IDEX BIOMETRIC AUTHENTICATOR INTEGRATION GUIDE

Last updated: December 2023

Document revision 1.0

Table of contents

Chapter 1.Introduction

This document defines the specification of the IDEX Biometric Authenticator (IBA) Applet. The IBA Applet provides to users the ability to enroll their fingerprints within their IDEX biometric card using an external platform either directly with the IBA Applet or indirectly via another installed applet.

It also allows one or more applets to perform biometric verification against the enrolled fingerprints in order to perform user verification.

1.1. Intended Audience

This document is intended for enrolment application developers, and for applet developers who wish to make use of the IDEX biometric capability to perform user verification.

1.2. Revision History

Version	Date	Changes
0.1	May-2023	Initial Draft
0.2	Jun-2023	Minor changes after initial draft review for IBA version 1.0
0.3	Jul-2023	Modified shareable interface API. Added DeleteFinger to shareable interface for IBA version 1.2
0.4	Aug-2023	Added DeleteFinger to APDU interface for IBA version 1.3. Renamed Verify Enroll sections to Enroll Qualification (and linked the two terms). Removed ability to delete templates from the Single Enroll command. Modified BIST command to indicate mandatory usage of external BIST limits.
0.5	Aug-2023	Added GetVersion and GetBTC commands Updated ETC, ETL, ETR acronyms
0.6	Sep-2023	Minor changes to correct typos
0.7	Oct-2023	Update system release version
1.0	Dec-2023	Move document from draft to release. Add details on match events that cause BTC to be incremented

1.3. References

Standard / Specification	Description	Reference
Java Card API v3.0.5	Java Card Platform API, Version 3.0.5	JCAPI
GlobalPlatform Card Specification v2.3.1	GlobalPlatform Card Specification, Version 2.3.1	GPCS
ISO/IEC 7816-4:2005	Identification cards – Integrated circuit cards – Part 4: Organization, security and commands for interchange	ISO7816-4
EMV_v4.3_Book_3 Application Specifcation	Defines the terminal and integrated circuit card	EMV Book 3

	(ICC) procedures necessary to effect a payment system transaction in an international interchange environment	

1.4. Terminology and Definitions

The following meanings apply to SHALL, SHOULD, and MAY in this document:

- SHALL indicates that the statement containing the SHALL must be followed as defined in this document.
- SHOULD indicates a recommendation. It is strongly recommended to follow the statement as defined in this document.
- MAY indicates an option.

The following table defines the expressions used within this document that use an upper case first letter in each word of the expression. Expressions within this document that use a lower case first letter in each word take the common-sense meaning.

Term	Definition
Application	Instance of an Executable Module after it has been installed.
Application Protocol Data Unit (APDU)	Standard communication messaging protocol between a card accepting device and a smart card.
Life Cycle	The existence of Card Content on a GlobalPlatform card and the various stages of this existence where applicable; or the stages in the life of the card itself.
Life Cycle State	A specific state within the Life Cycle of the card or of Card Content.
Secure Channel	A communication mechanism between an off-card entity and a card that provides a level of assurance, to one or both entities.
Security Domain	Application having the Security Domain privilege. This on-card entity provides support for the control, security, and communication requirements of an off-card entity such as the Card Issuer, an Application Provider, or a Controlling Authority.

1.5. Abbreviations and Notations

Abbreviation / Notation	Meaning
AID	Application Identifier
APDU	Application Protocol Data Unit
API	Application Programming Interface
CLA	Class Byte of the Command Message
DGI	Data Group Identifier
GP	GlobalPlatform
IBA	IDEX Biometric Authenticator
INS	Instruction Byte of the Command Message

ISO	International Organization for Standardization
LC	Expected Length of the Command Message
LE	Maximum Length of the Response Message
P1	Reference Control Parameter 1
P2	Reference Control Parameter 2
BTC	Biometric Try Count
BTL	Biometric Try Limit
ETC	Enroll Unlock Code Try Counter
ETL	Enroll Unlock Code Try Limit
ETR	Enroll Unlock Code Try Remaining
SW	Status Word

Chapter 2.General Description

2.1. Package Information

The IBA Deliverable contains the following:

- Package:
  - Package Name: com.idex.iba.service
  - Package AID: A000000905010002
  - Package Size: 0.2kB
- Package:
  - Package Name: com.idex.iba
  - Package AID: A000000905010001
  - Package Size: 4.5kB
- Applet:
  - Applet Name: iba
  - Applet AID: A00000090501000101
  - Applet Instance Size: 0.5kB
- Files:
  - com.idex.iba.service.cap
  - com.idex.iba.cap
  - com.idex.iba.service.jar

2.2. IBA Information

The IBA Applet is a Java Card Applet designed to be used with Java Card 3.0.5 systems. The applet also requires GlobalPlatform APIs in order to handle its lifecycle.

Chapter 3.Functionalities and Lifecycle

3.1. IBA Applet Features

The IBA Applet provides three main functionalities. During the “personalization” phase it can be used to configure the enrollment options of the Biometric System. During the “enrollment” phase it can be used to enroll fingerprints in the system using an external platform.

Finally, in the “verification” phase, it can be used to perform user verification for one or more applets.

3.2. IBA Applet Lifecycle

Name	Value	Description
Selectable	0x07	The initial state for the IBA Applet after it was installed with the "Install [for Install]" command. It can be considered as the "personalization" mode. When in this state the personalization of the applet is possible. Any other commands will be rejected.
Personalized	0x0F	The next state of the IBA Applet after it has been personalized with Store Data commands. It can be considered as the "enrollment" or "verification" mode. This state can be set by sending a Set Status command to its Security Domain or using the Store Data command with P1.b8 bit set to '1' (Last Store Data). When in this state personalization commands will be rejected. Any other commands will be processed depending on the applet configuration and system state.

3.3. Security Consideration

The IBA Applet presents three critical items in terms of security. Its Enroll Unlock Code object that shall insure integrity and confidentiality of the Enroll Unlock Code data. Its lifecycle that shall be protected so command execution cannot be forced while in a wrong lifecycle. Finally in case the Enroll Unlock Code Count or the Biometric Try Count can be reset, the applet shall ensure that the command is only accepted if a valid secure channel has been established.

3.4. Interaction with other Applets

The IBA Applet needs to interact with one or more applets installed on the card, either for enrollment or for verification. Therefore, it implements two Shareable Interfaces that permit interaction through the JCRE firewall.

The "enrollment" Shareable Interface can be configured during the personalization phase to be restricted to be accessed by a single applet (identified by the applet AID).

The "verification" Shareable Interface is accessible by all applets requiring user verification services.

