    {"time": "18: 00", "flag": 65}
  ]
}
```

### Success Response:

```
{
  "ret": "setbelltime",
  "sn": "LF00000001",
  "result": true
}
```

---

## 10.5 setholiday — Set Holiday Schedule

### Request Parameters:

| Parameter | Type | Description |
| --- | --- | --- |
| holidays | array | Array of holiday objects |

### Holiday Object Fields:

| Field | Type | Description |
| --- | --- | --- |
| name | string | Holiday name |
| shift | number | Assigned shift ID |
| dayzone | number | Time zone ID |
| startday | string | Start date "MM-DD" |
| endday | string | End date "MM-DD" |

### Request Example:

```
{
  "cmd": "setholiday",
  "sn": "LF00000001",
  "holidays": [
    {"name": "Spring Festival", "startday": "01-28", "endday": "02-03", "shift": 0, "dayzone": 0},
    {"name": "National Day", "startday": "10-01", "endday": "10-07", "shift": 1, "dayzone": 0}
```

```
  ]
}
```

**Success Response:**

```
{
  "ret": "setholiday",
  "sn": "LF00000001",
  "result": true
}
```

---

### 10.6 getholiday — Get Holiday Schedule

Reads configured holiday schedule. No parameters.

**Request Example:**

```
{
  "cmd": "getholiday",
  "sn": "LF00000001"
}
```

**Success Response:**

```
{
  "ret": "getholiday",
  "sn": "LF00000001",
  "result": true,
  "holidays": [
    {"name": "Spring Festival", "startday": "01-28", "endday": "02-03", "shift": 0, "dayzone": 0},
    {"name": "National Day", "startday": "10-01", "endday": "10-07", "shift": 1, "dayzone": 0}
  ]
}
```

---

## 11. Device Active Report Commands

These commands are initiated by the device and sent to the server. The server processes them and sends back a response.

### 11.1 sendlog — Device Reports Attendance Logs

Device proactively sends attendance/access log records to the server.

**Device → Server (Request):**

```
{
  "cmd": "sendlog",
  "sn": "LF00000001",
  "count": 10,
  "logindex": 0,
  "record": [
    {
      "enrollid": 123,
```

```
      "name": "John Doe",
      "time": "2025-01-01 08: 00: 00",
      "mode": 1,
      "inout": 0,
      "event": 0,
      "aliasid": "USR001",
      "image": "base64...",
      "note": ""
    }
  ]
}
```

**Server → Device (Response):**

```
{
  "ret": "sendlog",
  "result": true,
  "count": 10,
  "logindex": 0,
  "mark": true,
  "access": 1,
  "message": "Welcome",
  "fontsize": 24,
  "text_color": [255, 0, 0],
  "voice": "tts_url",
  "voiceindex": 0,
  "name": "John Doe",
  "questionnaire": true
}
```

**Server Response Fields:**

| Field | Type | Description |
|---|---|---|
| result | boolean | Whether log was accepted |
| count | number | Number of log records received |
| logindex | number | Log index for acknowledgment |
| mark | boolean | Whether to mark logs as read |
| access | number | Access decision: `1` = grant, `0` = deny |
| message | string | Display message on device |
| fontsize | number | Message font size |
| text_color | array | RGB color [R, G, B] |
| voice | string | TTS voice URL to play |
| voiceindex | number | Voice index number |
| name | string | User name for display |
| questionnaire | boolean | Whether to show questionnaire |

---

## 11.2 senduser — Device Reports User Data Changes

Device proactively sends user enrollment/modification data when biometrics or credentials are enrolled on the device.

**Device → Server (Request) — Password Change (backupnum=10):**

```
{
  "cmd": "senduser",
  "sn": "LF00000001",
  "enrollid": 123,
  "name": "John Doe",
  "backupnum": 10,
  "admin": 0,
  "record": 12345,
  "aliasid": "USR001",
  "card": 0,
  "pwd": 12345,
  "fpflag": 0,
  "palmflag": 0,
  "faceflag": 1,
  "enable": 1,
  "birthday": "01-15",
  "shiftid": 0,
  "verifymode": 0,
  "zoneid": 0,
  "groupid": 0,
  "access_times": 0,
  "postid": 0,
  "starttime": "2025-01-01 00: 00: 00",
  "endtime": "2025-12-31 23: 59: 59",
  "profile": "{}"
}
```


**Device → Server (Request) — Card Change (backupnum=11):**


```
{
  "cmd": "senduser",
  "sn": "LF00000001",
  "enrollid": 123,
  "name": "John Doe",
  "backupnum": 11,
  "admin": 0,
  "record": 1234567890,
  "aliasid": "USR001"
}
```


**Device → Server (Request) — Face Photo (backupnum=50):**


```
{
  "cmd": "senduser",
  "sn": "LF00000001",
  "enrollid": 123,
  "name": "John Doe",
  "backupnum": 50,
  "admin": 0,
  "record": "base64_jpg_data",
  "aliasid": "USR001"
}
```


**Device → Server (Request) — Face Feature (backupnum=51):**


