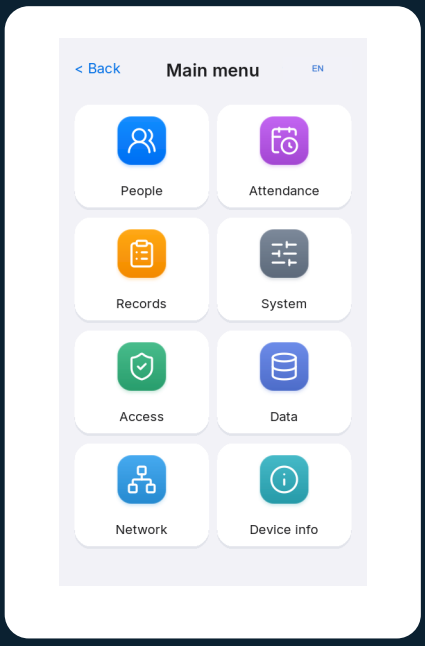
OPEN APPLICATION DEVELOPMENT

Build your application on TIMYONE A05

A Linux face-recognition terminal with editable C++ application source and resources to help your team build its own device experience.

Your interface. Your workflow.
Start from the attendance and access reference, or create a different application.



Illustrative interface from an earlier reference; screens vary by version.

One platform. Your product.

Keep the preinstalled face runtime and build the application around your own product requirements.

01

Device & runtime

Linux-based A05 terminal with touchscreen and camera interfaces; a matched face runtime and licence are installed per device.

02

Open application

Adapt the provided C++ screens, Web pages, business rules and reference workflows.

03

Your integration

Connect the reference application to customer software through the documented WSS / HTTPS interfaces.

Device configuration, supported peripherals and applicable software depend on the model supplied.

FROM REFERENCE TO PRODUCT

A clear path for your team

Start with a working reference structure, then shape the application and integration around your use case.

01

Prepare

Use Linux x86_64 or Windows with WSL. Obtain matching SDK files, cross-toolchain and application source.

02

Build & customize

Edit interface assets, C++ screens and business workflows. Build and package an application for the target.

03

Connect & verify

Follow the WSS / HTTPS references and validate the application on the matching device and system.

Open source where you need it

Adapt C++ screens, personnel, recognition and event workflows. The complete application source is provided for you to study and modify.

A05 APPLICATION PROJECT

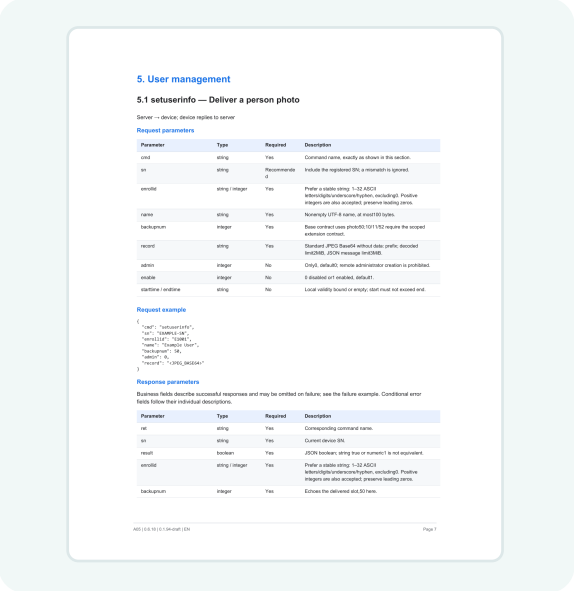
app / screens.cpp

app / workflow.cpp

firmware / web /

kit.py build | package

Reference protocol documentation includes fields, request and response examples, and integration guidance.



5. User management

5.1 setuserinfo — Deliver a person photo

Server → device; device replies to server

Request parameters

Parameter	Type	Required	Description
cmd	string	Yes	Command name, exactly as shown in this section.
sn	string	Recommende d	Include the registered SN; a mismatch is ignored.
enrollid	string / integer	Yes	Prefer a stable string: 1–32 ASCII letters/digits/underscore/hyphen, excluding0. Positive integers are also accepted; preserve leading zeros.
name	string	Yes	Nonempty UTF-8 name, at most180 bytes.
backupnum	integer	Yes	Base contract uses photo50,1011152 require the scoped extension contract.
record	string	Yes	Standard JPEG Base64 without data: prefix; decoded limit2MB, JSON message limit3MB.
admin	integer	No	Only0, default0. remote administrator creation is prohibited.
enable	integer	No	0 disabled or1 enabled, default1
starttime / endtime	string	No	Local validity bound or empty; start must not exceed end.

Request example

```
{
"cmd": "setuserinfo",
"sn": "5A4M01A-SN",
"enrollid": "120001",
"name": "Example User",
"backupnum": 50,
"admin": 0,
"record": "/JPEG_BASE64+"
}
```

Response parameters

Business fields describe successful responses and may be omitted on failure; see the failure example. Conditional error fields follow their individual descriptions.

Parameter	Type	Required	Description
ret	string	Yes	Corresponding command name.
sn	string	Yes	Current device SN.
result	boolean	Yes	JSON boolean; string true or numeric1 is not equivalent.
enrollid	string / integer	Yes	Prefer a stable string: 1–32 ASCII letters/digits/underscore/hyphen, excluding0. Positive integers are also accepted; preserve leading zeros.
backupnum	integer	Yes	Echoes the delivered slot,50 here.

