ADH-Tech

# *Data Sheet*

*Optical Fingerprint Recognition* ***EMBEDDED*** *Module*

## *GT-511C3*

**2016/04/11**

**V2.1**

sales@adh-tech.com.tw

www.adh-tech.com.tw

ADH-Tech

# Contents

# 1. Concept

This device is one chip module with;

- fingerprint algorithm
- optical sensor

The major functions are the followings.

- High-accuracy and high-speed fingerprint identification technology
- Ultra-thin optical sensor
- 1:1 verification, 1:N identification
- downloading fingerprint image from the device
- Reading & writing fingerprint template(s) from/to the device
- Simple UART & USB communication protocol

Technical Specification

| **Item** | **Value** |
|---|---|
| CPU | ARM Cortex M3 Core |
| Sensor | optical Sensor |
| Effective area of the Sensor | 14 x 12.5(mm) |
| Image Size | 202 x 258 Pixels |
| Resolution | 450 dpi |
| The maximum number of fingerprints | 200 fingerprints |
| Matching Mode | 1:1, 1:N |
| The size of template | 496 Bytes (template) + 2 Bytes (checksum) |
| Communication interface | UART, default baud rate = 9600bps after power on<br>USB Ver1.1, Full speed |
| False Acceptance Rate (FAR) | < 0.001% |
| False Rejection Rate(FRR) | < 0.1% |
| Enrollment time | < 3 sec (3 fingerprints) |
| Identification time | < 1.0 sec (200 fingerprints) |
| Operating voltage | DC 3.3~6V |
| Operating current | < 130mA |

| | | |
|---|---|---|
| Operating environment | Temperature | -20℃ ~ +60℃ |
| | Humidity | 20% ~ 80% |
| Storage environment | Temperature | -20℃ ~ +60℃ |
| | Humidity | 10% ~ 80% |

# 2. Protocol: Packet Structure

**(Multi-byte item is represented as Little Endian.)**

**Command Packet (Command)**

| OFFSET | ITEM | TYPE | DESCRIPTION |
|---|---|---|---|
| 0 | 0x55 | BYTE | Command start code1 |
| 1 | 0xAA | BYTE | Command start code2 |
| 2 | *Device ID* | WORD | Device ID: default is 0x0001, always fixed |
| 4 | *Parameter* | DWORD | Input parameter |
| 8 | *Command* | WORD | Command code |
| 10 | *Check Sum* | WORD | Check Sum (byte addition)<br>OFFSET[0]+…+OFFSET[9]=*Check Sum* |

**Response Packet (Acknowledge)**

| OFFSET | ITEM | TYPE | DESCRIPTION |
|---|---|---|---|
| 0 | 0x55 | BYTE | Response start code1 |
| 1 | 0xAA | BYTE | Response start code2 |
| 2 | *Device ID* | WORD | Device ID: default is 0x0001, always fixed |
| 4 | *Parameter* | DWORD | **Response == 0x30:** (ACK) Output Parameter<br>**Response == 0x31:** (NACK) Error code |
| 8 | *Response* | WORD | **0x30:** Acknowledge (ACK).<br>**0x31:** Non-acknowledge (NACK). |
| 10 | *Check Sum* | WORD | Check Sum (byte addition)<br>OFFSET[0]+…+OFFSET[9]=*Check Sum* |

**Data Packet (Data)**

| OFFSET | ITEM | TYPE | DESCRIPTION |
|---|---|---|---|
| 0 | 0x5A | BYTE | Data start code1 |
| 1 | 0xA5 | BYTE | Data start code2 |
| 2 | *Device ID* | WORD | Device ID: default is 0x0001, always fixed |
| 4 | *Data* | N BYTES | N bytes Data<br>The size is pre-defined per protocol stage |
| 4+N | *Check Sum* | WORD | Check Sum (byte addition)<br>OFFSET[0]+…+OFFSET[4+N-1]=*Check Sum* |

# 3. Protocol: Commands Summary

In a command packet *Command* can be one of below.

| Number (HEX) | Alias | Description |
|---|---|---|
| 01 | *Open* | Initialization |
| 02 | *Close* | Termination |
| 03 | *UsbInternalCheck* | Check if the connected USB device is valid |
| 04 | *ChangeBaudrate* | Change UART baud rate |
| 05 | *SetIAPMode* | Enter IAP Mode<br>In this mode, FW Upgrade is available |
| 12 | *CmosLed* | Control CMOS LED |
| 20 | *GetEnrollCount* | Get enrolled fingerprint count |
| 21 | *CheckEnrolled* | Check whether the specified ID is already enrolled |
| 22 | *EnrollStart* | Start an enrollment |
| 23 | *Enroll1* | Make 1st template for an enrollment |
| 24 | *Enroll2* | Make 2nd template for an enrollment |
| 25 | *Enroll3* | Make 3rd template for an enrollment, merge three templates into one template, save merged template to the database |
| 26 | *IsPressFinger* | Check if a finger is placed on the sensor |
| 40 | *DeleteID* | Delete the fingerprint with the specified ID |
| 41 | *DeleteAll* | Delete all fingerprints from the database |
| 50 | *Verify* | 1:1 Verification of the capture fingerprint image with the specified ID |
| 51 | *Identify* | 1:N Identification of the capture fingerprint image with the database |
| 52 | *VerifyTemplate* | 1:1 Verification of a fingerprint template with the specified ID |
| 53 | *IdentifyTemplate* | 1:N Identification of a fingerprint template with the database |

| Number (HEX) | Alias | Description |
|---|---|---|
| 60 | *CaptureFinger* | Capture a fingerprint image(256x256) from the sensor |
| 61 | *MakeTemplate* | Make template for transmission |
| 62 | *GetImage* | Download the captured fingerprint image(256x256) |
| 63 | *GetRawImage* | Capture & Download raw fingerprint image(320x240) |
| 70 | *GetTemplate* | Download the template of the specified ID |
| 71 | *SetTemplate* | Upload the template of the specified ID |
| 72 | *GetDatabaseStart* | Start database download, **obsolete** |
| 73 | *GetDatabaseEnd* | End database download, **obsolete** |
| 80 | *UpgradeFirmware* | Not supported |
| 81 | *UpgradeISOCDImage* | Not supported |
| F0 | *SetSecurityLevel* | Set Security Level |
| F1 | *GetSecurityLevel* | Get Security Level |
| 30 | *Ack* | Acknowledge. |
| 31 | *Nack* | Non-acknowledge. |

# 4. Protocol: Error Codes

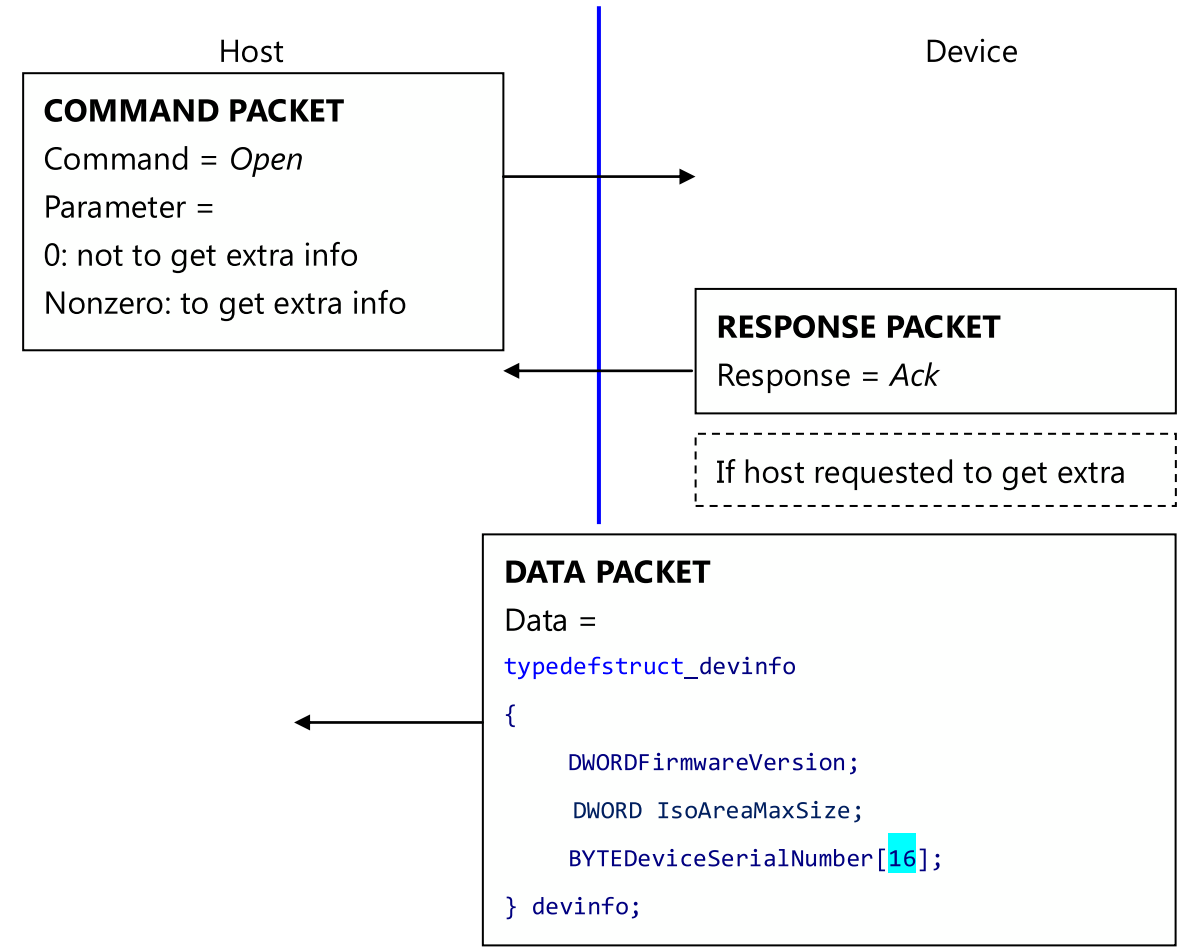
When response packet is Non-acknowledge, *Parameter* represents an error code as below.