Chapter 4.Install and Personalization Process

4.1. Install Process

The IBA Applet can be installed using the standard GlobalPlatform Install for Install command. The IBA Applet does not currently accept any specific parameters during its installation. During the

installation the applet will create instances of the objects it requires and will register itself with the card manager. Once installed the applet will be in the “selectable” state. From this point it can be personalized following the information defined in the next section. Only one instance of this applet can be created.

4.2. Personalization Process

Before it can be used by the final user the IBA Applet shall be personalized. This personalization shall be done using DGIs passed though standard “Store Data” commands. These DGIs allows the configuration of two main items:

- The Enroll Applet Unlock Code (Value, ETL, ...)
- The Biometric System Configuration (Enrollment Options, BTL, ...)

Once the applet has been personalized its state can be updated from “SELECTABLE” to “PERSONALIZED” using the Store Data command with the bit 8 of P1 set to ‘1’ (Last Store Data). After this operation it will not be possible to personalize the applet any further for its entire lifetime. It will be possible however, to reset some configuration items as detailed in section 4.3

Note: Qualification touches mentioned in this section are performed using the “Enroll Qualification” ADPU (described in 5.11), or the verifyEnroll function over the shareable interface (described in 7.3).

The table below gives an overview of some DGIs supported by the IBA.

Tag (hex)	Length	Name	Presence
9000	8	Enroll Unlock Code Data Block	Optional
9008	1	Enroll Unlock Code Try Limit	Optional
9010	1	Finger and Touches	Optional
9011	1	Sleeve Enroll Options	Optional
9013	1	Non-sleeve Enroll Options	Optional
9015	1	Enrollment Limit	Optional
9017	1	PAD Options	Optional
9018	1	Contact WFF	Optional
9019	1	Contactless WFF	Optional
9020	1	TopUp Options	Optional
9021	1	TopUp Limit	Optional
9024	1	Matcher Mode	Optional
9100	5-16	Client Applet AID	Optional
9102	62	Select Response	Optional
9104	1	Biometric Try Count Limit	Optional
9106	1	APDU/Shareable Interface Enroll Control	Optional
Any Other Value	-	RFU	-

The tables below give the details of the different DGIs data encoding. All RFU bits should be set to zero.

Enroll Unlock Code Data Block Encoding

C	N	P	P	P	P	P/F	P/F	P/F	P/F	P/F	P/F	P/F	P/F	F	F

See “Enroll Unlock Code ” section for more information.

Enroll Unlock Code Try Limit Encoding

b8	b7	b6	b5	b4	b3	b2	b1	Meaning
-	-	-	-	X	X	X	X	ETL (Range 1-15 / Default 3)
X	X	X	X	-	-	-	-	RFU

Finger and Touches Encoding

b8	b7	b6	b5	b4	b3	b2	b1	Meaning
-	-	-	-	-	-	X	X	Number of Fingers (Range 1-2 / Default 1)
		X	X	X	X			Number of Touches (Range 1-8 / Default 6) NOTE: the total number of touches (defined by “*Number of Fingers*” * (“*Number of Touches*” + “*Number of Touches To Be Added During TopUp*”)) MUST NOT exceed 16 See “TopUp Options” below, for more details on “*Number of Touches To Be Added During TopUp*”

X	X	-	-	-	-	-	-	RFU

Sleeve Enroll Options Encoding

b8	b7	b6	b5	b4	b3	b2	b1	Meaning
-	-	X	X	X	X	X	X	RFU
-	X	-	-	-	-	-	-	Sleeve Re-Enroll Enable (Default: 0 / Disabled)
X	-	-	-	-	-	-	-	Sleeve Enroll Enable (Default: 0 / Disabled)

Non-sleeve Enroll Options Encoding

b8	b7	b6	b5	b4	b3	b2	b1	Meaning
-	-	-	-	-	X	X	X	Qualification Touch Count Limit (Range: 0-7 / Default: 0)
-	-	X	X	X	-	-	-	Qualification Touch Pass Threshold (Range: 0-7 / Default: 0)
-	X	-	-	-	-	-	-	Delete Enable (Default: 1 / Enabled)
X	-	-	-	-	-	-	-	Enroll Enable (Default: 1 / Enabled)

Enrollment Limit Encoding

b8	b7	b6	b5	b4	b3	b2	b1	Meaning
X	X	X	X	X	X	X	X	Lifetime Enroll Attempts Limit (Range: 1-255 / 255: Unlimited / Default: 255)

PAD Options

b8	b7	b6	b5	b4	b3	b2	b1	Meaning
X	X	X	X	X	X	X	X	PAD scheme (Default 0 / Disabled)

Contact WFF

b8	b7	b6	b5	b4	b3	b2	b1	Meaning
X	X	X	X	X	X	X	X	Contact WFF Timeout in ms/100 (Range: 0-255 / Default: 40=>4s)

Contactless WFF

b8	b7	b6	b5	b4	b3	b2	b1	Meaning
X	X	X	X	X	X	X	X	Contactless WFF Timeout in ms/100 (Range: 0-255 / Default: 1=>100ms)

TopUp Options

b8	b7	b6	b5	b4	b3	b2	b1	Meaning

-	-	-	-	X	X	X	X	Number of Touches To Be Added During TopUp (Range: 0-15 / Default: 10) NOTE: the total number of touches (defined by "*Number of Fingers*" * ("*Number of Touches*" + "*Number of Touches To Be Added During TopUp*")) MUST NOT exceed 16 See "Finger and Touches Encoding" above, for more details on "*Number of Fingers*" and "*Number of Touches*"
-	-	X	X	-	-	-	-	RFU
X	X	-	-	-	-	-	-	TopUp Interface Mode (Range: 0=>Contact, 1=>Contactless, 2=>Both / Default: 1)

TopUp Limit

b8	b7	b6	b5	b4	b3	b2	b1	Meaning
X	X	X	X	X	X	X	X	Max Matches Allowed During TopUp (Range: 0-255 / 255: Unlimited / Default: 255)

Matcher Mode

b8	b7	b6	b5	b4	b3	b2	b1	Meaning
X	X	X	X	X	X	X	X	Select matcher mode (Range: 1=>22RP, 2=>64RP, 255=>Use default / Default: 2)

Client Applet AID Encoding

AID[0]	AID[1]	AID[2]	...	AID[15]

Between 5 and 16 bytes of AID to identify the allowed client Applet.

Select Response Encoding

Response[0]	Response[1]	Response[2]	...	Response[61]

Up to 62 bytes to be returned in the APDU response to Selecting the IBA applet.