A05 | 0.6.10 | 0.1.54-draft | EN Page 7

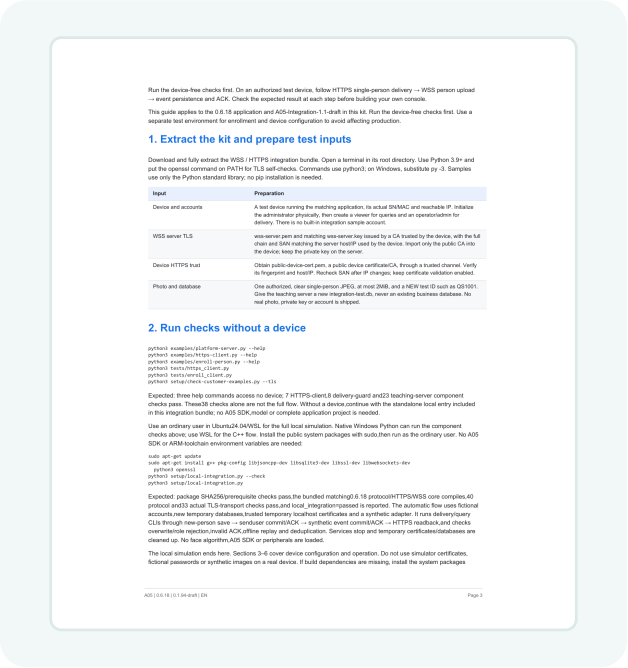
WSS / HTTPS protocol reference

The WSS / HTTPS services belong to the reference application. A custom application must retain or implement them to expose the same interfaces.

WHAT YOUR TEAM RECEIVES

Resources for independent development

Source, tutorials and practical examples provide a starting point for application development and software integration.



Run the device-free checks first. On an authorized test device, follow HTTPS single-person delivery → WSS person upload → event persistence and ACK. Check the expected result at each step before building your own console.

This guide applies to the 0.6.18 application and A05-Integration-1.1-draft in this kit. Run the device-free checks first. Use a separate test environment for enrollment and device configuration to avoid affecting production.

1. Extract the kit and prepare test inputs

Download and fully extract the WSS / HTTPS integration bundle. Open a terminal in its root directory. Use Python 3.9+ and put the openssl command on PATH for TLS self-checks. Commands use python3; on Windows, substitute py -3. Samples use only the Python standard library; no pip installation is needed.

Input	Preparation
Device and accounts	A test device running the matching application, its actual SN/MAC and reachable IP. Initialize the administrator physically, then create a viewer for queries and an operator/admin for delivery. There is no built-in integration sample account.
WSS server TLS	wss-server.pem and matching wss-server.key issued by a CA trusted by the device, with the full chain and SAN matching the server hostIP used by the device. Import only the public CA into the device; keep the private key on the server.
Device HTTPS trust	Obtain public-device-cert.pem, a public device certificate/CA, through a trusted channel. Verify its fingerprint and hostIP. Recheck SAN after IP changes; keep certificate validation enabled.
Photo and database	One authorized, clear single-person JPEG, at most 2MB, and a NEW test ID such as QS1001. Give the teaching server a new integration-test.db, never an existing business database. No real photo, private key or account is shipped.

2. Run checks without a device

```
python3 examples/platform-server.py --help
python3 examples/https-client.py --help
python3 examples/enroll-person.py --help
python3 tests/https_client.py
python3 tests/enroll_client.py
python3 setup/check-customer-examples.py --tls
```

Expected: three help commands access no device; 7 HTTPS-client,8 delivery-guard and23 teaching-server component checks pass. These38 checks alone are not the full flow. Without a device,continue with the standalone local entry included in this integration bundle; no A05 SDK,model or complete application project is needed .

Use an ordinary user in Ubuntu24.04WSL for the full local simulation. Native Windows Python can run the component checks above; use WSL for the C++ flow. Install the public system packages with sudo,then run as the ordinary user. No A05 SDK or ARM-toolchain environment variables are needed:

```
sudo apt-get update
sudo apt-get install g++ pkg-config libjsoncpp-dev libsqlite3-dev libssl-dev libwebsockets-dev
    python3 openssl
python3 setup/local-integration.py --check
python3 setup/local-integration.py
```

Expected: package SHA256/prerequisite checks pass,the bundled matching0.6.18 protocol/HTTPS/WSS core compiles,40 protocol and33 actual TLS-transport checks pass,and local_integration=passed is reported. The automatic flow uses fictional accounts,new temporary databases,trusted temporary localhost certificates and a synthetic adapter. It runs delivery/query CLIs through new-person save → sendUser commit/ACK → synthetic event commit/ACK → HTTPS readback,and checks overwrite/role rejection,invalid ACK,offline replay and deduplication. Services stop and temporary certificates/databases are cleaned up. No face algorithm,A05 SDK or peripherals are loaded.

The local simulation ends here. Sections 3–6 cover device configuration and operation. Do not use simulator certificates, fictional passwords or synthetic images on a real device. If build dependencies are missing, install the system packages

A05 | 0.6.18 | 0.1.94-draft | EN

Page 3

Integration quick start / 0.6.18

01

Complete application source

Reference C++ application, UI assets, build scripts and packaging tools.

02

Bilingual developer guides

Chinese and English instructions for building, customizing and using the kit.

03

API and examples

Hardware / SDK examples, protocol manuals and WSS / HTTPS integration samples.

Before you deploy

Preserve the device’s preinstalled runtime, models and individual licence. Do not copy a licence from another device. SDK and toolchain delivery is subject to the applicable terms and matched component versions.

Application packages do not include a complete Linux system image. Full system rebuilding or recovery requires separate, matching BSP, rootfs, system image and recovery materials.

Developer materials are a preview. The official installation tool and application package are not yet available for download. Contact TIMYONE to confirm the applicable SDK / toolchain delivery and licensing terms.