| NACK Parameter | Value | Description |
|---|---|---|
| NACK_TIMEOUT | 0x1001 | **Obsolete**, capture timeout |
| NACK_INVALID_BAUDRATE | 0x1002 | **Obsolete**, Invalid serial baud rate |
| NACK_INVALID_POS | 0x1003 | The specified ID is not between 0~199 |
| NACK_IS_NOT_USED | 0x1004 | The specified ID is not used |
| NACK_IS_ALREADY_USED | 0x1005 | The specified ID is already used |
| NACK_COMM_ERR | 0x1006 | Communication Error |
| NACK_VERIFY_FAILED | 0x1007 | 1:1 Verification Failure |
| NACK_IDENTIFY_FAILED | 0x1008 | 1:N Identification Failure |
| NACK_DB_IS_FULL | 0x1009 | The database is full |
| NACK_DB_IS_EMPTY | 0x100A | The database is empty |
| NACK_TURN_ERR | 0x100B | **Obsolete**, Invalid order of the enrollment (The order was not as: **EnrollStart -> Enroll1 -> Enroll2 -> Enroll3**) |
| NACK_BAD_FINGER | 0x100C | Too bad fingerprint |
| NACK_ENROLL_FAILED | 0x100D | Enrollment Failure |
| NACK_IS_NOT_SUPPORTED | 0x100E | The specified command is not supported |
| NACK_DEV_ERR | 0x100F | Device Error, especially if Crypto-Chip is trouble |
| NACK_CAPTURE_CANCELED | 0x1010 | **Obsolete**, The capturing is canceled |
| NACK_INVALID_PARAM | 0x1011 | Invalid parameter |
| NACK_FINGER_IS_NOT_PRESSED | 0x1012 | Finger is not pressed |

| | | |
|---|---|---|
| Duplicated ID | 0 – 199 | There is duplicated fingerprint (while enrollment or setting template), This error describes just duplicated ID |

# 5. Protocol: Command Details

### 5.1. Initialization(*Open*)



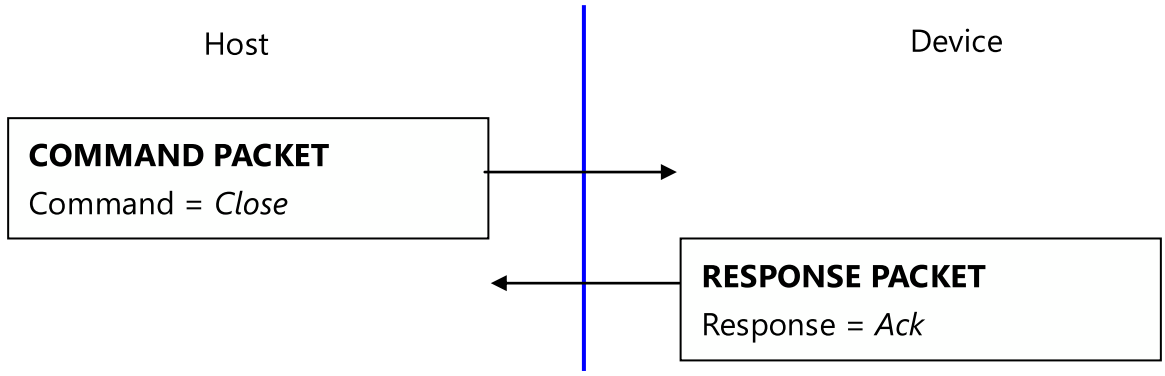

*Open* command is used to initialize the device; especially it gets device's static info.

**Description of *devinfo* structure**

| Field | Sample | Description |
|---|---|---|
| *FirmwareVersion* | FirmwareVersion: 20120225 | Firmware version |
| *IsoAreaMaxSize* | IsoAreaMaxSize: 0 KB | Maximum size of ISO CD image |
| *DeviceSerialNumber* | DeviceSN: EF15EF4016C66250-888F1A4139000000 | Unique serial number of the device |

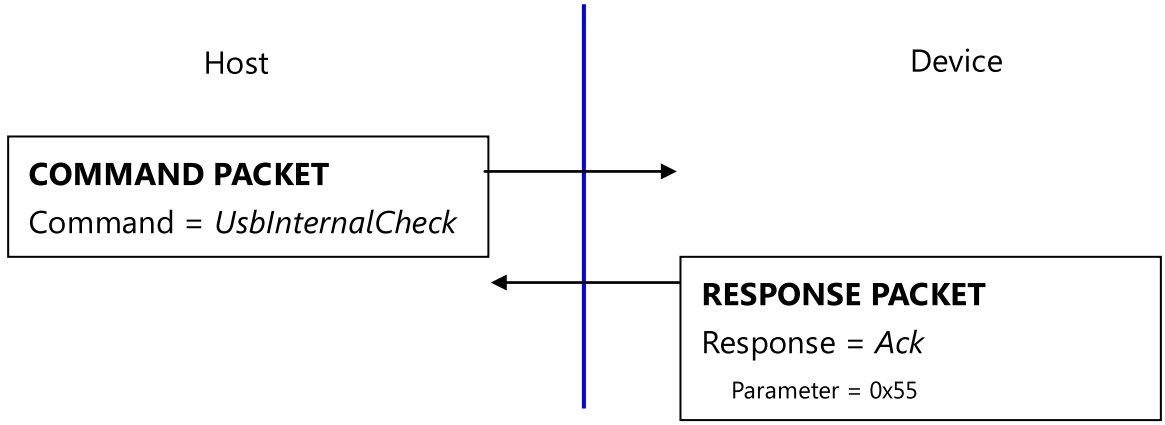
If the Device's Serial Number is zero, then there is no guarantee for stable operation of the device.

### 5.2. Termination(*Close*)



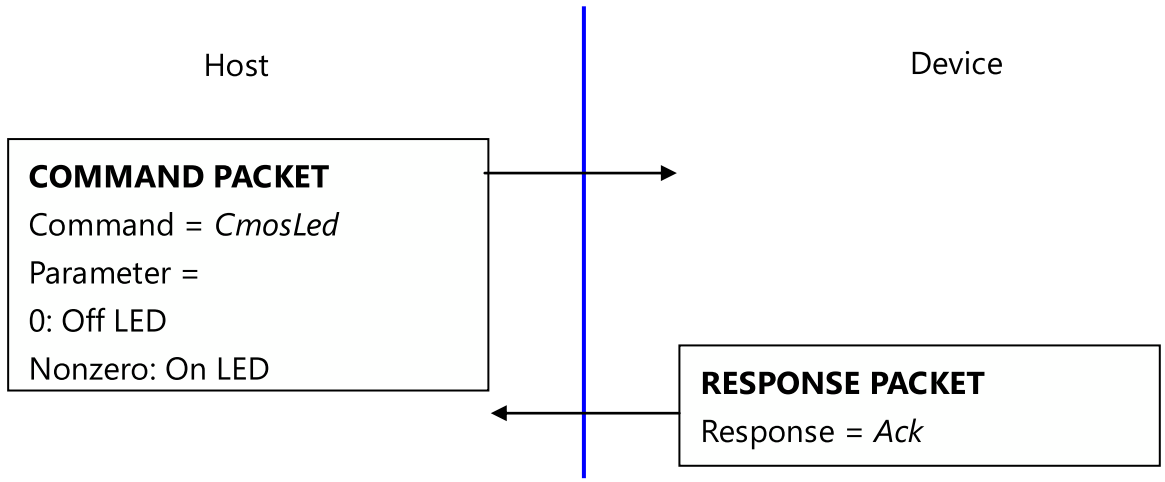

*Close* command does nothing.

### 5.3. Fast searching of the device(*UsbInternalCheck*)



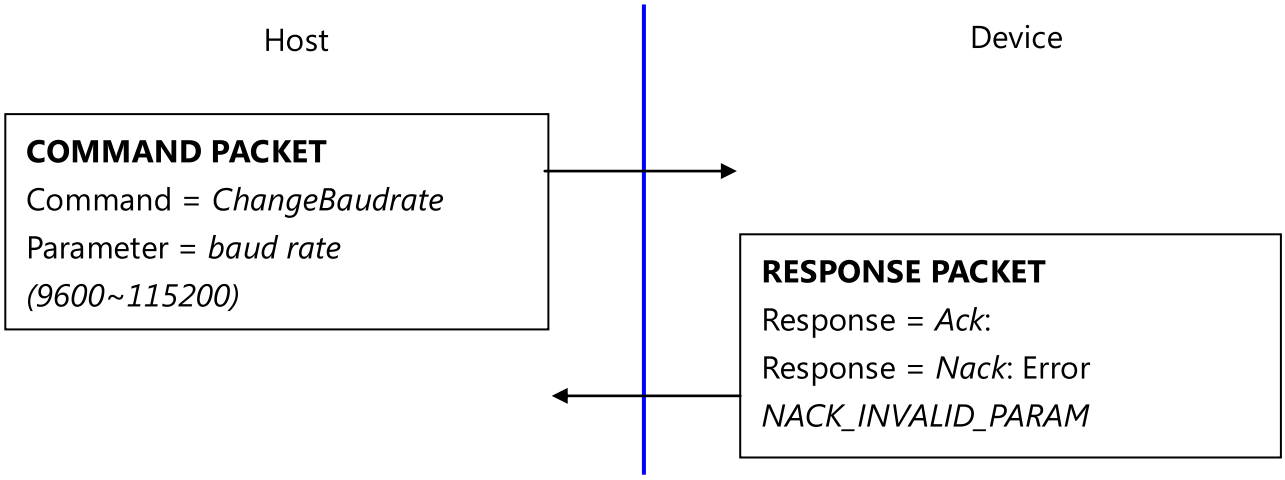

The device operates as removable CD drive. If another removable CD drive exists in the system, connection time maybe will be long. To prevent this, *UsbInternalCheck* command is used for fast searching of the device.

### 5.4. CMOS LED control(CmosLed)




Default state of CMOS (Sensor) LED is OFF state.
(But while booting, LED blinks once, this says the LED is OK.)
Therefore, please issue LED ON command prior to any capture.