Biometric Try Limit

b8	b7	b6	b5	b4	b3	b2	b1	Meaning
0	X	X	X	X	X	X	X	BTL (Range 1-126 / Default 127 – no limit)

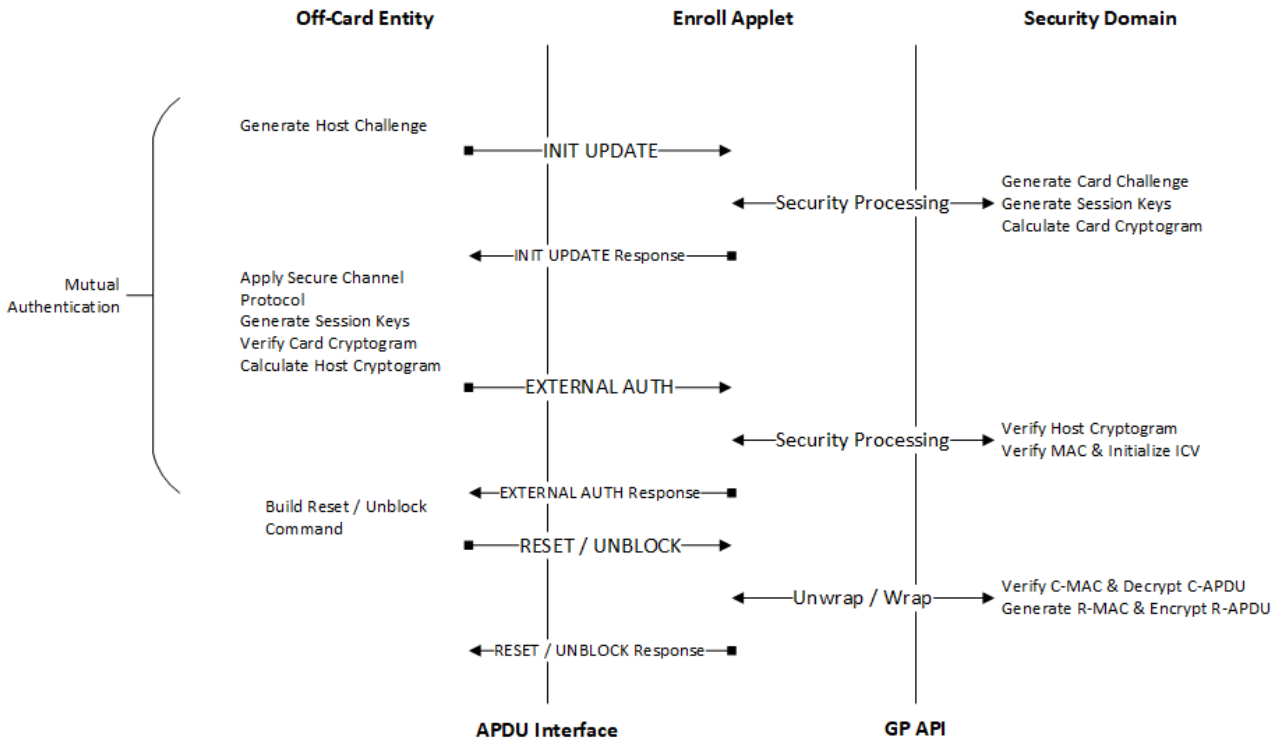
APDU/Shareable Interface Enroll Control Options Encoding

b8	b7	b6	b5	b4	b3	b2	b1	Meaning
-	-	-	-	-	-	-	X	APDU Interface Enable/Disable (Default: 1 / Enabled)
-	-	-	-	-	-	X	-	Shareable Interface Enable/Disable (Default: 0 / Disabled)
X	X	X	X	X	X	-	-	RFU

4.3. Reset Enroll Unlock Code, Lifetime Enroll Counter or Biometric Try Counter

In the case the Enroll Unlock Code becomes blocked (e.g., too many wrong codes entered), the Lifetime Enroll Count reaches the limit, or the Biometric Try Count reaches the limit, the applet provides a means to reset these.

In order to protect access to such operations it is mandatory to establish a valid Secure Channel session between the IBA Applet and the entity trying to perform the operation. This Secure Channel session can be opened using the protocol and keys available with the Security Domain owning the IBA Applet. Once the secure channel is opened the applet can accept and process the reset / unblock command defined in section 5.12. Below is a flowchart describing the reset process.



Chapter 5.APDU Interface

This chapter describes the Command-APDUs supported by the IBA Applet and their associated Response-APDUs. Depending on the applet lifecycle, the applet personalization and the system configuration some Command-APDUs may or may not be available. These Command-APDUs and their availability is summarized in the following table:

C-APDU \ APPLET STATE	SELECTABLE	PERSONALIZED
SELECT	Yes	Yes
INIT UPDATE	Yes	Yes
EXTERNAL AUTH	Yes	Yes
STORE DATA	Yes	No
VERIFY UNLOCK CODE	No	Yes (1)
GET CURRENT	Yes	Yes
GET ENROLL STATUS	No	Yes (2)
GET VERSION	Yes	Yes
GET BTC	Yes	Yes
BIST	No	Yes
SINGLE ENROLL	No	Yes (2)
ENROLL QUALIFICATION	No	Yes (2)
DELETE FINGER	No	Yes (2)
RESET UNBLOCK	No	Yes

Notes:

(1) This command is available only if the “Enroll Unlock Code” was set during the personalization

(2) In case the “Enroll Unlock Code” was set during the personalization, a successful Verify Unlock Code shall be performed before using these commands. These commands may not be available if the Applet Enrollment feature is de-activated at the Biometric System level.

5.1. IDEX Biometric Authenticator Status Words

Command-APDUs that are sent directly or via the shareable interface to the IDEX Biometric Authenticator applet may return any status word defined by IDEX. The table below gives a list of these possible status words:

Status Word	Symbol	Source
9000	IDX_ISO_SUCCESS	ALL
62A1	IDX_ISO_ERR_BAD_RMAC	Biometric System
62F1	IDX_ISO_ERR_BAD_CMAC	Biometric System

6300	IDX_ISO_ERR_NO_MATCH	Biometric System
63C0	IDX_ISO_ERR_EXT_AUTH_FAILED	Biometric System
6501	IDX_ISO_ERR_FLASH_ERASE	Biometric System
6502	IDX_ISO_ERR_FLASH_WRITE	Biometric System
6503	IDX_ISO_ERR_CRYPTO	Biometric System
6641	IDX_ISO_ERR_NO_KEYS	Biometric System
6669	IDX_ISO_ERR_NO_PADDING	ALL
6741	IDX_ISO_ERR_COMM	ALL
6743	IDX_ISO_ERR_CALIB	Biometric System
6745	IDX_ISO_ERR_TOO_SMALL	Biometric System
6746	IDX_ISO_ERR_ABORTED	Biometric System
6747	IDX_ISO_ERR_BAD_QUALITY	Biometric System
6748	IDX_ISO_ERR_TIMEOUT	Biometric System
6749	IDX_ISO_ERR_HI_TIMEOUT	Biometric System
6969	IDX_ISO_ERR_UNKNOWN	ALL
6985	IDX_ISO_ERR_CONDITIONS	ALL
6986	IDX_ISO_ERR_ILLEGAL_CMD	Biometric System
6A80	IDX_ISO_ERR_BAD_PARAMS	ALL
6A81	IDX_ISO_ERR_NOT_SUPPTD	ALL
6A83	IDX_ISO_ERR_NOT_FOUND	ALL
6A84	IDX_ISO_ERR_NO_SPACE	ALL
6A86	IDX_ISO_ERR_DATA_SIZE	ALL
6A89	IDX_ISO_ERR_HAVE_KEYS	Biometric System
6D00	IDX_ISO_ERR_BAD_COMMAND	Biometric System
6F41	IDX_ISO_ERR_MATCHER_EXTRACT_INIT	Biometric System
6F42	IDX_ISO_ERR_MATCHER_EXTRACT_PROCESS	Biometric System
6F43	IDX_ISO_ERR_INCONSISTENT	Biometric System
6F87	IDX_ISO_ERR_SNS_POWER	Biometric System
6F88	IDX_ISO_ERR_BIO_POWER	Biometric System