```
{
  "cmd": "senduser",
  "sn": "LF00000001",
  "enrollid": 123,
  "name": "John Doe",
  "backupnum": 51,
  "admin": 0,
```

```
  "record": "base64_feat_data",
  "aliasid": "USR001"
}
```

**Device → Server (Request) — Fingerprint (backupnum=0-9):**

```
{
  "cmd": "senduser",
  "sn": "LF00000001",
  "enrollid": 123,
  "name": "John Doe",
  "backupnum": 0,
  "admin": 0,
  "record": "hex_fp_data_with_compression",
  "aliasid": "USR001"
}
```

**Device → Server (Request) — Palm Vein (backupnum=40/41):**

```
{
  "cmd": "senduser",
  "sn": "LF00000001",
  "enrollid": 123,
  "name": "John Doe",
  "backupnum": 40,
  "admin": 0,
  "record": "base64_palm_data",
  "aliasid": "USR001"
}
```

**Server → Device (Response):**

```
{
  "ret": "senduser",
  "result": true,
  "enrollid": 123,
  "backupnum": 10
}
```

---

## 11.3 sendpin — Device Reports PIN Input

Device reports when a user enters a PIN code on the device keypad.

**Device → Server (Request):**

```
{
  "cmd": "sendpin",
  "sn": "LF00000001",
  "pin": "123456",
  "time": "2025-01-01 08: 00: 00"
}
```

**Server → Device (Response):**

```
{
  "ret": "sendpin",
  "result": true,
  "access": 1,
```

```
  "message": "Welcome",
  "fontsize": 24,
  "text_color": [255, 0, 0],
  "voice": "tts_url",
  "voiceindex": 0,
  "aliasid": "USR001",
  "name": "John Doe"
}
```

**Failure Response:**

```
{
  "ret": "sendpin",
  "result": false
}
```

---

## 11.4 sendqrcode — Device Reports QR Code Scan

Device reports when a user scans a QR code.

**Device → Server (Request):**

```
{
  "cmd": "sendqrcode",
  "sn": "LF00000001",
  "record": "qrcode_data_string"
}
```

**Server → Device (Response):**

```
{
  "ret": "sendqrcode",
  "result": true,
  "access": 1,
  "enrollid": 123,
  "username": "John Doe",
  "message": "Welcome",
  "fontsize": 24,
  "text_color": [255, 0, 0],
  "voice": "tts_url",
  "voiceindex": 0
}
```

**Failure Response:**

```
{
  "ret": "sendqrcode",
  "result": false
}
```

---

## 11.5 sendgps — Device Reports GPS Location [Requires USE_USB4G]

Device proactively sends GPS location data when a USB 4G module is available.

**Device → Server (Request):**

```
{
  "cmd": "sendgps",
  "sn": "LF00000001",
  "satellites": 8,
  "location": "latitude,longitude",
  "timeStamp": 1704067200
}
```

**Request Fields:**

| Field | Type | Description |
|---|---|---|
| satellites | number | Number of GPS satellites in view |
| location | string | GPS location information string |
| timeStamp | number | Unix timestamp in seconds |

> Note: The server processes this data but does not send a specific response back to the device.

---

## 11.6 otacheck — OTA Update Check

Device checks if a firmware update is available.

**Device → Server (Request):**

```
{
  "cmd": "otacheck",
  "sn": "LF00000001",
  "fwname": "ai230_device",
  "version": "1.0.0"
}
```

**Server → Device (Response) — Update Available:**

```
{
  "ret": "otacheck",
  "result": true,
  "filesize": 1048576,
  "filename": "update.pkg",
  "md5": "abc123def456..."
}
```

**Server → Device (Response) — No Update:**

```
{
  "ret": "otacheck",
  "result": false
}
```

---

## 11.7 otaget — OTA File Download

Device requests a chunk of the OTA firmware file.

**Device → Server (Request):**

```
{
  "cmd": "otaget",
  "sn": "LF00000001",
  "filename": "update.pkg",
  "index": 0,
  "count": 102400
}
```

| Parameter | Type | Description |
|---|---|---|
| filename | string | Firmware filename |
| index | number | Byte offset to start reading |
| count | number | Number of bytes to read |

**Server → Device (Response):**

```
{
  "ret": "otaget",
  "result": true,
  "index": 0,
  "count": 102400,
  "record": "base64_encoded_firmware_chunk"
}
```

**Failure Response:**

```
{
  "ret": "otaget",
  "result": false
}
```

---

# 12. File Management Commands

## 12.1 getdir — List Directory Files

Lists files in a specified directory.

**Request Parameters:**

| Parameter | Type | Description |
|---|---|---|
| path | string | Directory path (must start with `/data/`) |

**Request Example:**

```
{
  "cmd": "getdir",
  "sn": "LF00000001",
  "path": "/data/photos"
}
```

**Success Response:**