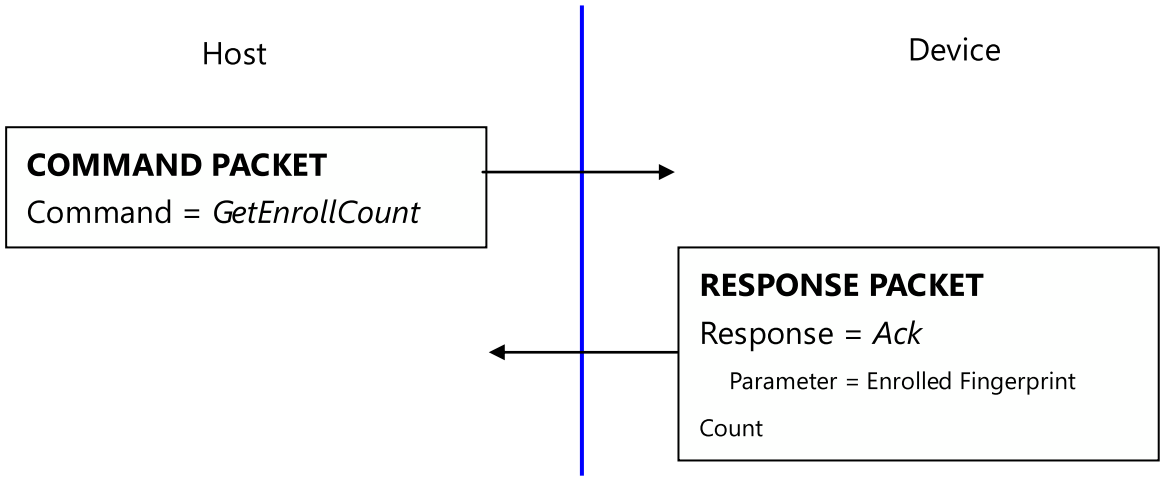
### 5.5. Changing UART baud rate (*ChangeBaudrate*)



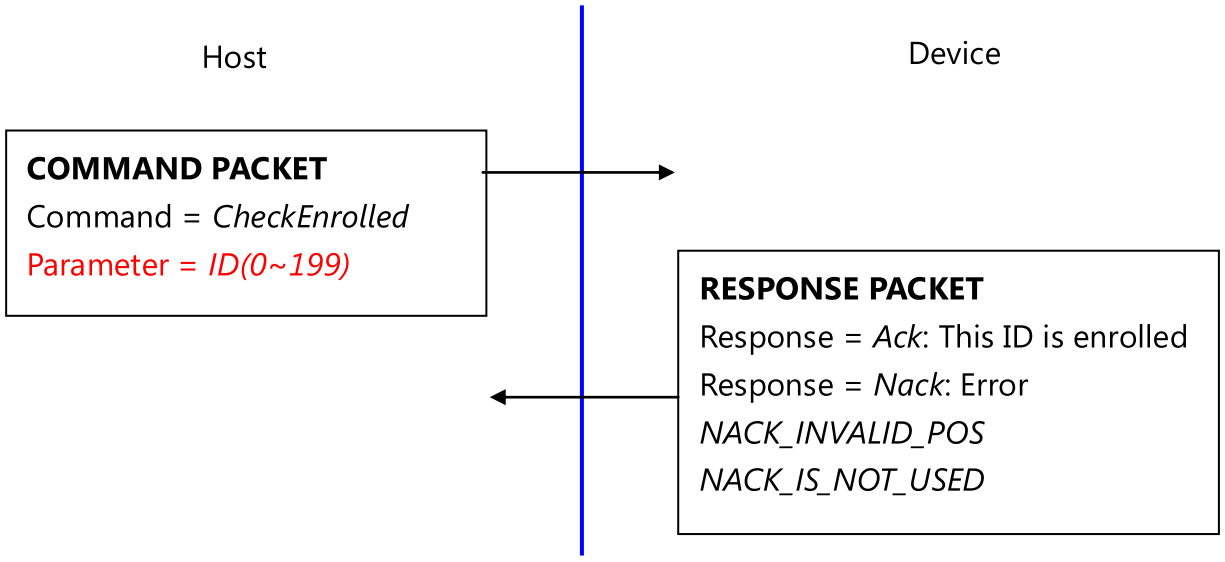

This command changes the UART baud rate at the run-time.

The device initializes its UART baud rate to 9600 bps after power on.

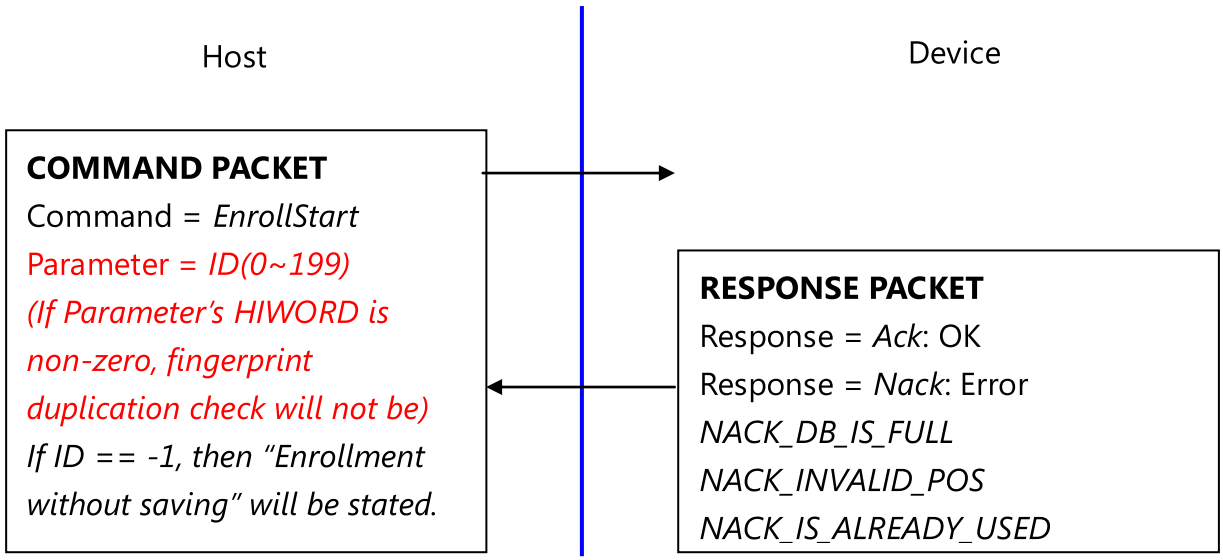
### 5.6. Get enrolled fingerprint count(*GetEnrollCount*)



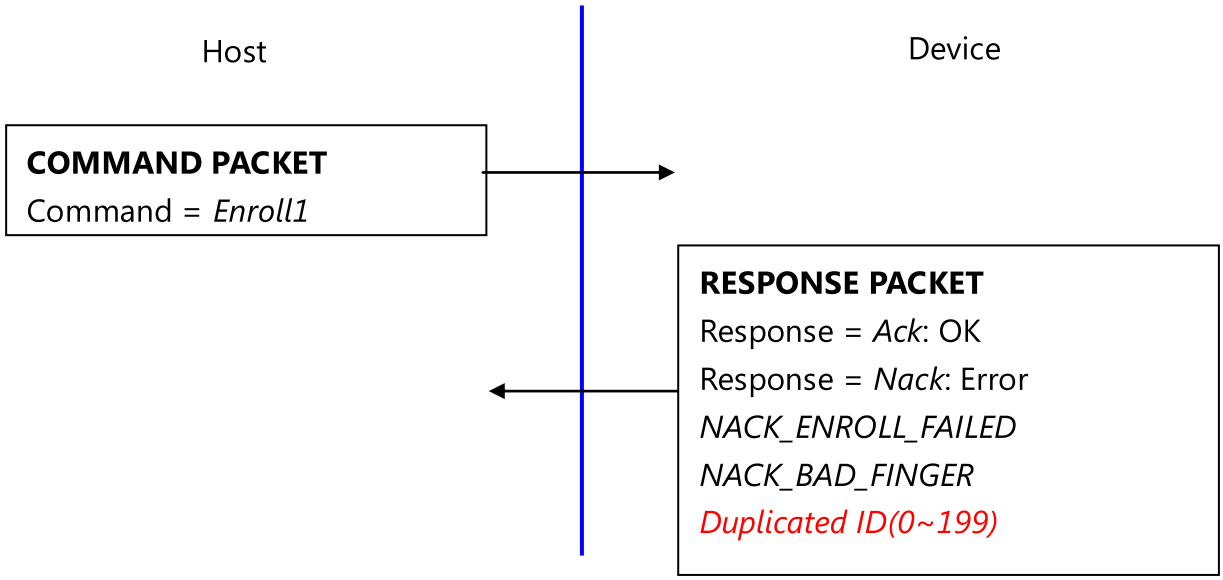

### 5.7. Check enrollment status(*CheckEnrolled*)



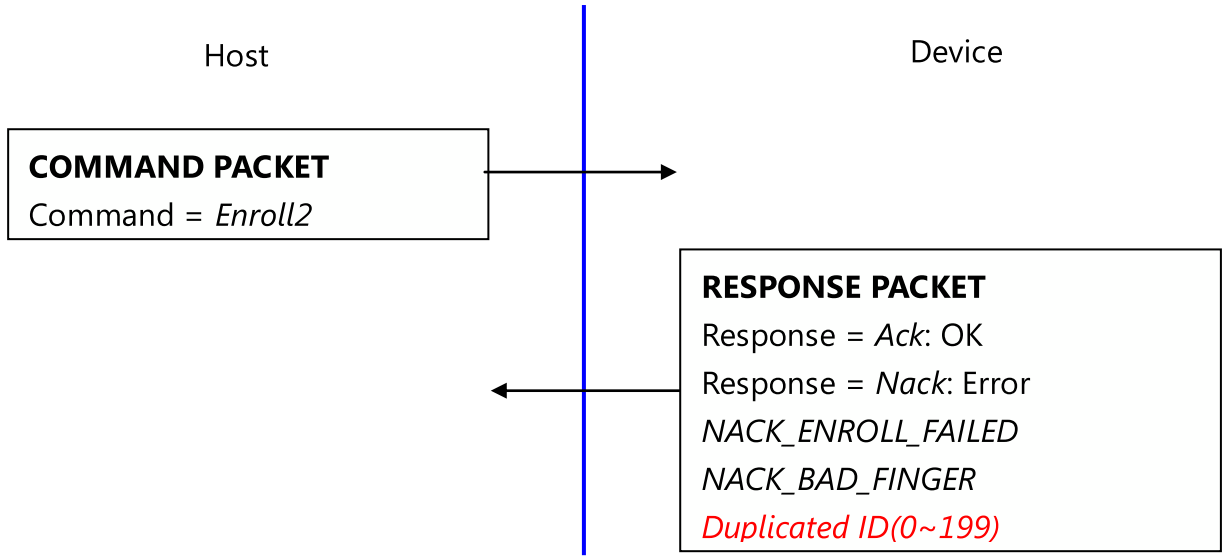
### 5.8. Start an enrollment(*EnrollStart*)