5.2. Select

5.2.1. Description

The “Select” Command APDU is used to perform the selection of the IBA Applet. It is processed as defined in the GlobalPlatform and Java Card specifications. No further actions are performed by the applet itself. If a “Select” APDU is sent to the applet while it is already selected nothing will happen and a “9000” status word will be returned.

5.2.2. Command APDU

CLA	INS	P1	P2	LC	DATA
00	A4	04	00	09	A00000090501000101

5.2.3. Response APDU

The “Select” command does not return any data.

Status words:

Status Word	Reason
9000	Success

5.3. Store Data

5.3.1. Description

The “Store Data” Command APDU can be used to personalize the applet after it was installed. The implementation of the command is a subset of the “Store Data” defined in the GlobalPlatform specification. The command only supports DGI formatting, any other format will be rejected. The list of supported DGIs is detailed in a previous chapter.

5.3.2. Command APDU

CLA	INS	P1	P2	LC	DATA
80	E2	08 or 88	00	XX	DGI

5.3.3. P1 Parameter

The P1 Parameter specifies the Data Format and the Store Data Status (Last Command). The Enroll Applet only supports the “DGI” format. Any other value will be rejected. The command also supports the “last command” bit that can be used to switch the applet to the “personalized mode”.

b8	b7	b6	b5	b4	b3	b2	b1	Flag Name	Meaning
1	-	-	-	-	-	-	-	Last Store Data	Set Enroll Applet to “Personalized” Mode
-	-	-	-	1	-	-	-	DGI Format	DGI Data Format
-	0	0	0	-	0	0	0	RFU	RFU (Set to 0)

5.3.4. Response APDU

The “Store Data” command does not return any data.

Status words:

Status Word	Reason
9000	Success
6A80	Incorrect Value in Command Data (e.g., Wrong DGI)
6985	Condition Not Satisfied (Personalized Mode)
6A86	Incorrect P1 / P2

5.4. Verify Enroll Unlock Code

5.4.1. Description

This command verifies that the Enroll Unlock Code sent in the APDU matches the one that was (optionally) personalized. This command shall be called before performing any enrollment operation in the case where the Enroll Unlock Code was personalized.

Note, if the Enroll Unlock Code has not been personalized then there is no requirement to perform this operation.

This command is based on the EMV VERIFY command (which checks the Transaction PIN Data sent in the APDU) as defined in the EMV Book 3 Specification (which is different to the VERIFY command used for biometric matching defined in this document), it uses the same APDU and data block definition but does not support PIN encryption.

5.4.2. Command APDU

CLA	INS	P1	P2	LC	DATA
00	20	00	80	08	Enroll Unlock Code data block

5.4.3. Enroll Unlock Code data block

A plaintext “Enroll Unlock Code data block” shall be formatted as follows:

C	N	P	P	P	P	P/F	P/F	P/F	P/F	P/F	P/F	P/F	P/F	F	F

Where:

	Name	Description
C	Control Field	4-Bits binary number with value of 0010 (Hex '2')
N	Unlock Code Length	4-Bits binary number with permissible values of 0100 to 1100 (Hex '4' to 'C')
P	Unlock Code Digit	4-Bits binary number with permissible values of 0000 to 1001 (Hex '0' to '9')
P/F	Unlock Code Digit / Filler	Depends on Unlock Code Length
F	Filler	4-Bit binary number with a value of 1111 (Hex 'F')

For example, the Enroll Unlock Code “1234” shall be formatted “241234FFFFFFFFFF”. Although the applet will not verify the formatting of the Enroll Unlock Code data block. It shall match the data that was used during the personalization of the Enroll Unlock Code. This “EMV” format is given as an example since it is a standard commonly used to store PIN blocks.

5.4.4. Response APDU

The “Verify Enroll Unlock Code” command does not return any data.

Status words:

Status Word	Reason

9000	Unlock Code Verification Successful
6700	Wrong Length
6985	Conditions Not Satisfied (Wrong Life Cycle)
6D00	'INS' Not Supported (No Enroll Unlock Code)
63CX	Unlock Code Verification Failure (X -> Tries Remaining)

5.5. Get Current

5.5.1. Description

The Get Current APDU returns the available current in uA, determined from the sensor (or SE). It is included in the functions provided by the IBA Applet with the intent that an application running on a mobile phone (or similar device) powering a payment card via its NFC field might use the information to provide a "power meter" graphical element – for example, to indicate when the card is optimally located in the NFC field.

5.5.2. Command APDU

CLA	INS	P1	P2	LC	DATA
00	59	01	00	01	00

5.5.3. Response APDU

Data	Length	Description	Presence
Current	2	The value of the current (uA) parameter returned by the sensor (or SE).	Mandatory

Status words:

Status Word	Reason
9000	Success
6XXX	Any Status Word defined in section 5.1

5.6. Get Enroll Status

5.6.1. Description

The Get Enroll Status APDU returns the enrollment status from the Biometric System. It is included in the functions provided by the IBA Applet with the intent that an application running on a mobile phone (or similar device) powering a payment card via its NFC field might need to get some information about the enrollment status of the card.