```
{
  "ret": "getdir",
  "sn": "LF00000001",
  "result": true,
  "files": ["file1.jpg", "file2.txt", "photo3.jpg"]
}
```

**Failure Response:**

```
{
  "ret": "getdir",
  "sn": "LF00000001",
  "result": false
}
```

---

## 12.2 getfile — Read File

Reads the contents of a file and returns it as base64.

**Request Parameters:**

| Parameter | Type | Description |
|---|---|---|
| path | string | Full file path |

**Request Example:**

```
{
  "cmd": "getfile",
  "sn": "LF00000001",
  "path": "/data/photos/user123.jpg"
}
```

**Success Response:**

```
{
  "ret": "getfile",
  "sn": "LF00000001",
  "result": true,
  "base64": "base64_encoded_file_content"
}
```

**Failure Response:**

```
{
  "ret": "getfile",
  "sn": "LF00000001",
  "result": false
}
```

---

### 12.3 writefile — Write File

Writes base64-encoded content to a file.

**Request Parameters:**

| Parameter | Type | Description |
|---|---|---|
| path | string | Destination file path |
| base64 | string | File content in base64 encoding |

**Request Example:**

```
{
  "cmd": "writefile",
  "sn": "LF00000001",
  "path": "/data/config/settings.json",
  "base64": "base64_encoded_json_data"
}
```

**Success Response:**

```
{
  "ret": "writefile",
  "sn": "LF00000001",
  "result": true
}
```

**Failure Response:**

```
{
  "ret": "writefile",
  "sn": "LF00000001",
  "result": false
}
```

---

## 13. Other Commands

### 13.1 upgrade — Firmware Upgrade via Server

Uploads firmware upgrade data to the device in chunks.

**Request Parameters:**

| Parameter | Type | Description |
|---|---|---|
| filename | string | Target filename ( `.zip` or `.pkg` ) |
| md5 | string | File MD5 checksum |
| allcount | number | Total file size in bytes |
| index | number | Current byte offset |
| count | number | Size of this data chunk in bytes |

| Parameter | Type | Description |
| --- | --- | --- |
| record | string | Base64-encoded file data chunk |

**Request Example (first chunk):**

```
{
  "cmd": "upgrade",
  "sn": "LF00000001",
  "filename": "update.pkg",
  "md5": "abc123def456...",
  "allcount": 10485760,
  "index": 0,
  "count": 102400,
  "record": "base64_encoded_chunk_data"
}
```

**Success Response:**

```
{
  "ret": "upgrade",
  "sn": "LF00000001",
  "result": true
}
```

**Failure Responses:**

| reason | Message |
| --- | --- |
| 1 | is not this machine upgrade file / disk space insufficient |
| 2 | md5 check error |
| 3 | move the zip file error |
| 4 | need zip or pkg |

```
{
  "ret": "upgrade",
  "sn": "LF00000001",
  "result": false,
  "reason": 1,
  "msg": "is not this machine upgrade file"
}
```

---

## 13.2 forceota — Force OTA Update

Triggers an immediate OTA update check and download. No parameters.

**Request Example:**

```
{
  "cmd": "forceota",
  "sn": "LF00000001"
}
```

**Success Response:**

```
{
  "ret": "forceota",
  "sn": "LF00000001",
  "result": true
}
```

---

## 13.3 setotaserver — Set OTA Server Address

**Request Parameters:**

| Parameter | Type | Description |
|---|---|---|
| address | string | OTA server address (URL or IP) |

**Request Example:**

```
{
  "cmd": "setotaserver",
  "sn": "LF00000001",
  "address": "https://ota.example.com"
}
```

**Success Response:**

```
{
  "ret": "setotaserver",
  "sn": "LF00000001",
  "result": true
}
```

**Failure Response:**

```
{
  "ret": "setotaserver",
  "sn": "LF00000001",
  "result": false
}
```

---

## 13.4 getotaserver — Get OTA Server Address

Reads the configured OTA server address. No parameters.

**Request Example:**

```
{
  "cmd": "getotaserver",
  "sn": "LF00000001"
}
```

**Success Response:**

```
{
  "ret": "getotaserver",
  "sn": "LF00000001",
```

```
  "result": true,
  "address": "https://ota.example.com"
}
```

## 13.5 setscreensaver — Set Screensaver

Configures the device screensaver with an image.

**Request Parameters:**

| Parameter | Type | Description |
|---|---|---|
| filename | string | Screensaver filename |
| photo | string | Image data as base64 |
| duration | number | Display duration in seconds |
| remove | boolean | `true` = delete the specified file |
| removeall | boolean | `true` = delete all screensavers |

**Request Example:**

```
{
  "cmd": "setscreensaver",
  "sn": "LF00000001",
  "filename": "screensaver1.jpg",
  "photo": "base64_encoded_image",
  "duration": 30,
  "remove": false,
  "removeall": false
}
```

**Success Response:**