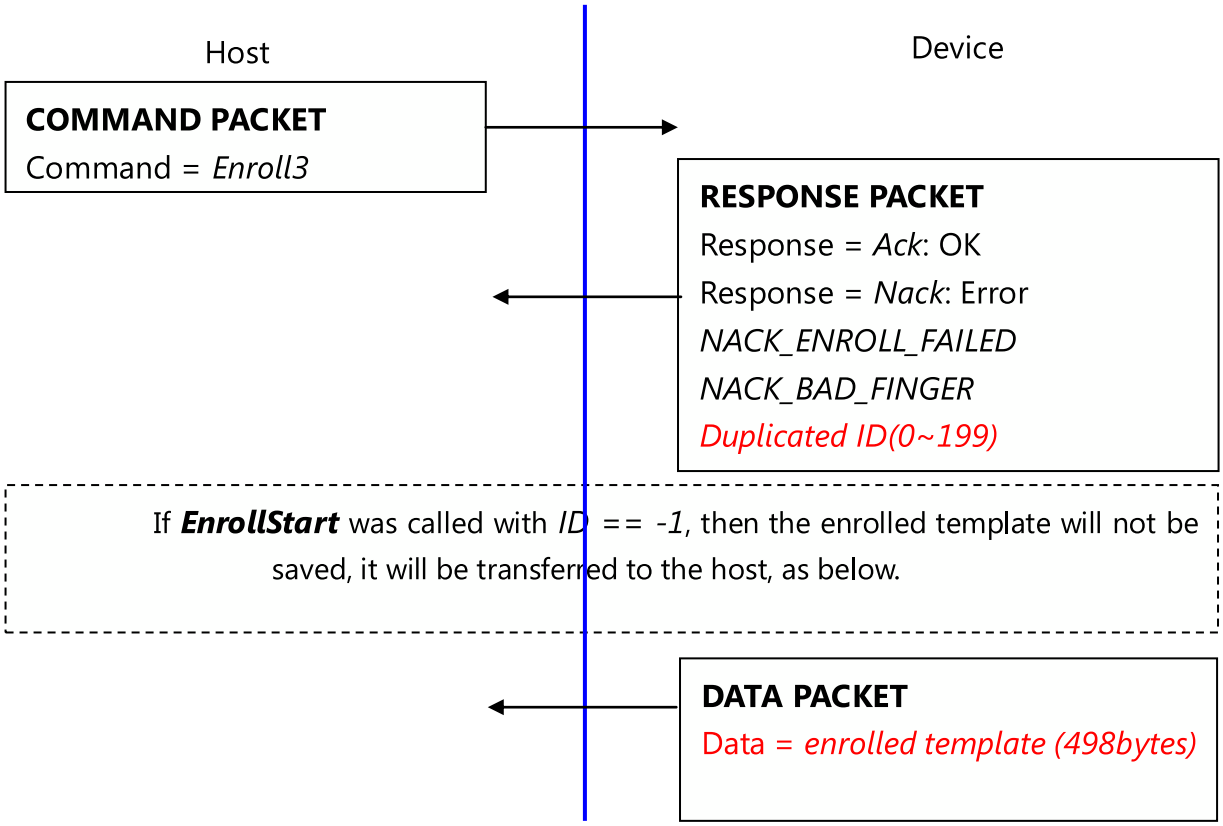

### 5.9. Make 1st template for an enrollment(*Enroll1*)



### 5.10. Make 2nd template for an enrollment(*Enroll2*)

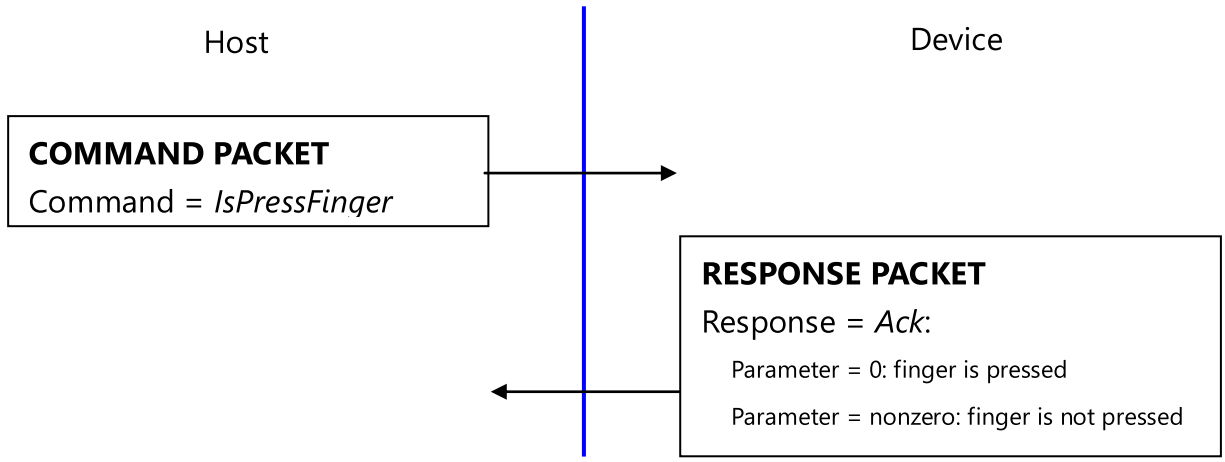



### 5.11. Make 3rd template for an enrollment, merge three templates(*Enroll3*)



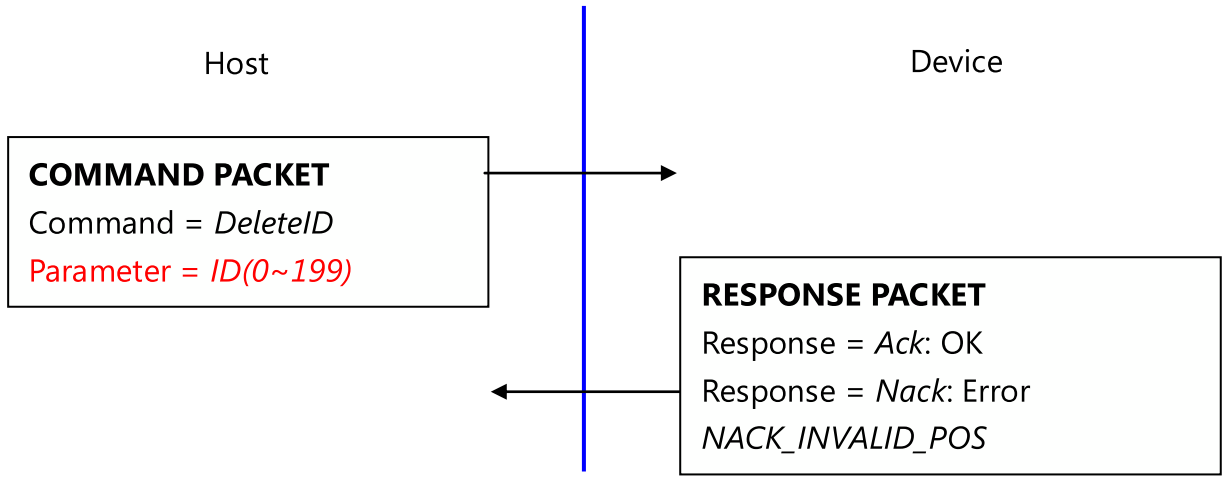

To enroll a fingerprint, the host must issue above 4 commands, later chapter describes how to organize these commands.

### 5.12. Check finger pressing status(*IsPressFinger*)

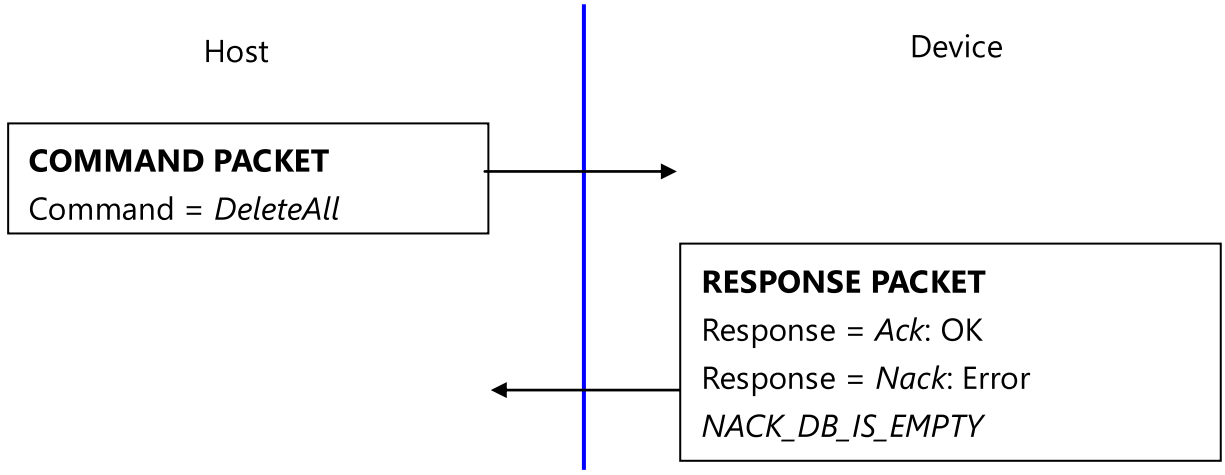



This command is used while enrollment, the host waits to take off the finger per enrollment stage.