5.6.2. Command APDU

CLA	INS	P1	P2	LC	DATA
00	59	04	00	01	00

5.6.3. Response APDU

Definition for "Enrollment status response structure version" 1

Data	Length	Description	Presence
Enrollment status response structure version	1	Returns version of this response structure – this table describes version **1**	Mandatory
IDX5409 Fingerprint SensorUID	30	Fingerprint Sensor UID can be used to uniquely identify a card	Mandatory
Number of fingers	1	Total number of fingers to enroll (1 ..n)	Mandatory
The following 8 fields (from "Seed touches enrolled" untill "TopUp attempts left" inclusive) are repeated once for each finger to enroll, i.e. there are 1 .. n repetitions			
Seed touches enrolled	1	*Number of seed touches already enrolled for finger n*	Mandatory
Seed touches left to enroll	1	*Number of seed touches left to enroll for finger n*	Mandatory
Top-up touches enrolled	1	*Number of post-enrollment touches stored for finger n*	Mandatory
Qualification touch attempts left	1	Number of qualification touch attempts remaining to reach the qualification pass threshold; may be zero if qualification is not personalized (see Non-sleeve Enroll Options Encoding in 4.2)	Mandatory
Qualification passes left	1	Number of qualification passes left before the enrollment is considered successful; may be zero if qualification is not personalized (see Non-sleeve Enroll Options Encoding in 4.2)	Mandatory
Biometric mode	1	Biometric mode: 0x00 = enrollment 0x01 = qualification 0x02 = verification (with top-up) 0x03 = verification (without top-up / top-up complete) 0x04 = qualification failed (delete finger required)	Mandatory
TopUp touches left to enroll	1	Number of post-enrollment touches yet to be stored; total over all fingers; may be zero if top-up is not personalized or if all top-ups have been stored.	Mandatory
TopUp attempts left	1	Number of transactions remaining during which top-up touches may be stored.	Mandatory
Reenroll attempts left	1	Number of times an enrollment may be re-started once all seed touches are enrolled; if zero, enrollment is no longer possible; 0xFF/255 means unlimited reenroll attempts	Mandatory
Finger to enroll next	1	Finger(number) to enroll next	Mandatory
Verify mode	1	Verify mode: 0x00 = verification not possible 0x01 = verification possible	Mandatory

Status words:

Status Word	Reason
9000	Success
6XXX	Any Status Word defined in section 5.1

5.7. Get Version

5.7.1. Description

The Get Version APDU is a debug command that returns the version of the IBA applet. Command APDU

5.7.2. Command APDU

CLA	INS	P1	P2	LC
00	CA	01	00	00

5.7.3. Response APDU

Data	Length	Description	Presence
Version number	Variable	The version number in ASCII is formatted as: XXX.YYY.AAAAAA-BBB-CCCCC (these field lengths may vary)	Mandatory

Status words:

Status Word	Reason
9000	Success
6A86	Incorrect P1 / P2

5.8. Get BTC

5.8.1. Description

The Get BTC APDU is a debug command that returns the Biometric Try Count from the IBA applet. This indicates the remaining biometric attempts available before the card will be locked from any further biometric transactions.
Note: if the BTC is read before personalization, the value will read as 0x7F.

5.8.2. Command APDU

CLA	INS	P1	P2	LC
00	CA	02	00	00

5.8.3. Response APDU

Data	Length	Description	Presence
BTC	1	Biometric Try Count value	Mandatory

Status words:

Status Word	Reason
9000	Success
6A86	Incorrect P1 / P2

5.9. Sensor BIST

5.9.1. Description

The “Sensor BIST” APDU can be used to perform tests to check the status of the Sensor.

If an open/short or pixel test is requested, then the test should indicate a pass in the result flags. If a failure is detected the result flags and extended data must be shared with IDEX for RMA.

Note, this command requires that the BIST limits are passed in with the command APDU data. The length of this data is 40 bytes and the value is given below:

- BIST limits: 8100FD008B000B01F1FF1000EDFF13009700F90021002B00EFFF1300EEFF1200EFFF1E00DDFF2800

Example Sensor BIST APDUs:

- Open/Short and Pixel Test, without returning extended data: 0057450029**34**8100FD008B000B01F1FF1000EDFF13009700F90021002B00EFFF1300EEFF1200EFFF1E00DDFF2800
- Open/Short and Pixel Test, returning extended data: 0057450029**36**8100FD008B000B01F1FF1000EDFF13009700F90021002B00EFFF1300EEFF1200EFFF1E00DDFF2800

5.9.2. Command APDU

CLA	INS	P1	P2	LC	DATA
00	57	45	00	29	Flags (See description below)

BIST Flags:

b8	b7	b6	b5	b4	b3	b2	b1	Flag Name	Meaning
0	0	-	-	0	-	-	0	RFU	RFU (Set to 0)
-	-	-	-	-	-	1	-	RETURN_EXTENDED_DATA	Return extended data
-	-	-	-	-	-	0	-		Do not return extended data
-	-	-	-	-	1	-	-	LIMITS_SET_IN_CMD	BIST limits passed in with command (MUST be set to ‘1’)
-	-	-	1	-	-	-	-	EXEC_OPEN_SHORT_TEST	Perform open/short BIST
-	-	-	0	-	-	-	-		Do not perform open/short BIST

-	-	1	-	-	-	-	-	EXEC_PIXEL_TEST	Perform Pixel BIST
-	-	0	-	-	-	-	-		Do not perform Pixel BIST

5.9.3. Response APDU

Data	Length	Description	Presence
Result Flags	1	Result Flags	Mandatory
Extended Data	40	Extended Data	Conditional

BIST Result Flags:

b8	b7	b6	b5	b4	b3	b2	b1	Flag Name	Meaning
X	X	-	-	X	X	X	X	RFU	
-	-	-	1	-	-	-	-	OPEN_SHORT_TEST_RESULT	Test passed
-	-	-	0	-	-	-	-		Test failed
-	-	1	-	-	-	-	-	PIXEL_TEST_RESULT	Test passed
-	-	0	-	-	-	-	-		Test failed

Status words:

Status Word	Reason
9000	Success
6XXX	Any Status Word defined in section 5.1

5.10. Single Enroll

5.10.1. Description

Enrolls a single touch for a specified finger. The IMMEDIATE_RETURN flag provides for enrollment which will not block for an extended period of time (for example, waiting for a finger to be placed on the sensor) and is intended for use where an application running on a mobile phone or enrollment terminal is controlling the enrollment process (where the GUI needs to be continually updated, so a blocking call is unacceptable).