```
{
  "ret": "setscreensaver",
  "sn": "LF00000001",
  "result": true
}
```

## 13.6 setvoice — Set Custom Voice

Uploads and optionally plays a custom voice file.

**Request Parameters:**

| Parameter | Type | Description |
|---|---|---|
| index | number | Voice file index number |
| voice | string | WAV file data as base64 |
| play | boolean | `true` = play immediately |
| repeat_seconds | number | Repeat interval in seconds |

**Request Example:**

```
{
  "cmd": "setvoice",
  "sn": "LF00000001",
  "index": 10,
  "voice": "base64_encoded_wav_data",
  "play": true,
  "repeat_seconds": 60
}
```

**Success Response:**

```
{
  "ret": "setvoice",
  "sn": "LF00000001",
  "result": true
}
```

---

## 13.7 keypad — Control On-Screen Keypad

Activates or deactivates the on-screen numeric keypad.

**Request Parameters:**

| Parameter | Type | Description |
|---|---|---|
| password | boolean | Whether password is required |
| keep | boolean | `true` = persist, `false` = temporary |
| timeout | number | Timeout in seconds |
| on | boolean | `true` = activate, `false` = deactivate |

**Request Example:**

```
{
  "cmd": "keypad",
  "sn": "LF00000001",
  "password": true,
  "keep": false,
  "timeout": 30,
  "on": true
}
```

**Success Response:**

```
{
  "ret": "keypad",
  "sn": "LF00000001",
  "result": true
}
```

---

## 13.8 verify — Biometric Verification

Performs face or fingerprint verification on the server side.

**Request Parameters:**

| Parameter | Type | Description |
|---|---|---|
| enrollid | number/string | User ID to verify against |
| card | number/string | RFID card number |
| pwd | number | Password |
| name | string | User name |
| face | string | Face photo as base64 |
| finger | string | Fingerprint data as hex (with compression) |
| finger2 | string | Second fingerprint as hex |
| profile | string | User profile JSON |

**Request Example:**

```
{
  "cmd": "verify",
  "sn": "LF00000001",
  "enrollid": 123,
  "card": 12345,
  "pwd": 0,
  "name": "John Doe",
  "face": "base64_face_photo",
  "finger": "",
  "finger2": "",
  "profile": "{}"
}
```

**Success Response:**

```
{
  "ret": "verify",
  "enrollid": 123,
  "aliasid": "USR001",
  "sn": "LF00000001",
  "result": true
}
```

**Failure Responses (reason codes):**

| reason | Message |
|---|---|
| 1 | base64 decode error |
| 2 | File too large (>512KB) |
| 3 | File size is 0 |
| 5 | No face detected in image |
| 6 | More than 1 face in image |
| 7 | Image dimensions too large |

| reason | Message |
|---|---|
| 8 | Face too large |
| 9 | Face too small |
| 10 | Image quality too low |
| 11 | Face position offset too large |
| 12 | No valid data (no face and no fingerprint) |

```
{
  "ret": "verify",
  "enrollid": 123,
  "aliasid": "USR001",
  "sn": "LF00000001",
  "result": false,
  "reason": 5,
  "msg": "no face in picture"
}
```

---

## 13.9 setdepartment — Set Department List

### Request Parameters:

| Parameter | Type | Description |
|---|---|---|
| delete | boolean | `true` = clear all departments |
| name | array | Array of department names |

### Request Example (set departments):

```
{
  "cmd": "setdepartment",
  "sn": "LF00000001",
  "delete": false,
  "name": ["Engineering", "Sales", "HR", "Finance", "Operations"]
}
```

### Request Example (clear all):

```
{
  "cmd": "setdepartment",
  "sn": "LF00000001",
  "delete": true
}
```

### Success Response:

```
{
  "ret": "setdepartment",
  "sn": "LF00000001",
  "result": true
}
```

---

## 13.10 getdepartment — Get Department List

Reads the configured department list. No parameters.

**Request Example:**

```
{
  "cmd": "getdepartment",
  "sn": "LF00000001"
}
```

**Success Response:**

```
{
  "ret": "getdepartment",
  "sn": "LF00000001",
  "result": true,
  "count": 5,
  "name": ["Engineering", "Sales", "HR", "Finance", "Operations"]
}
```

**Failure Response:**

```
{
  "ret": "getdepartment",
  "sn": "LF00000001",
  "result": false,
  "reason": 1,
  "msg": "can not find department file"
}
```

---

## 13.11 setcompanyname — Set Company Name

**Request Parameters:**

| Parameter | Type | Description |
|---|---|---|
| name | string | Company name |

**Request Example:**

```
{
  "cmd": "setcompanyname",
  "sn": "LF00000001",
  "name": "Acme Corporation"
}
```

**Success Response:**

```
{
  "ret": "setcompanyname",
  "sn": "LF00000001",
  "result": true
}
```

---

## 13.12 getcompanyname — Get Company Name

Reads the configured company name. No parameters.

**Request Example:**

```
{
  "cmd": "getcompanyname",
  "sn": "LF00000001"
}
```

**Success Response:**