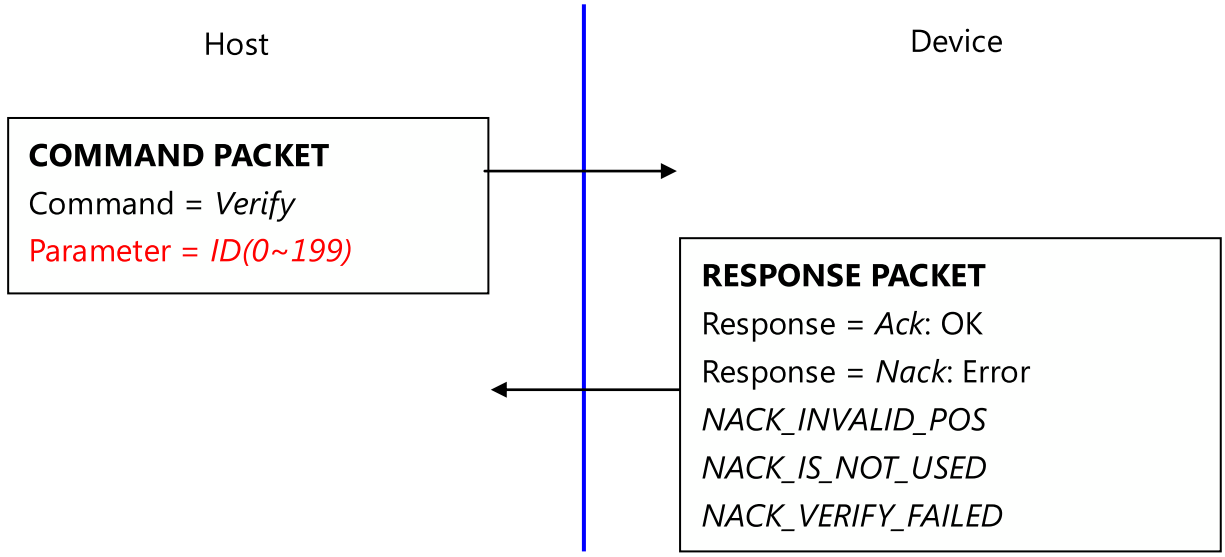
### 5.13. Delete one fingerprint(*DeleteID*)



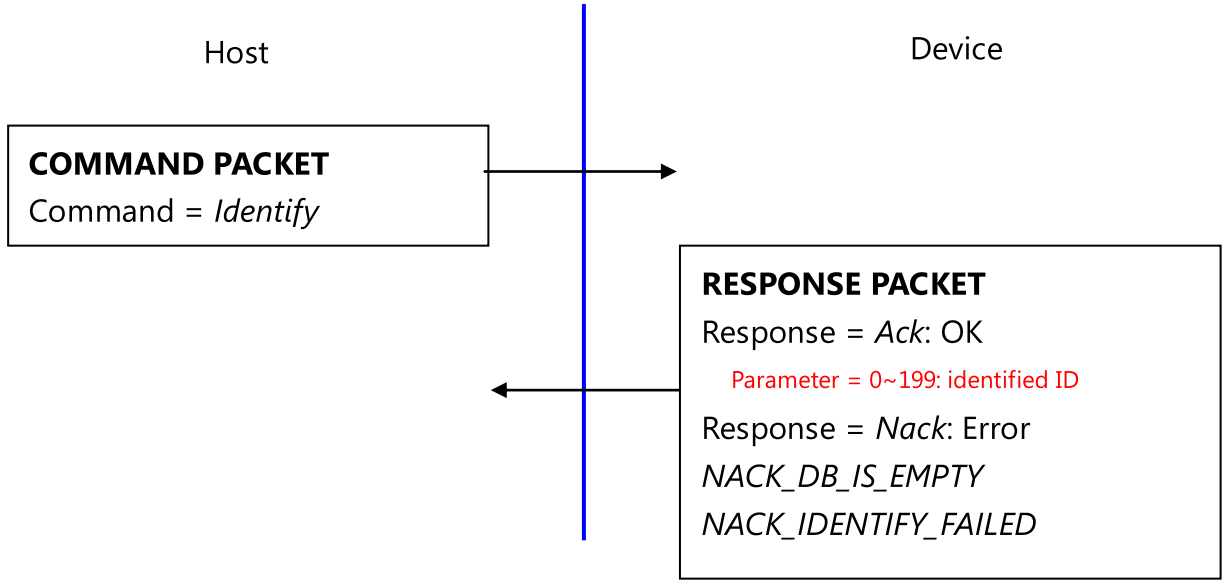

### 5.14. Delete all fingerprints(*DeleteAll*)



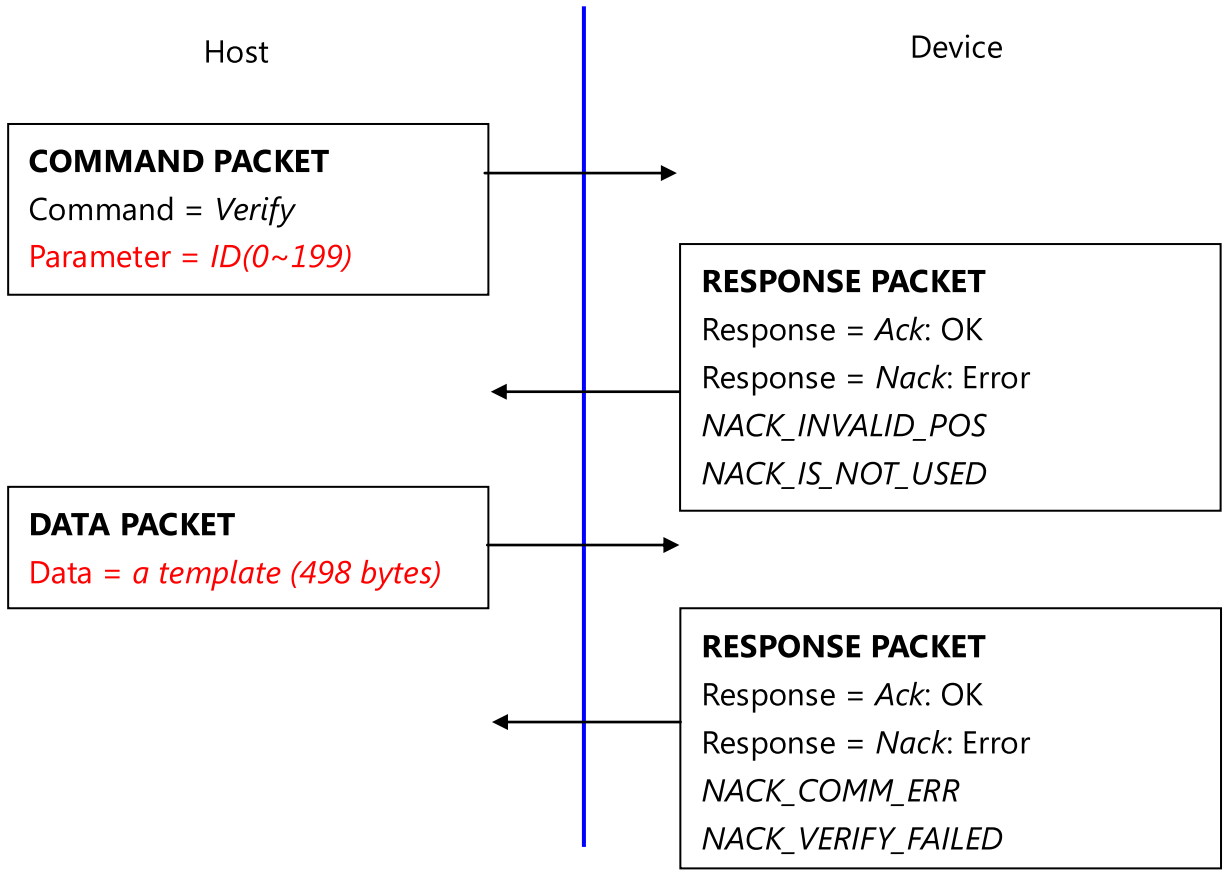
### 5.15. 1:1 Verification(*Verify*)