5.10.2. Command APDU

CLA	INS	P1	P2	LC	DATA
00	59	03	00	02	Single Enroll Data

Single Enroll Data:

Data	Length	Description	Presence
Flags	1	See Description Below	Mandatory
Finger	1	Finger Index (Minimum Value: 1)	Mandatory

Single Enroll Flags:

b8	b7	b6	b5	b4	b3	b2	b1	Flag Name	Meaning
-	-	-	-	1	-	-	-	EXTENDED_DATA	Return extended data
-	-	-	-	0	-	-	-		Do not return extended data

-	-	-	1	-	-	-	-	IMMEDIATE_RETURN	Return immediately if no finger is on the sensor, else continue as specified by IR_NOT_ACQUIRE bit
-	-	-	0	-	-	-	-		Wait for finger to be placed before acquiring fingerprint
-	-	1	-	-	-	-	-	IR_NOT_ACQUIRE	Return without acquiring a fingerprint. Indicates a finger is present by a return status of 0x9000 with touch=0x00 in response data
-	-	0	-	-	-	-	-		Acquire a fingerprint
0	0	-	-	-	0	0	0	RFU	RFU (Set to 0)

5.10.3. Response APDU

Data	Length	Description	Presence
Touch	1	Indicates which touch has been enrolled (in the fingerprint database). Counts from one.	Mandatory
Extended Data	24	Only present if the EXTENDED_DATA bit in the Flags byte is one. Extended response data used internally by IDEX for verification purposes.	Conditional

Status words:

Status Word	Reason
9000	Success
6XXX	Any Status Word defined in section 5.1

5.11. Enroll Qualification

5.11.1. Description

Performs a “qualification” biometric verification against stored templates after a finger has been enrolled. The purpose of including this function in the IBA Applet is to allow for a controlling enrollment application (running on a mobile phone or an enrollment terminal) to do a “trial” verification before the card is switched from “enrollment” to “verification” mode. Therefore, this function will attempt a match when the card is in “enrollment” mode.

5.11.2. Command APDU

CLA	INS	P1	P2	LC	DATA
00	59	00	00	01	Enroll Qualification Flags (See Table Below)

Enroll Qualification Flags:

b8	b7	b6	b5	b4	b3	b2	b1	Flag Name	Meaning
-	-	-	-	-	-	1	-	TIMESTAMP	Return timestamp data
-	-	-	-	-	-	0	-		Do not return timestamp data
-	-	-	-	-	1	-	-	EXTENDED_DATA	Return extended response data

-	-	-	-	-	0	-	-		Do not return extended response data
-	-	-	1	-	-	-	-	IMMEDIATE_RETURN	Return immediately if no finger is on the sensor, else continue as specified by IR_NOT_ACQUIRE bit
-	-	-	0	-	-	-	-		Wait for finger to be placed before acquiring fingerprint
-	-	1	-	-	-	-	-	IR_NOT_ACQUIRE	Return without acquiring a fingerprint. Indicates a finger is present by a return status of 0x9000 with no response data
-	-	0	-	-	-	-	-		Acquire a fingerprint
0	0	-	-	0	-	-	0	RFU	RFU (Set to 0)

5.11.3. Response APDU

Data	Length	Description	Presence
Touch Index	1	When the match results in IDX_ISO_SUCCESS, these fields indicate touch and finger, respectively, of the matching template. Both count from one.	Mandatory
Finger Index	1		Mandatory
Match Signature	8	This field gives a "signature" that reflects the internal operation of the matcher. It is used internally by IDEX for verification purposes.	Mandatory
Timestamps	38 or 39	Only present if the TIMESTAMP bit in flags is set. If only one finger is to be enrolled then length is 38, if two fingers are to be enrolled then length is 39.	Conditional
Extended Data	52	Only present if the EXTENDED_DATA bit in flags is set. Used internally by IDEX for verification purposes.	Conditional

Status words:

Status Word	Reason
9000	Success
6XXX	Any Status Word defined in section 5.1

5.12. Delete Finger

5.12.1. Description

Used to remove the templates for a single finger ***or*** all existing templates currently enrolled. Once this command has been successfully executed, if no templates exist then it is no longer possible to perform biometric user verification until re-enrollment has been completed. If a single finger is removed but another finger remains, then it will still be possible to perform biometric user verification using the remaining finger.

5.12.2. Command APDU

CLA	INS	P1	P2	LC	DATA
00	59	02	00	01	Finger to delete (See Table Below)

Finger to delete options:

Value	Meaning
00	Remove all templates
01	Remove F1 template
02	Remove F2 template

5.12.3. Response APDU

The “Delete Finger” command does not return any data.

Status words:

Status Word	Reason
6A80	Incorrect data
6XXX	Any Status Word defined in section 5.1

5.13. Reset /Unblock

5.13.1. Description

The “Reset Unblock” Command APDU can be used to reset Lifetime Enroll counter, the Biometric Try counter or the Enroll Unlock Code try counter and therefore unblock the card. A secure channel shall be established before sending this command to the applet. If not, the command will be rejected. The command optionally accepts as input data a list of tags that defines the values to be reset.

5.13.2. Command APDU

CLA	INS	P1	P2	LC	DATA
00	21	00	00	XX	List of Tags

List of Tags Options:

Tag	Description
9009	Enroll Unlock Code Count Reset
9016	Lifetime Enrollment Count Reset
9105	Biometric Try Count Reset

5.13.3. Response APDU

The “Reset / Unblock” command does not return any data.

Status words:

Status Word	Reason
9000	Success
6A80	Incorrect Value in Command Data
6985	Condition Not Satisfied (Not Personalized)

6A86	Incorrect P1 / P2

Chapter 6.Shareable Interface Overview

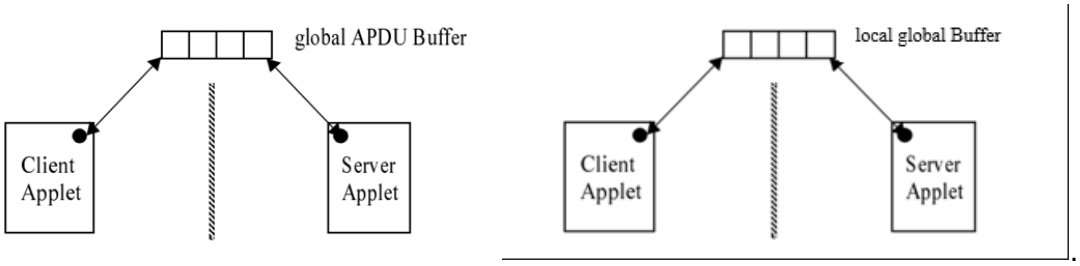
The JCRE enforces a strict firewall between applets to prevent one applet from accessing data belonging to another applet.

Shareable interfaces allow applets to explicitly share data by defining a set of shared interface methods. The applets that provide these shared interface methods are considered server applets (in this case the IBA Applet) and the applets that make use of these shared interface methods are considered client applets.

A server applet can make its shared methods available to all other applets, or it can restrict access to certain other applets based on the client applet AIDs.

The IBA applet implements two shareable interfaces, one for verification which is accessible by all client applets that wish to make use of the biometric verification service offered by the IBA, and one for enrollment which may (optionally) be restricted to a single client applet.

Communication between the client and server applets can be performed using a global APDU buffer or a locally allocated global buffer as shown below:



6.1. Using the Shareable Interfaces

6.1.1. Enrollment Shareable Interface Configuration

In order to restrict the enrollment shareable interface to a single client applet, the AID of the nominated client applet needs to be personalised using the “Store Data” command with a DGI of 9100 and a value length of between 5 and 16 bytes.

Once the AID is configured any other client applet will receive a null response.