```
{
  "ret": "getcompanyname",
  "sn": "LF00000001",
  "result": true,
  "name": "Acme Corporation"
}
```

---

## 13.13 setquestionnaire — Set Questionnaire

Configures a questionnaire/survey that is displayed to users after verification.

**Request Parameters:**

| Parameter | Type | Description |
|---|---|---|
| title | string | Questionnaire title |
| voice | string | Voice prompt filename |
| errmsg | string | Error message text |
| radio | boolean | Single-choice mode |
| optionflag | number | Option flags |
| items | array | Array of option strings (max 16) |
| schedules | array | Display schedules `["HH:MM-HH:MM*N"]` |
| usequestion | boolean | Enable questionnaire |
| useschedule | boolean | Enable schedule |
| card | number/string | Associated card number |
| switch_card | number/string | Switching card number |
| enrollid | number/string | Associated user ID |
| open | boolean | Open immediately |

**Request Example:**

```
{
  "cmd": "setquestionnaire",
  "sn": "LF00000001",
  "title": "Health Declaration",
  "voice": "health.wav",
  "errmsg": "Please complete the survey",
```

```
  "radio": true,
  "optionflag": 0,
  "items": ["Option A", "Option B", "Option C"],
  "schedules": ["08: 00-18: 00*1"],
  "usequestion": true,
  "useschedule": false,
  "card": 0,
  "switch_card": 0,
  "enrollid": 0,
  "open": false
}
```

**Success Response:**

```
{
  "ret": "setquestionnaire",
  "sn": "LF00000001",
  "result": true
}
```

---

## 13.14 getquestionnaire — Get Questionnaire

Reads the current questionnaire configuration. No parameters.

**Request Example:**

```
{
  "cmd": "getquestionnaire",
  "sn": "LF00000001"
}
```

**Success Response:**

```
{
  "ret": "getquestionnaire",
  "sn": "LF00000001",
  "result": true,
  "title": "Health Declaration",
  "voice": "health.wav",
  "errmsg": "Please complete the survey",
  "radio": true,
  "optionflag": 0,
  "usequestion": true,
  "useschedule": false,
  "card": 0,
  "switch_card": 0,
  "items": ["Option A", "Option B", "Option C"],
  "schedules": ["08: 00-18: 00*1"]
}
```

**Not Configured Response:**

```
{
  "ret": "getquestionnaire",
  "sn": "LF00000001",
  "result": false
}
```

---

### 13.15 cleanquestionnaire — Clear Questionnaire

Removes the questionnaire configuration. No parameters.

**Request Example:**

```
{
  "cmd": "cleanquestionnaire",
  "sn": "LF00000001"
}
```

**Success Response:**

```
{
  "ret": "cleanquestionnaire",
  "sn": "LF00000001",
  "result": true
}
```

---

## 14. Video Intercom Commands

> **Note:** These commands require the `VIDEO_INTERCOM` feature to be enabled on the device.

### 14.1 enablewebrtc — Enable WebRTC

Configures and enables WebRTC video intercom capabilities.

**Request Parameters:**

| Parameter | Type | Description |
|---|---|---|
| intercom | boolean | Enable voice/video intercom |
| monitor | boolean | Enable monitoring |
| waitseconds | number | Wait timeout in seconds |
| 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 / boolean | Audio-only intercom: `1` /true = audio only, `0` /false = audio + video (also controls video on/off) |
| stunserver | string | STUN server URL |
| turnserver | string | TURN server URL |
| turnusername | string | TURN authentication username |
| turnpassword | string | TURN authentication password |

**Request Example:**

```
{
  "cmd": "enablewebrtc",
  "sn": "LF00000001",
  "intercom": true,
  "monitor": true,
  "waitseconds": 30,
  "inputmode": 0,
  "stunserver": "stun:stun.example.com: 3478",
  "turnserver": "turn:turn.example.com: 3478",
  "turnusername": "user",
  "turnpassword": "password"
}
```

**Success Response:**

```
{
  "ret": "enablewebrtc",
  "sn": "LF00000001",
  "result": true
}
```

**Failure Response:**

```
{
  "ret": "enablewebrtc",
  "sn": "LF00000001",
  "result": false,
  "reason": 1,
  "msg": "this device have not this function"
}
```

---

## 14.2 callaccept — Accept or Reject Call

Server-side response to an incoming intercom call from the device.

**Request Parameters:**

| Parameter | Type | Description |
|---|---|---|
| result | boolean | `true` = accept, `false` = reject |
| sessionid | string | Call session ID |

**Request Example (accept):**

```
{
  "cmd": "callaccept",
  "sn": "LF00000001",
  "result": true,
  "sessionid": "session_abc123"
}
```

**Device Response — Sends WebRTC Offer:**

```
{
  "cmd": "offer",
  "sn": "LF00000001",
  "sessionid": "session_abc123",
```

```
  "type": "offer",
  "sdp": "v=0\r\no=..."
}
```

---

## 14.3 viewready — Monitor Ready

Server indicates it is ready to receive video monitoring stream.

**Request Parameters:**

| Parameter | Type | Description |
|---|---|---|
| sessionid | string | Session ID |

**Request Example:**