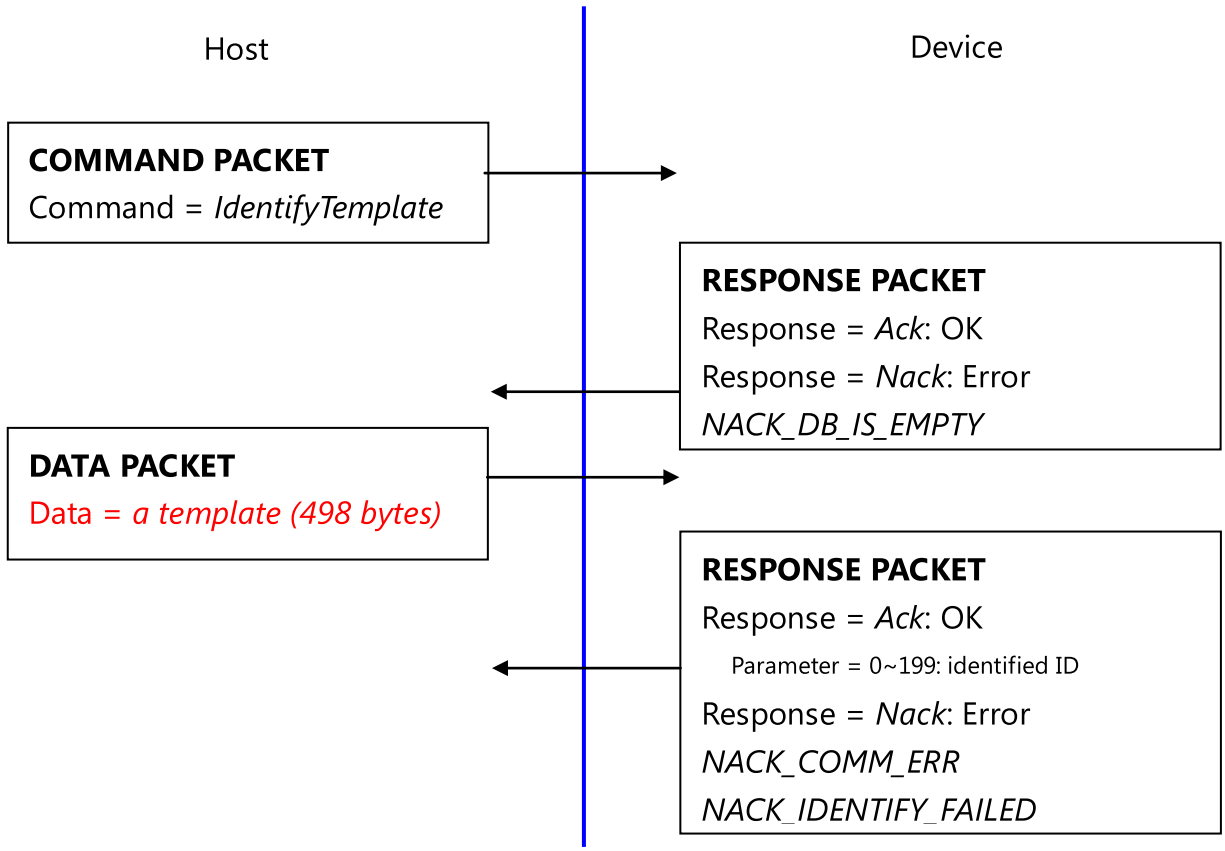

### 5.16. 1:N Identification(*Identify*)



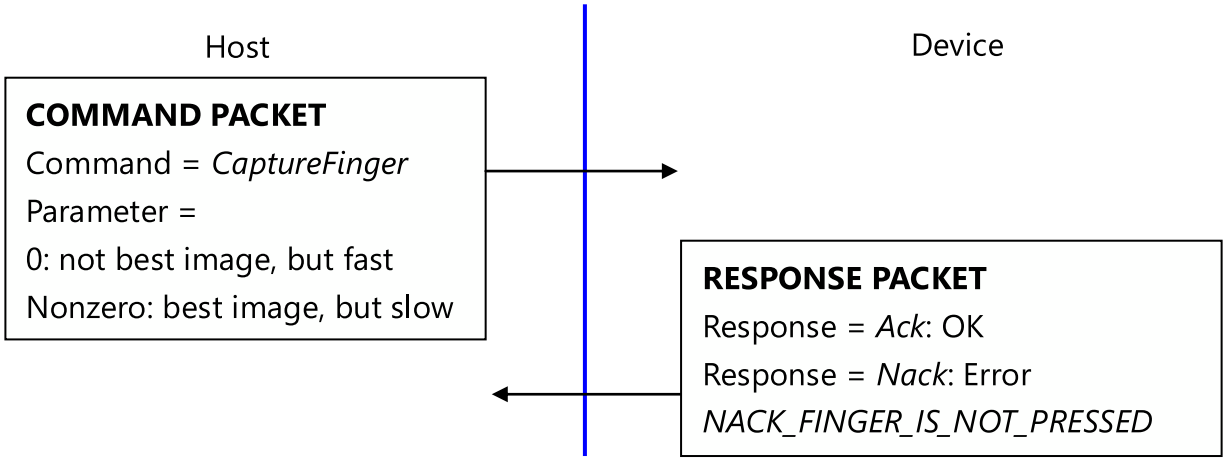
### 5.17. 1:1 Verification of Template(*VerifyTemplate*)



### 5.18. 1:N Identification of Template(*IdentifyTemplate*)

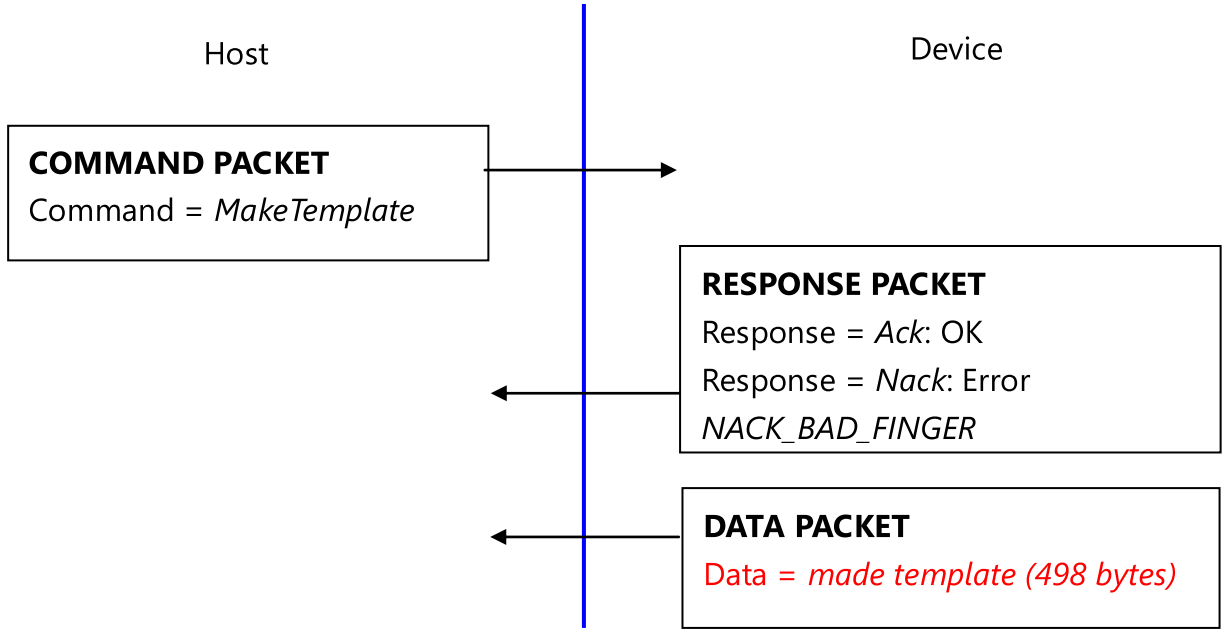


### 5.19. Capture fingerprint(*CaptureFinger*)



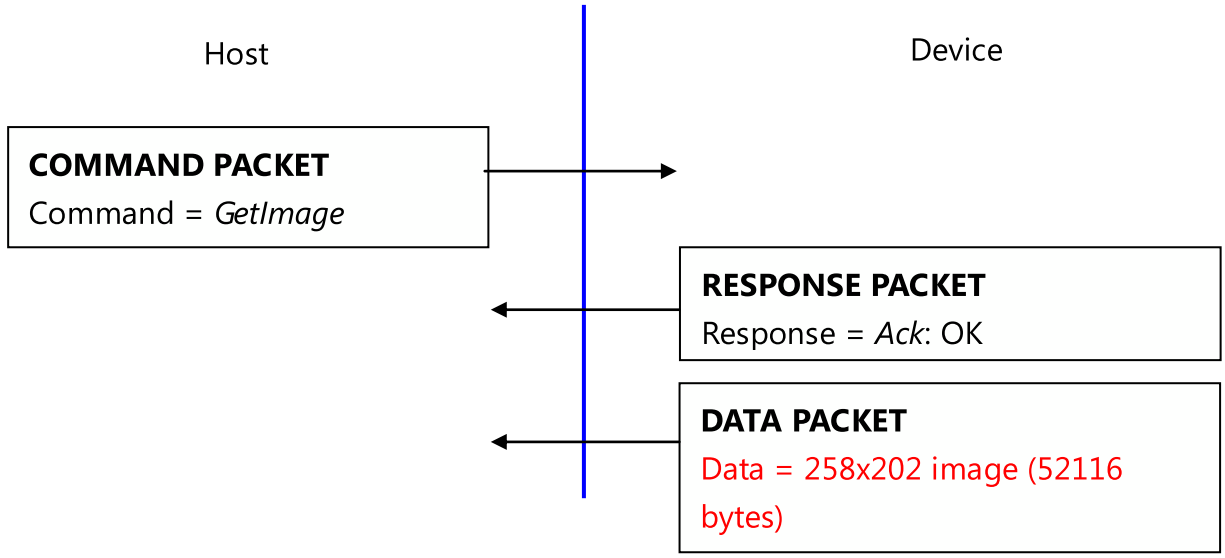

The fingerprint algorithm uses 450dpi 256x256 image for its input.
This command captures raw image from the sensor and converts it to 256x256 image for the fingerprint algorithm. If the finger is not pressed, this command returns with non-acknowledge.
Please use best image for enrollment to get best enrollment data.
Please use not best image for identification (verification) to get fast user sensibility.

### 5.20. Make Template(*MakeTemplate*)

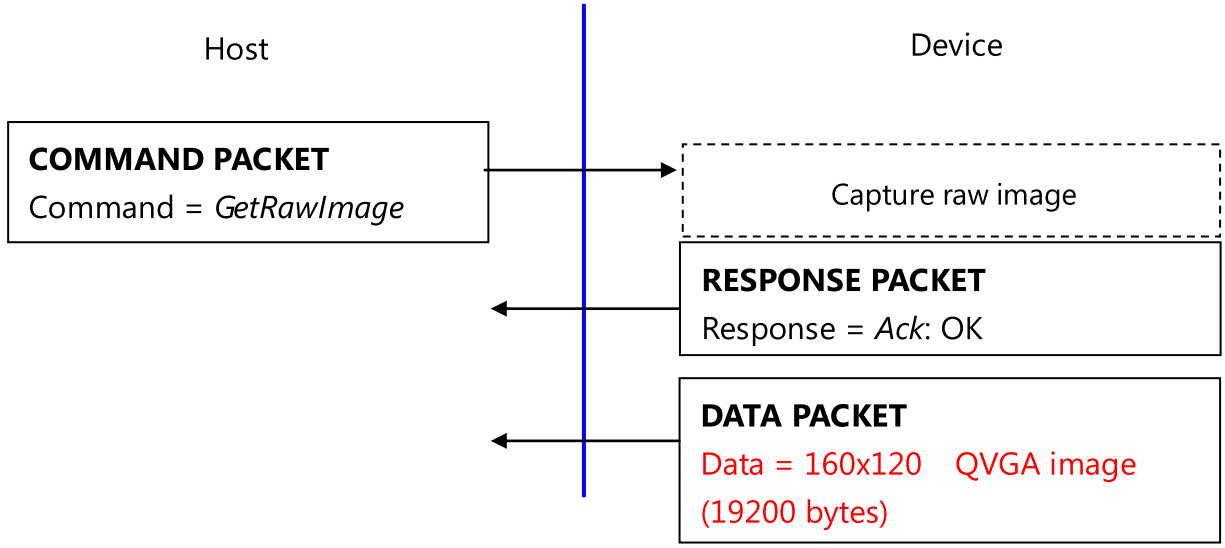



This function makes template for transmission. *CaptureFinger* command should be previously issued. Do not use the template for registration.