In order to avoid issues if the verify shareable interface is cast to the enrollment shareable interface by a malicious client, the AID of the client applet is also checked on each access to an enrollment shareable interface API access.

If this DGI is not configured then any client applet will be able to use the enrollment shareable interface.

6.1.2. Accessing the Shareable Interfaces from a Client Applet

The IBA AID can be retrieved as follows:

```
//  IBA Applet AID bytes
private   final byte[] ibaAIDBytes = {(byte)0xA0,(byte)0x00,(byte)0x00,(byte)0x09,
   (byte)0x05, (byte)0x01,(byte)0x00,(byte)0x01,(byte)0x01};
private AID ibaAID;
ibaAID = JCSystem.lookupAID(ibaAIDBytes, (short)0x00, (byte)ibaAIDBytes.length);
```

The Verify Shareable Interface is accessed by specifying parameter = (byte)0 when getting the IBA Applet shareable interface object:

```
SIOvfy = (verifyInterface)JCSystem.getAppletShareableInterfaceObject(ibaAID, (byte)0);
```

The Enrollment Shareable Interface is accessed by specifying parameter = (byte)1 when getting the IBA Applet shareable interface object:

```
SIOen = (enrollInterface)JCSystem.getAppletShareableInterfaceObject(ibaAID, (byte)1);
```

6.1.3. General structure of the shareable interface APIs

Input parameters (where present to the APIs are passed explicitly in named parameters.

Output data is placed in to a global buffer (either the global APDU buffer or a locally allocated global buffer) specified by the ba_OutBuffer parameter. It will reside at the offset specified in the s_OutOffset parameter of the API.

On success the return code is always a short indicating the length of the output data.

If there is an error, the return code is always -1 and the data in the output data (at the specified offset) is a 2 byte error code.

6.1.4. Using the global APDU buffer in the client

In this example code, the global APDU Buffer contains the parameters for the call which are copied to the input parameters, and on success the response data is returned from the API in the global APDU buffer already aligned for return to the host

```
   byte[] apduBuf = apdu.getBuffer();
   byte flag = apduBuf[ISO7816.OFFSET_CDATA];
   byte fingerIndex = apduBuf[ISO7816.OFFSET_CDATA+1];
   sRspLen =((enrollInterface) SIO).singleEnroll(fingerIndex, flag, apduBuf,
(short)ISO7816.OFFSET_CDATA);
   if (sRspLen == (short)-1)
      ISOException.throwIt(Util.makeShort(apduBuf[ISO7816.OFFSET_CDATA],
apduBuf[ISO7816.OFFSET_CDATA +1]));
```

6.1.5. Using a locally allocated global buffer in the client

In this example code, a local global buffer is allocated. The global APDU buffer contains the parameters for the call which are copied to the input parameters, and on success the response data is returned from the API in the local global buffer. In this case the response data needs to be copied to the global APDU buffer for return to the host.

Note: the size of the locally allocated global buffer must be sized to ensure that it cannot overflow, taking into account the worst case response data length + offset.

```
   Object oArray = JCSystem.makeGlobalArray(JCSystem.ARRAY_TYPE_BYTE, (short)0x80);
   byte[] baOutBuf = (byte [])oArray;
   byte flag = apduBuf[ISO7816.OFFSET_CDATA];
   byte fingerIndex = apduBuf[ISO7816.OFFSET_CDATA+1];
   sRspLen =((enrollInterface) SIO).singleEnroll(fingerIndex, flag, baOutBuf, (short)0
   if (sRspLen == (short)-1)
      ISOException.throwIt(Util.makeShort(baOutBuf[0], baOutBuf[1]));
   Util.arrayCopyNonAtomic(baOutBuf, (short)0, apduBuf, (short)ISO7816.OFFSET_CDATA,
sRspLen);
```

Chapter 7.Enrollment Shareable Interfaces

API \ APPLET STATE	SELECTABLE	PERSONALIZED
GET ENROLL STATUS	No	Yes
SINGLE ENROLL	No	Yes
ENROLL QUALIFICATION	No	Yes
DELETE TEMPLATES	No	Yes
DELETE FINGER	No	Yes

7.1. Get Enroll Status

```
short getEnrollStatus(byte[] ba_OutBuffer, short s_OutOffset);
```

7.1.1. Description

Returns the enrollment status from the Biometric System. It is included in the functions provided by the IBA Applet with the intent that an application running on a mobile phone (or similar device) powering a payment card via its NFC field might need to get some information about the enrollment status of the card.

7.1.2. Parameters

`ba_OutBuffer: global buffer which the IBA will use to return the response data`

`s_OutOffset: offset in the global buffer at which the IBA will place the response data`

7.1.3. Return Value

`Success: see section 5.6 for details of response data structure and the length returned on success`

`Failure: -1`

Failure status words:

Status Word	Reason
6XXX	Any Status Word defined in section 5.1

7.2. Single Enroll

```
short singleEnroll(byte b_FingerIndex, byte b_Flags, byte[]
ba_OutBuffer, short s_OutOffset);
```

7.2.1. Description

Enrolls a single touch for a specified finger. The IMMEDIATE_RETURN flag provides for enrollment which will not block for an extended period of time (for example, waiting for a finger to be placed on the sensor) and is intended for use where an application running on a mobile phone or enrollment terminal is controlling the enrollment process (where the GUI needs to be continually updated, so a blocking call is unacceptable).

7.2.2. Parameters

`b_FingerIndex: finger which is being enrolled, see below for options`

`b_Flags: see section 5.8 for details of the available flags`

`ba_OutBuffer: global buffer which the IBA will use to return the response data`

`s_OutOffset: offset in the global buffer at which the IBA will place the response data`

b_FingerIndex options:

Value	Meaning
01	Remove F1 template
02	Remove F2 template

7.2.3. Return Value

`Success: see section 5.8 for details of response data structure and the length returned on success`

`Failure: -1`

Failure status words:

Status Word	Reason
6XXX	Any Status Word defined in section 5.1

7.3. Enroll Qualification

```
short verifyEnroll(byte b_Flags, byte[] ba_OutBuffer, short
s_OutOffset);
```

7.3.1. Description

Performs a “qualification” biometric verification against stored templates after a finger has been enrolled. The purpose of including this verify function in the IBA Applet is to allow for a controlling enrollment application (running on a mobile phone or an enrollment terminal) to do a “trial” verification before the card is switched from “enrollment” to “verification” mode. Therefore, this function will attempt a match when the card is in “enrollment” mode.

Note: the function name 'verifyEnroll' is a legacy naming and subject to change in a future release.