```
{
  "cmd": "viewready",
  "sn": "LF00000001",
  "sessionid": "session_abc123"
}
```

**Device Response — Sends WebRTC Offer:**

```
{
  "cmd": "offer",
  "sn": "LF00000001",
  "sessionid": "session_abc123",
  "type": "offer",
  "sdp": "v=0\r\no=..."
}
```

---

## 14.4 answer — Process WebRTC Answer

Server sends the WebRTC answer SDP to the device.

**Request Parameters:**

| Parameter | Type | Description |
|---|---|---|
| sessionid | string | Session ID |
| sdp | string | SDP answer content |

**Request Example:**

```
{
  "cmd": "answer",
  "sn": "LF00000001",
  "sessionid": "session_abc123",
  "sdp": "v=0\r\no=..."
}
```

**Behavior:** No direct response. Device processes the answer internally and may retry sending offer if needed.

---

## 14.5 icecandidate — Process ICE Candidate

Server sends an ICE candidate to the device for NAT traversal.

**Request Parameters:**

| Parameter | Type | Description |
|---|---|---|
| sessionid | string | Session ID |
| candidate | string | ICE candidate content |

**Request Example:**

```
{
  "cmd": "icecandidate",
  "sn": "LF00000001",
  "sessionid": "session_abc123",
  "candidate": "candidate: 1 1 UDP 123456 192.168.1.100 12345 typ host"
}
```

**Behavior:** No direct response. Device adds the ICE candidate internally.

---

## 14.6 callcancel — Cancel Call

Cancels an active or pending intercom call.

**Request Parameters:**

| Parameter | Type | Description |
|---|---|---|
| sessionid | string | Session ID |
| type | number | `1` = intercom, `2` = monitoring |

**Request Example:**

```
{
  "cmd": "callcancel",
  "sn": "LF00000001",
  "sessionid": "session_abc123",
  "type": 1
}
```

**Behavior:** No direct response.

---

## 14.7 talklock — Door Control During Call

Controls the door lock during an active intercom call.

**Request Parameters:**

| Parameter | Type | Description |
|---|---|---|
| type | number | `1` = open door, `2` = trigger alarm |

**Request Example:**

```
{
  "cmd": "talklock",
  "sn": "LF00000001",
  "type": 1
}
```

**Success Response:**

```
{
  "ret": "talklock",
  "sn": "LF00000001",
  "result": true
}
```

---

## 14.8 monitorcall — Start Monitoring

Server initiates a monitoring session to view the device's video feed.

**Request Parameters:**

| Parameter | Type | Description |
|---|---|---|
| force | boolean | `true` = force disconnect any existing monitoring session |
| sessionid | string | Session ID |

**Request Example:**

```
{
  "cmd": "monitorcall",
  "sn": "LF00000001",
  "force": false,
  "sessionid": "session_abc123"
}
```

**Success Response:**

```
{
  "ret": "monitorcall",
  "sn": "LF00000001",
  "lanscape": 1,
  "result": true,
  "sessionid": "session_abc123"
}
```

**Busy Response:**

```
{
  "ret": "monitorcall",
  "sn": "LF00000001",
  "lanscape": 1,
  "result": false,
  "reason": 1,
  "msg": "another user have monitored",
  "sessionid": "session_abc123",
```

```
  "monitoring": "other_session_id"
}
```

---

## 14.9 setviselection — Set Visual Selection Menu

Configures the interactive visual selection menu displayed on the device's touchscreen.

> **Each page (building) is either/or**: send only `items[]` (flat mode) **or** only `floors[]` (floor mode). Do **not** send both in the same page.

**Request Parameters:**

| Parameter | Type | Description |
|---|---|---|
| title | string | Menu title (top title bar) |
| voice | string | Voice prompt content (TTS text, not a filename) |
| pages | array | Page array (max 6 pages, one page = one building) |
| pages[].itemtitle | string | Tab title, e.g. "Building 1" |
| **Flat mode (either/or)** | | |
| pages[].items | array | Flat room list (max 500), mutually exclusive with `floors` |
| items[].section | string | Button label, e.g. "101" |
| items[].keynum | number | Key number (call target) |
| **Floor mode (either/or)** | | |
| pages[].floors | array | Floor array (max 100), mutually exclusive with `items` |
| floors[].floortitle | string | Floor label, e.g. "Floor 1" |
| floors[].items | array | Room list of this floor (max 50 per floor) |
| floors[].items[].section | string | Room label, e.g. "101" |
| floors[].items[].keynum | number | Room key number (call target) |

**Request Example (Mode 1: Flat / legacy):**

```
{
  "cmd": "setviselection",
  "sn": "LF00000001",
  "title": "Visitor Menu",
  "voice": "Please select a room",
  "pages": [
    {
      "itemtitle": "Building 1",
      "items": [
        {"section": "101", "keynum": 101},
        {"section": "102", "keynum": 102}
      ]
    }
```