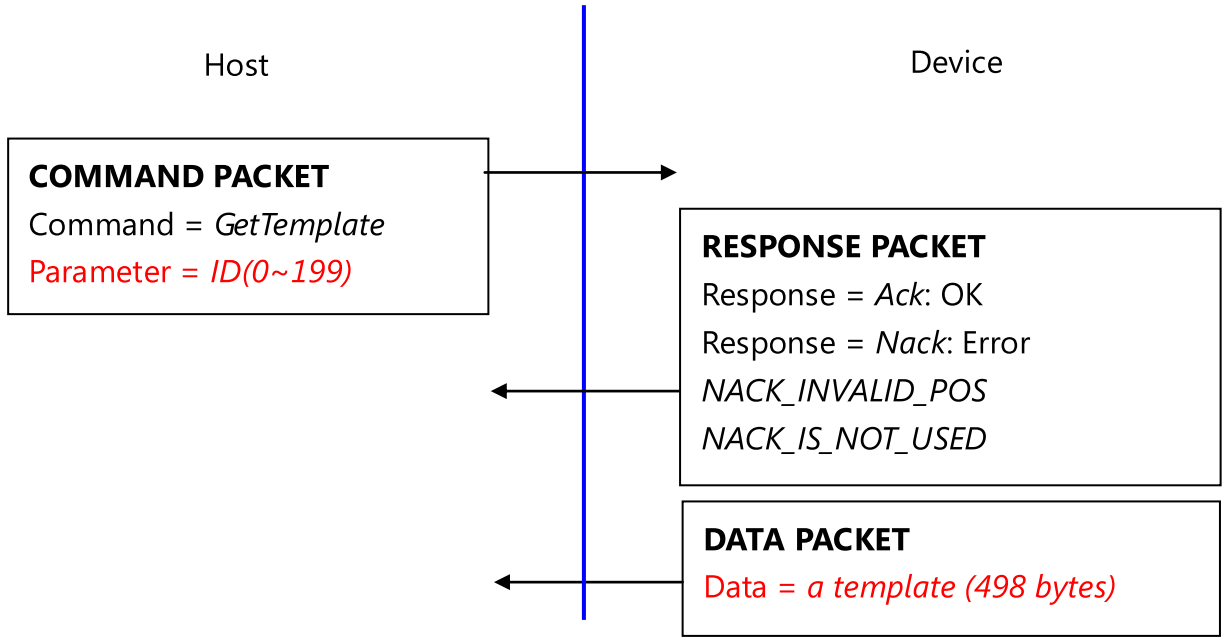
### 5.21. Get fingerprint image(*GetImage*)



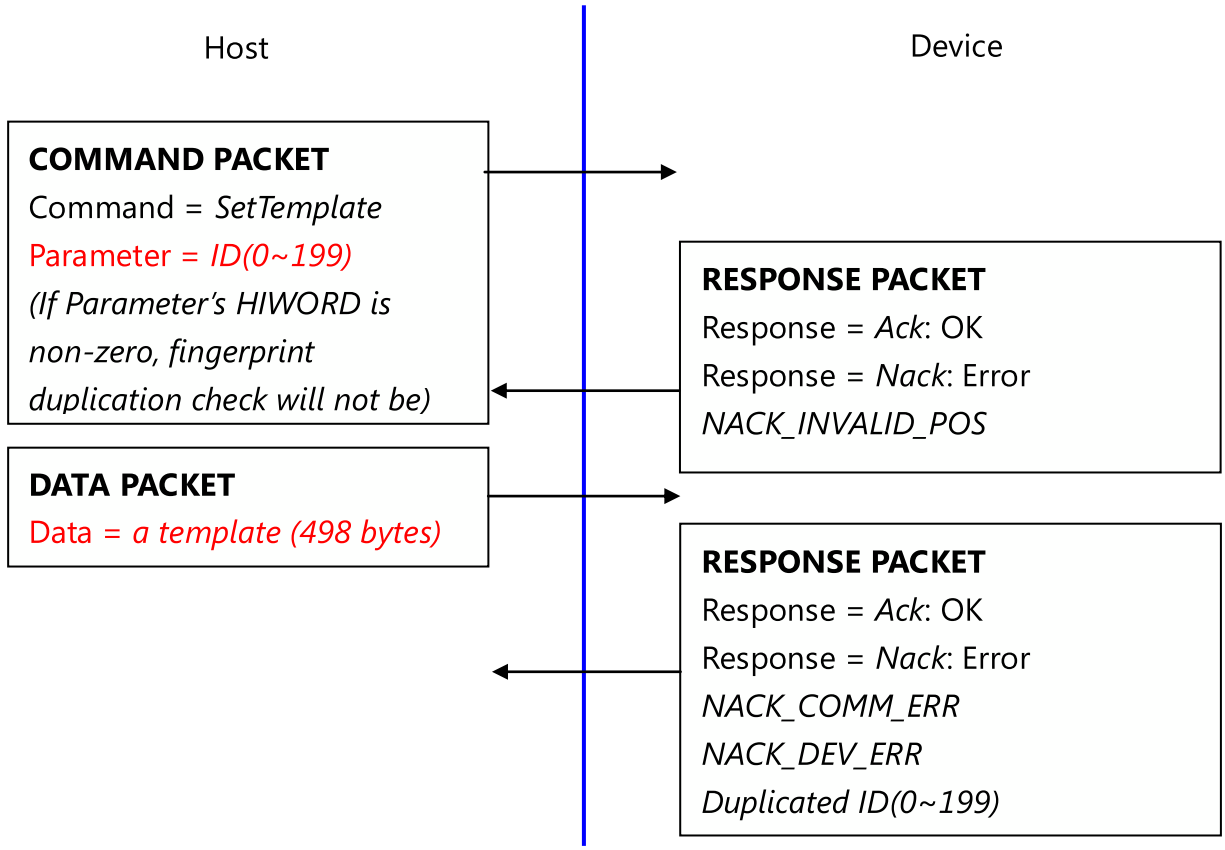
### 5.22. Get raw image(*GetRawImage*)



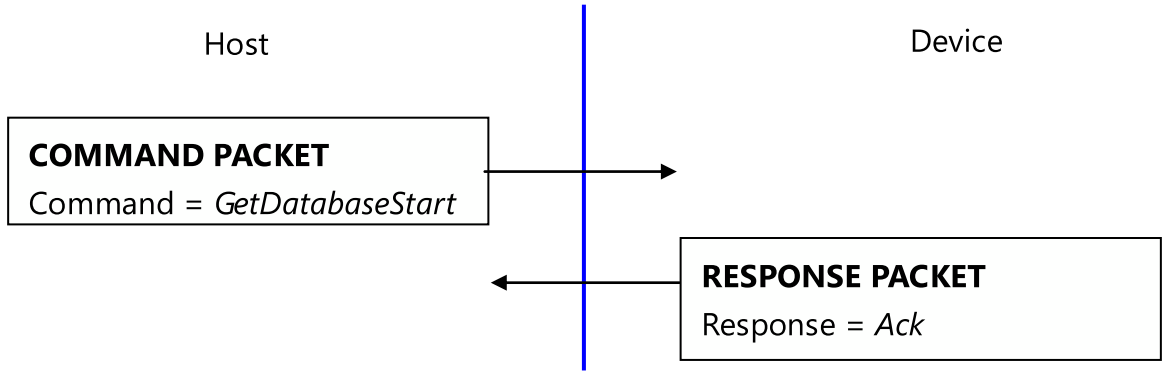

### 5.23. Get template(*GetTemplate*)



### 5.24. Set template(*SetTemplate*)

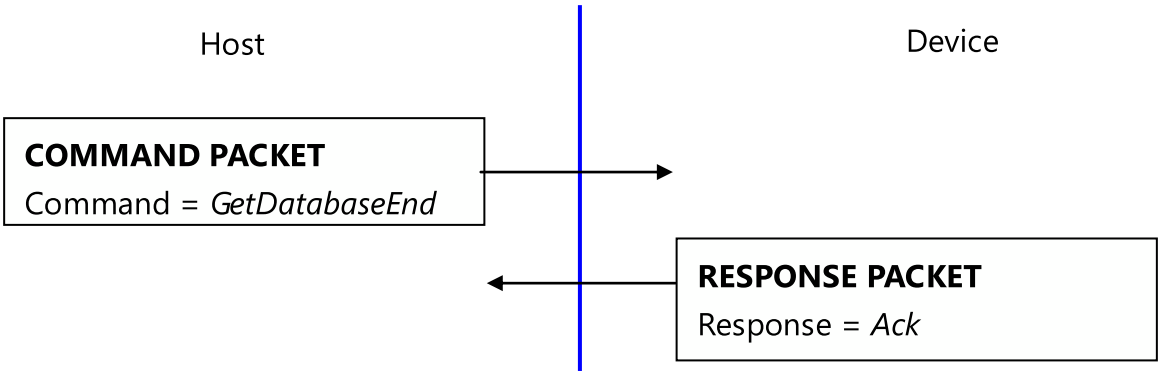



### 5.25. Start database download, obsolete(*GetDatabaseStart*)




*GetDatabaseStart* command does nothing. It exists for historical reason; it was used for RS232 communication.

### 5.26. End database download, obsolete(*GetDatabaseEnd*)




*GetDatabaseEnd* command does nothing. It exists for historical reason; it was used for RS232 communication.

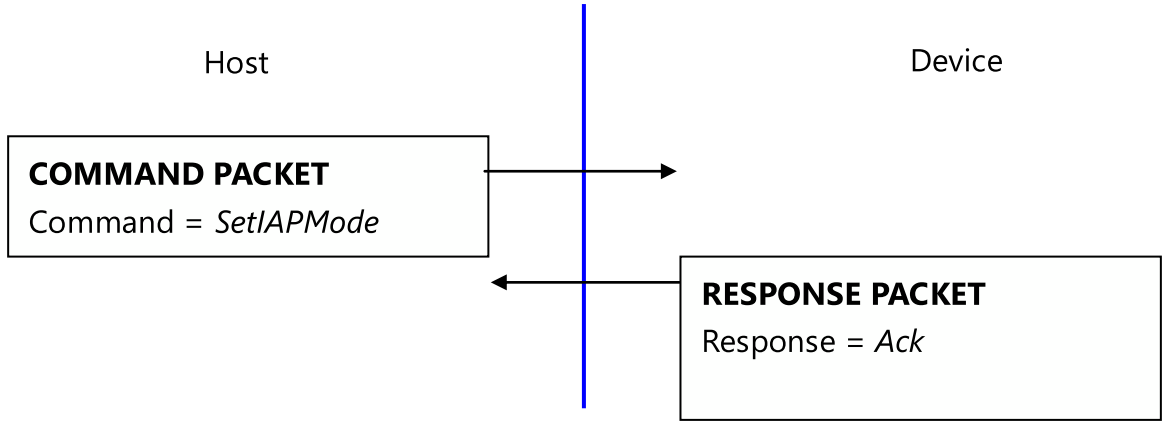
### 5.27. Upgrade Firmware(*UpgradeFirmware*)