7.3.2. Parameters

`b_Flags: see section 5.9 for details of the available flags`

`ba_OutBuffer: global buffer which the IBA will use to return the response data`

`s_OutOffset: offset in the global buffer at which the IBA will place the response data`

7.3.3. Return Value

`Success: see section 5.9 for details of response data structure and the length returned on success`

`Failure: -1`

Failure status words:

Status Word	Reason
6XXX	Any Status Word defined in section 5.1

7.4. Delete Templates

```
short deleteTemplates(byte[] ba_OutBuffer, short s_OutOffset);
```

7.4.1. Description

Used to remove all existing templates currently enrolled. Once this command has been successfully executed it is no longer possible to perform biometric user verification, until re-enrollment has been completed.

Note: it is the responsibility of the client applet to ensure the security surrounding the deletion and subsequent re-enrollment process.

7.4.2. Parameters

`ba_OutBuffer: global buffer which the IBA will use to return the response data`

`s_OutOffset: offset in the global buffer at which the IBA will place the response data`

7.4.3. Return Value

`Success: 0`

`Failure: -1`

Failure status words:

Status Word	Reason
6986	Illegal command
6985	Condition not satisfied
6700	Incorrect command length
9B02	Biometric MCU does not allow the usage of this method

9B03	Biometric MCU does not support the usage of this method
9B07	Biometric MCU cannot perform the operation due to an internal error

7.5. Delete Finger

```
short deleteFinger(byte FingerIndex, byte[] ba_OutBuffer, short
s_OutOffset);
```

7.5.1. Description

Used to remove the templates for a single finger, ***or*** all existing templates currently enrolled. Once this command has been successfully executed if all no templates exist then it is no longer possible to perform biometric user verification, until re-enrollment has been completed. If a single finger is removed but another finger remains, then it will still be possible to perform biometric user verification using the remaining finger.

Note: it is the responsibility of the client applet to ensure the security surrounding the deletion and subsequent re-enrollment process.

7.5.2. Parameters

`b_FingerIndex: see section 5.10 for details of the available values`

`ba_OutBuffer: global buffer which the IBA will use to return the response data`

`s_OutOffset: offset in the global buffer at which the IBA will place the response data`

7.5.3. Return Value

`Success: see section 5.10 for details of response data structure and the length returned on success`

`Failure: -1`

Failure status words:

Status Word	Reason
6A80	Incorrect data
6XXX	Any Status Word defined in section 5.1

Chapter 8.Verification Shareable Interfaces

API \ APPLET STATE	SELECTABLE	PERSONALIZED
GET CURRENT	No	Yes
VERIFY	No	No
CHECK VERIFY RESULT	No	Yes

8.1. Get Current

```
short getCurrent(byte[] ba_OutBuffer, short s_OutOffset);
```

8.1.1. Description

Use to retrieve the available current in uA, determined from the sensor (or SE). It is included in the functions provided by the IBA Applet with the intent that an application running on a mobile phone (or similar device) powering a payment card via its NFC field might use the information to provide a “power meter” graphical element – for example, to indicate when the card is optimally located in the NFC field.

8.1.2. Parameters

`ba_OutBuffer: global buffer which the IBA will use to return the response data`

`s_OutOffset: offset in the global buffer at which the IBA will place the response data`

8.1.3. Return Value

`Success: see section 5.5 for details of response data structure and the length returned on success`

`Failure: -1`

Failure status words:

Status Word	Reason
6XXX	Any Status Word defined in section 5.1

8.2. Verify

```
short verify(byte[] ba_OutBuffer, short s_OutOffset);
```

8.2.1. Description

Used to request biometric user verification be performed. The response data indicates whether there was a match or not by returning a response state.

In case the biometric user verification fails due to not finding a match (i.e. a “no match” occurs) then the BTC will be incremented (until it reaches the BTL limit, at which point no further user verification will be possible), or it may be reset to zero if non-zero when a user verification is successful. All other results (i.e. anything other than “OK” or “no match”) will result in the BTC remaining at the previous value.

8.2.2. Parameters

`ba_OutBuffer: global buffer which the IBA will use to return the response data`

`s_OutOffset: offset in the global buffer at which the IBA will place the response data`

8.2.3. Return Value

`Success: 1`

Response state:

State	Meaning	Value
BIO_LAST_RESULT_OK	The last verification was a “match”	0xCA
BIO_LAST_RESULT_NO_MATCH	The last verification was a “no match”	0x35
BIO_LAST_RESULT_NO_TEMPLATE	There are no enrolled fingers	0x02
BIO_LAST_RESULT_BTC_EXCEEDED	The biometric try count has exceeded the limit and no verification was attempted	0x09
BIO_LAST_RESULT_TIMEOUT	No finger was detected within the timeout period	0x06
BIO_LAST_RESULT_UNKNOWN_ERROR	An unspecified error occurred	0xFF

`Failure: -1`

Failure status words:

Status Word	Reason
6986	Illegal command
6985	Condition not satisfied
6700	Incorrect command length
6A80	Incorrect command data
6581	Memory corruption

8.3. Check Verify Result

```
byte checkVerifyResult(byte[] ba_OutBuffer, short s_OutOffset);
```

8.3.1. Description

Used to retrieve the outcome of the last “Verify” command from the Biometric System. This is useful for providing an additional check on the outcome of a “Verify” command where the result is “match”, thus providing an extra level of security by avoiding a single point of attack.

8.3.2. Parameters

`ba_OutBuffer: global buffer which the IBA will use to return the response data`

`s_OutOffset: offset in the global buffer at which the IBA will place the response data`

8.3.3. Return Value

`Success: 1`

Response state:

State	Meaning	Value
BIO_LAST_RESULT_OK	The last verification was a “match”	0xCA
BIO_LAST_RESULT_NO_MATCH	The last verification was a “no match”	0x35
BIO_LAST_RESULT_NO_TEMPLATE	There are no enrolled fingers	0x02
BIO_LAST_RESULT_BTC_EXCEEDED	The biometric try count has exceeded the limit and no verification was attempted	0x09
BIO_LAST_RESULT_TIMEOUT	No finger was detected within the timeout period	0x06
BIO_LAST_RESULT_UNKNOWN_ERROR	An unspecified error occurred	0xFF

`Failure: -1`

Failure status words:

Status Word	Reason
6985	Condition not satisfied
6700	Incorrect command length
6A80	Incorrect command data

Contact Us

HEAD OFFICE

IDEX BIOMETRICS ASA
Dronning Eufemias gate 16, NO-0191
Oslo
Norway
Tel: +47 6783 9119
Email:
mailbox@idexbiometrics.com

For sales enquiries:
sales@idexbiometrics.com

For IR/PR enquiries:
ir@idexbiometrics.com

IDEX USA

IDEX America Inc., Wilmington
187 Ballardvale Street
Suite A-260
Wilmington, MA 01887

IDEX America Inc., Rochester
4545 E River Road
Suite 200
West Henrietta, NY 14586

IDEX EUROPE

IDEX Biometrics UK Ltd,
Abbey House
282 Farnborough Road
Farnborough GU14 7NA
United Kingdom