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 operating over **HTTP/HTTPS** polling communication.

> ⚠️ **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.

---

1. Transport Protocol Specifications

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.

2.1 General Request Format

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

2.2 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>"
}
```

2.3 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
3	Operation failed
4	Invalid format
5	No data / not found
6-12	Specific validation failures

---

3. 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`

---

4. Device Registration — `reg` Command

4.1 Device → Server (Registration Request)

Also known as `Active_send_reg`. The device sends this command automatically immediately after the HTTP connection is established.

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
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

Field	Type	Description
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

4.2 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`

---

5. User Management Commands

5.1 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)
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)

Parameter	Type	Description
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"
}
```

5.2 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"
}
```

5.3 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
}
```

5.4 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"
}
```

5.5 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"
}
```

5.6 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": []
}
```

5.7 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.

5.8 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"
}
```

5.9 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
}
```

5.10 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"
}
```

5.11 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"
}
```

5.12 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
}
```

5.13 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

---

5.14 canceladduser — Cancel User Enrollment

Cancels the current user enrollment process. No parameters.

Request Example:

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

Success Response:

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

5.15 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"
}
```

5.16 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"
}
```

5.17 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"
}
```

---

6. Log Management Commands

6.1 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

6.2 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)
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": ""
    }
  ]
}
```

6.3 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
}
```

6.4 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
}
```

---

7. Device Management Commands

7.1 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.

7.2 setdevinfo — Set Device System Parameters

Sets one or more system parameters.

Request Parameters:

Parameter	Type	Description
nowtime	number	Current Unix timestamp (seconds)

Parameter	Type	Description
(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": 81
}
```

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`).

7.3 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

7.4 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"
}
```

7.5 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"
}
```

7.6 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
}
```

7.7 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
}
```

7.8 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
}
```

7.9 cleanadmin — Clear All Administrators

Removes all administrator privileges. No parameters.

Request Example:

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

Success Response:

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

7.10 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
}
```

7.11 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.

7.12 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
}
```

7.13 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
}
```

7.14 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"
  }
}
```


7.15 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
}
```

8. Access Control Commands

8.1 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)

Parameter	Type	Description
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
}
```

8.2 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
}
```

8.3 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)

8.4 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)
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"}]}]`

Parameter	Type	Description
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
}
```

8.5 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": []
}
```

8.6 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"
}
```

8.7 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"
}
```

8.8 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"
}
```

8.9 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
}
```

---

9. Attendance Management Commands

9.1 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

9.2 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
}
```

9.3 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)

9.4 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
}
```

9.5 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
}
```

9.6 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}
  ]
}
```

---

10. 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.

10.1 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

10.2 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
}
```

10.3 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
}
```

10.4 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
}
```

10.5 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.

10.6 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
}
```

10.7 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
}
```

11. File Management Commands

11.1 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
}
```

11.2 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
}
```

11.3 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
}
```

---

12. Other Commands

12.1 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"
}
```

12.2 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
}
```

12.3 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
}
```

12.4 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"
}
```

12.5 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
}
```

12.6 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
}
```

12.7 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
}
```

12.8 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"
}
```

12.9 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
}
```

12.10 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"
}
```

12.11 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
}
```

12.12 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"
}
```

12.13 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
}
```

12.14 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
}
```

12.15 cleanquestionnaire — Clear Questionnaire

Removes the questionnaire configuration. No parameters.

Request Example:

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

Success Response:

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

---

13. Video Intercom Commands

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

13.1 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"
}
```

13.2 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=..."
}
```

13.3 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=..."
}
```

13.4 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.

13.5 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.

13.6 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.

13.7 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
}
```

13.8 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"
}
```

13.9 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
}
```

13.10 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
}
```

13.11 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"
}
```

13.12 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.

14. 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+