Not supported

### 5.28. Upgrade ISO CD Image(UpgradeISOCDImage)

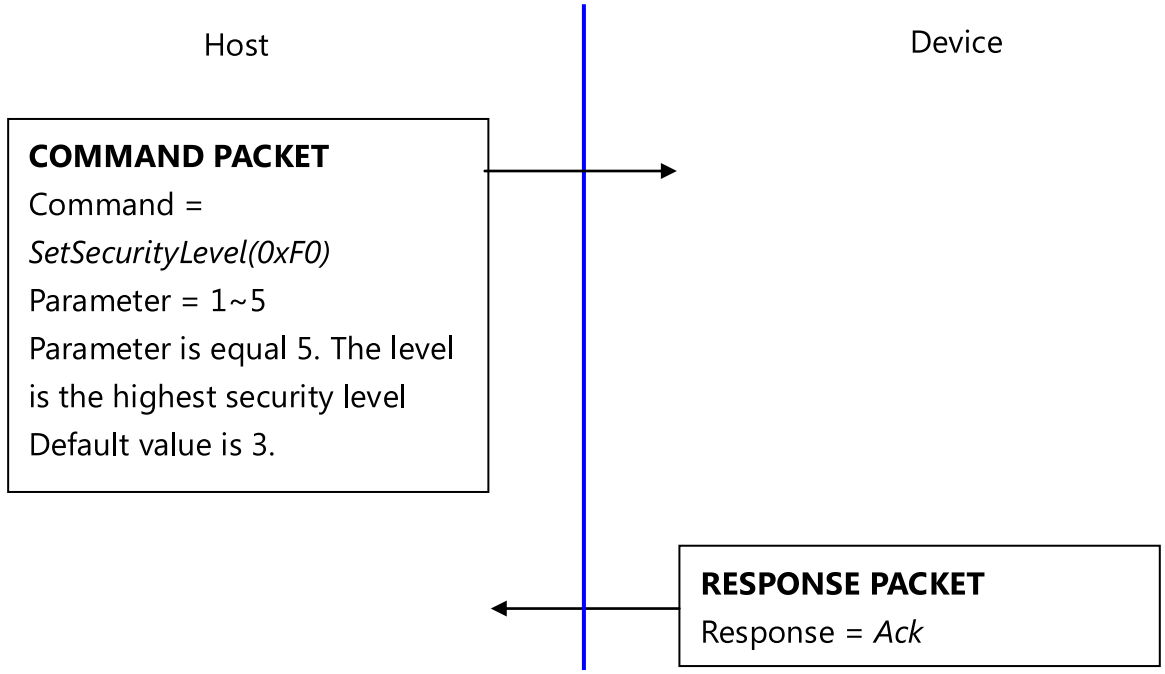
Not supported

### 5.29. Set IAP Mode(SetIAPMode)

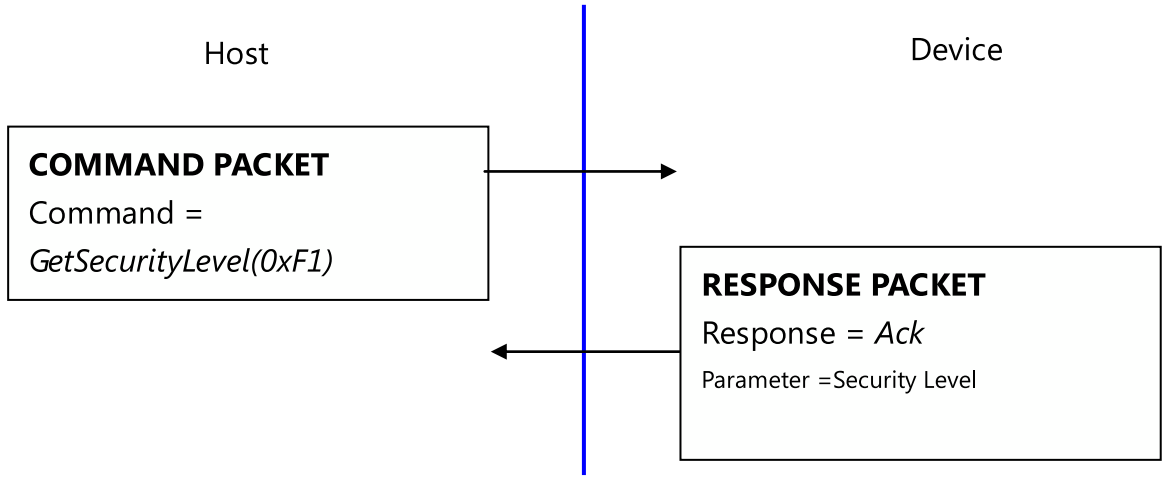



The Device enter in IAP Mode,
In this mode, FW upgrade is available.

### 5.30. Set SecurityLevel(0xF0)




### 5.31. GetSecurityLevel(*0xF1*)



# 6. Protocol: Flowchart, description

## 6.1 Capture of the fingerprint image

*IsPressFinger* checks whether a finger placed on the sensor.
This function is used especially while enrollment.

*CaptureFinger* captures a fingerprint image (256x256), if a finger isn't placed on the sensor, it returns with error.
If this function returns with success, the device's internal RAM keeps valid fingerprint image for the subsequent commands. If the host issues other command, the fingerprint image will be used and destroyed.

*GetRawImage* captures a raw live image (320x240), it doesn't check whether a finger placed on the sensor, this function is used for debug or calibration.

## 6.2 Identifying and Verifying

*Identify* and *IdentifyTemplate* perform 1: N matching operation.
*Verify* and *VerifyTemplate* perform 1: 1 matching operation.

Just before calling of image-related matching functions (*Identify*, *Verify*), the host must call *CaptureFinger*.

## 6.3 Enrollment

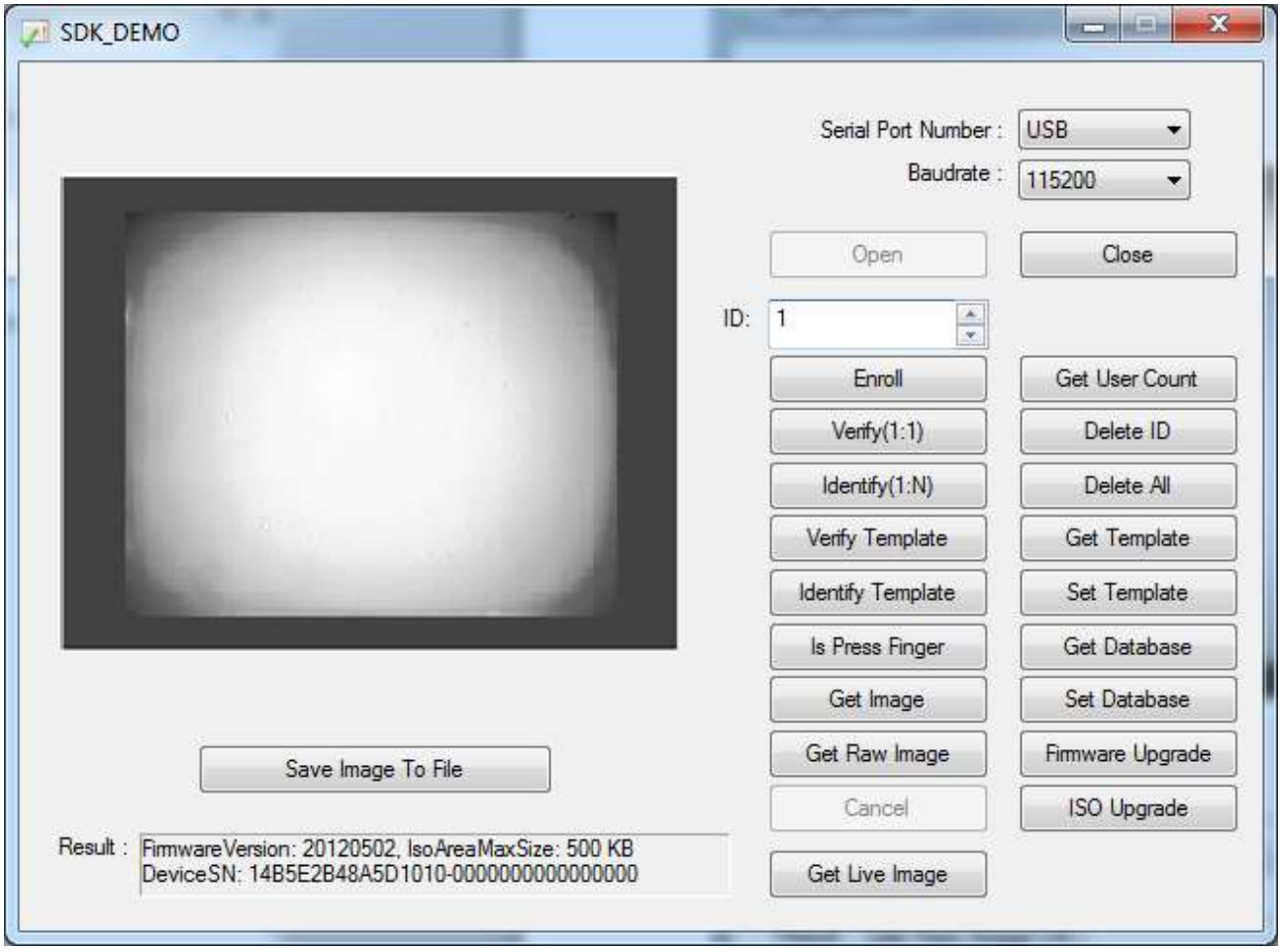
An enrollment flowchart is as below.

1. *EnrollStart* with a (not used) ID
2. *CaptureFinger*
3. *Enroll1*
4. Wait to take off the finger using *IsPressFinger*

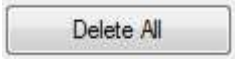
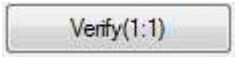
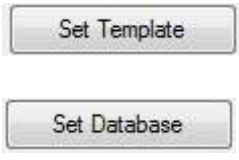
5. *CaptureFinger*
6. *Enroll2*
7. Wait to take off the finger using *IsPressFinger*
8. *CaptureFinger*
9. *Enroll3*

# 7. PC Demo

PC demo program describes how to use the device with its source code.




| Command Alias | UI item to test it |
|---|---|
| *Open, UsbInternalCheck, ChangeBaudrate* | Open |
| *Close* | Close |
| *GetEnrollCount* | Get User Count |

| Command Alias | UI item to test it |
|---|---|
| *CheckEnrolled, EnrollStart, Enroll1, Enroll2, Enroll3, IsPressFinger* |   |
| *DeleteID* |   |
| *DeleteAll* |   |
| *Verify* |   |
| *Identify* |   |
| *VerifyTemplate* |   |
| *IdentifyTemplate* |   |
| *CaptureFinger, GetImage* |   |
| *GetRawImage* |   |
| *GetTemplate, GetDatabaseStart, GetDatabaseEnd* |   |
| *SetTemplate* |   |
| *UpgradeFirmware* |   |
| *UpgradeISOCDImage* |   |

Demo program is supported with its source code.

The project is Microsoft Visual C++ 6.0 project.
We selected VC6.0 to minimize the size of the executable.

The demo program checks whether it is running on removable CD drive, if it is the case, it copies itself to "*My Document*" folder and executes copied version. This is for direct access to the device's removable CD drive.