```
  ]
}
```

**Request Example (Mode 2: Floors / new):**

```
{
  "cmd": "setviselection",
  "sn": "LF00000001",
  "title": "Visitor Menu",
  "voice": "Please select a floor",
  "pages": [
    {
      "itemtitle": "Building 1",
      "floors": [
        {
          "floortitle": "Floor 1",
          "items": [
            {"section": "101", "keynum": 101},
            {"section": "102", "keynum": 102}
          ]
        },
        {
          "floortitle": "Floor 2",
          "items": [
            {"section": "201", "keynum": 201}
          ]
        }
      ]
    }
  ]
}
```

**Success Response:**

```
{
  "ret": "setviselection",
  "sn": "LF00000001",
  "result": true
}
```

---

## 14.10 talkcall — Device Initiates Call

Device proactively initiates an intercom call to the server.

**Device → Server (Request):**

```
{
  "cmd": "talkcall",
  "sn": "LF00000001",
  "inputnum": 0,
  "inputmode": 0,
  "landscape": 1
}
```

**Server → Device (Response):**

```
{
  "ret": "talkcall",
  "result": true,
```

```
  "sessionid": "session_abc123"
}
```

**Failure Response (reason codes):**

| reason | Meaning |
|---|---|
| 1 | Subscriber list is empty |
| 2 | Subscriber is offline |
| 3 | No permission |
| 4 | Call rejected by subscriber |
| 5 | Subscriber is busy |
| 6 | Subscriber state is incorrect |

```
{
  "ret": "talkcall",
  "result": false,
  "reason": 1
}
```

---

## 14.11 vicheckonline — Check Online Status

Device checks if subscribers are online.

**Device → Server (Request):**

```
{
  "cmd": "vicheckonline",
  "sn": "LF00000001"
}
```

**Server → Device (Response):**

```
{
  "ret": "vicheckonline",
  "result": true,
  "count": 1,
  "online": "key1,key2"
}
```

## 14.12 WebRTC DataChannel Extension

After a WebRTC connection is established, the device and its peer (indoor unit / outdoor unit / App) open two additional **WebRTC DataChannels** for peer-to-peer real-time control commands (they do **not** go through the server and do not use the WebSocket/MQTT channel):

| Channel | Purpose |
|---|---|
| `intercom_channel` | Intercom session (during call / talk) |
| `monitor_channel` | Monitor session (during monitoring) |

- Payload is **UTF-8 JSON text** (not binary); a single message should be < 256 bytes.
- Field style follows the WebSocket/MQTT `cmd` / `ret` convention, but messages are **peer-to-peer** and bypass the server.

**1) Peer → Device (received by the device):**

| Command | Field | Type | Required | Description |
|---|---|---|---|---|
| `opendoor` | `cmd` | string | yes | Fixed value `"opendoor"` |
| | `type` | number | yes | `1` = unlock; any other value = trigger alarm |
| | `enrollid` | number | no | Card / user ID used for Wiegand output; defaults to `99999997` |
| | `floor` | number | no | Floor number, effective only when elevator control is enabled; `monitor_channel` requires `floor > 0` |
| `talk` | `cmd` | string | yes | Fixed value `"talk"` (`monitor_channel` **only**) |
| | `enable` | number / boolean | yes | `1`/true = start talking (device plays the peer audio); `0`/false = stop |
| `msg` | `msg` | string | yes | No `cmd`; a top-level `msg` key identifies this message. Shows a prompt on the device screen for 3 seconds; max 127 bytes (buffer limit, not validated, longer input overflows) |

**Example (unlock, on `intercom_channel` or `monitor_channel`):**

```
{"cmd":"opendoor","type": 1,"enrollid": 123456,"floor": 5}
```

**Example (talk switch during monitoring, `monitor_channel` only):**

```
{"cmd":"talk","enable": 1}
```

**Example (screen message):**

```
{"msg":"Please wait"}
```

**2) Device → Peer (sent by the device):**

| Command | Field | Type | Description |
|---|---|---|---|
| `opendoor` | `ret` | string | Fixed value `"opendoor"` |
| | `result` | boolean | Whether the action was executed |

| Command | Field | Type | Description |
|---|---|---|---|
| | `type` | number | Echoes the `type` from the request |

```
{"ret":"opendoor","result": true,"type": 1}
```

> `intercom_channel` and `monitor_channel` share exactly the same message format; only the session type differs. `talk` is valid on `monitor_channel` only.

---

# 15. Appendix

## Appendix A: backupnum Value Reference

| backupnum | Data Type | Encoding | Max Capacity | Description |
|---|---|---|---|---|
| 0 | Fingerprint #1 | hex | 10000 | First fingerprint slot |
| 1 | Fingerprint #2 | hex | 10000 | Second fingerprint slot |
| 2-9 | Fingerprint #3-10 | hex | 10000 each | Additional fingerprint slots |
| 10 | Password | number | 10000 | User PIN/password |
| 11 | Card Number | number | 10000 | RFID card number |
| 12 | Full User Delete | — | — | Delete entire user (same as 13) |
| 13 | Full User Delete | — | — | Delete entire user (same as 12) |
| 40 | Palm Vein (Left) | base64 | 10000 | Left palm vein |
| 41 | Palm Vein (Right) | base64 | 10000 | Right palm vein |
| 50 | Face Photo | base64 (JPG) | 10000 | Face photograph |
| 51 | Face Feature | base64 | 10000 | Face feature template |

## Appendix B: verifymode Reference

`verifymode` selects the verification method(s) a user may use. 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 |

| verifymode | Verification method | Notes |
|---|---|---|
| 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.

## Appendix C: mode Field in Log Records

The `mode` field of a log record reports the verification method actually used and uses the same values as `verifymode`:

| mode value | Verification method |
|---|---|
| 0 | Any method |
| 1 | Fingerprint |
| 2 | Password |
| 3 | Card |
| 4 | Fingerprint + card |
| 5 | Fingerprint + password |

| mode value | Verification method |
|---|---|
| 6 | Password + fingerprint |
| 7 | Card + fingerprint |
| 8 | Face |
| 9 | Face + password |
| 10 | Face + card |
| 11 | Card + password |
| 12 | Face + fingerprint |
| 13 | QR code |
| 14 | Face + (card or password) |
| 15 | ID card (SFZ) |
| 16 | Palm vein |
| 19 | 1:N automatic match |
| 20 | Any biometric of the person |
| 21 | Palm vein + face |
| 22 | Fingerprint + palm vein |
| 23 | Card + palm vein |
| 24 | Palm vein + password |
| 25 | Card + face + password |

> The `event` field of a record reports access-control events (door open/close, duress, ...) and is a different field from `mode`.
> In exported xls reports the text of this column is derived by the device itself, so a few methods may appear under a similar name; values read through the API are always the raw values.

## Appendix D: Device Capacity Field Descriptions

| Field | Description |
|---|---|
| usersize | Maximum number of user profiles |
| facesize | Maximum number of enrolled face templates |
| avatarsize | Maximum number of avatar photos |
| fpsize | Maximum number of fingerprints |
| palmsize | Maximum palm vein templates |
| cardsize | Maximum RFID card numbers |
| pwdsize | Maximum passwords |
| logsize | Maximum log record storage |
| useduser | Number of currently enrolled users |
| usedface | Number of enrolled face templates in use |

| Field | Description |
| --- | --- |
| usedavatar | Number of avatar photos in use |
| usedfp | Number of fingerprints in use |
| usedpalm | Number of palm veins in use |
| usedcard | Number of registered cards |
| usedpwd | Number of registered passwords |
| usedlog | Total number of log records stored |
| usednewlog | Number of unread/new log records |
| usedrtlog | Number of real-time log queue entries |

### Appendix E: Unknown Command Default Response

When a command is not recognized by the device:

```
{
  "ret": "<original_command>",
  "sn": "LF00000001",
  "result": false,
  "reason": 1,
  "msg": "can not find this command"
}
```

# Command Summary Table

| Category | Commands |
| --- | --- |
| **User Management** | setuserinfo, deleteuser, cleanuser, setusername, getusername, getuserlist, getuserids, getunuserdid, checkuserid, getuserinfo, getallusers, adduser, checkregstatus, canceladduser, enableuser, getuserprofile, setuserprofile |
| **Log Management** | getnewlog, getalllog, cleanlog, cleanlogphoto |
| **Device Management** | getdevinfo, setdevinfo, getdevcap, gettime, settime, initsys, initmenu, cleandatebase, reboot, disabledevice, enabledevice, cleanadmin, cleaninactiveuser, checklive, getreg |
| **Access Control** | opendoor, lockctrl, getdoorstatus, setdevlock, getdevlock, getuserlock, setuserlock, deleteuserlock, cleanuserlock |
| **Attendance** | getshift, setshift, getbelltime, setbelltime, setholiday, getholiday |
| **Device Active Reports** | sendlog, senduser, sendpin, sendqrcode, sendgps, otacheck, otaget |
| **File Management** | getdir, getfile, writefile |
| **System / Other** | upgrade, forceota, setotaserver, getotaserver, setscreensaver, setvoice, keypad, verify, setdepartment, getdepartment, setcompanyname, getcompanyname, setquestionnaire, getquestionnaire, cleanquestionnaire |
| **Video Intercom** | enablewebrtc, callaccept, viewready, answer, icecandidate, callcancel, talklock, monitorcall, setviselection, talkcall, vicheckonline |

| Category | Commands |
| --- | --- |
| **Device Registration** | reg (Active_send_reg) |
| **Total Unique Commands** | **83** |

---

*Document Version: 1.0*
*Last updated: 2026-09-22 — corrected the verifymode value reference, added the value table and fixed the example values*
*Compatible with AiFace Firmware: 1.0.0+*

# 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": "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA..."
}
```

> `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": "CAYAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA..."
}
```

> `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*