INTERNATIONAL STANDARD

ISO/IEC 19784-1

First edition
2006-05-01

AMENDMENT 1
2007-12-01

Information technology — Biometric application programming interface

Part 1: BioAPI specification

AMENDMENT 1: BioGUI specification

Technologies de l'information — Interface de programmation d'applications biométriques

Partie 1: Spécifications BioAPI

AMENDEMENT 1: Spécifications BioGUI

Reference number
ISO/IEC 19784-1:2006/Amd.1:2007(E)

ISO copyright office
Case postale 56 • CH-1211 Geneva 20
Tel. + 41 22 749 01 11
Fax + 41 22 749 09 47
E-mail copyright@iso.org
Web www.iso.org

Published in Switzerland

Foreword

ISO (the International Organization for Standardization) and IEC (the International Electrotechnical Commission) form the specialized system for worldwide standardization. National bodies that are members of ISO or IEC participate in the development of International Standards through technical committees established by the respective organization to deal with particular fields of technical activity. ISO and IEC technical committees collaborate in fields of mutual interest. Other international organizations, governmental and non-governmental, in liaison with ISO and IEC, also take part in the work. In the field of information technology, ISO and IEC have established a joint technical committee, ISO/IEC JTC 1.

International Standards are drafted in accordance with the rules given in the ISO/IEC Directives, Part 2.

The main task of the joint technical committee is to prepare International Standards. Draft International Standards adopted by the joint technical committee are circulated to national bodies for voting. Publication as an International Standard requires approval by at least 75 % of the national bodies casting a vote.

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. ISO and IEC shall not be held responsible for identifying any or all such patent rights.

Amendment 1 to ISO/IEC 19784-1 was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information technology*, Subcommittee SC 37, *Biometrics*.

It defines a new version of BioAPI (BioAPI 2.1) that:

a) extends and improves the "application-controlled GUI" feature of BioAPI;

b) produces alignment between BioAPI and ISO/IEC 19785-1:2006, *Information technology — Common Biometric Exchange Formats Framework — Data element specification*

c) provides for other standards to modify BioAPI framework behaviour to support the use of biometric service providers (BSPs) that are remote from the controlling applications.

Introduction

With this amendment, ISO/IEC 19784-1 specifies two versions of BioAPI: 2.0 and 2.1. All the provisions that apply only to one version of BioAPI (either 2.0 or 2.1) are labeled as such.

The main difference of BioAPI 2.1 from BioAPI 2.0 is the support it provides for BioGUI. BioGUI stands for "BioAPI Graphical User Interface". The functionality specified in this amendment enables an application to control the display of graphics at enrollment, verification and identification as an alternative to using the graphical user interface provided by BSPs.

Secondly, BioAPI 2.1 also aligns the values of the type definition BioAPI_BIR_BIOMETRIC_TYPE with those specified in ISO/IEC 19785-1:2006.

Thirdly, it provides additional functions and parameters that are redundant in a purely local implementation of BIoAPI, but which enable other standards (interworking standards - see 4.29) to modify the behaviour of a BioAPI framework to support interactions between an application and remote BSPs.

Finally, it provides improvements to some of the functions and parameters defined for BioAPI 2.0, particularly in relation to support for tenprint capture, the electronic capture of ten human fingerprints.

This amendment redefines (in the specificaton of BioAPI 2.1) portions of the BioAPI 2.0 specification that define BioAPI types, macros, functions and callback functions (particularly those related to the application-controlled GUI feature) with a new set of definitions that provide more functionality. Some of the old BioAPI 2.0 definitions are completely replaced by new definitions (with the same or with different names), while others are extended by the addition of one or more parameters. Some types and functions are entirely new in BioAPI 2.1.

The resulting specification is expected to better meet the needs of biometric applications that wish to have full control of the user interface during enrollment, verification and identification, that need to be able to work with remote BSPs, or that need added functionality for interaction with local BSPs.

Information technology — Biometric application programming interface

Part 1:
BioAPI specification

AMENDMENT 1: BioGUI specification

1) General amendment items

1-1) Replace the last paragraph of the Foreword with the following:

The first edition of this part of ISO/IEC 19784 was the first ISO/IEC standard on BioAPI. Previous versions of the specification were published by ANSI and the BioAPI Consortium. As the last official non-ISO specification was designated BioAPI 1.1, the versions specified in this part of ISO/IEC 19784 are designated BioAPI 2.0 and BioAPI 2.1.

1-2) Replace the first paragraph of the Introduction with:

This part of ISO/IEC 19784 provides a high-level generic biometric authentication model suited to most forms of biometric technology. No explicit support for multimodal biometrics is provided.

1-3) Insert the following paragraph after the third paragraph of the Introduction

This part of ISO/IEC 19784 specifies the behaviour of the BioAPI Framework when applications and BSPs are in the same system. Other interworking standards (see 4.29) specify modifications of that behaviour that enable both BSPs and Graphical User Interfaces to be remote from the system containing an application.

NOTE: ISO/IEC 24708 specifying the BioAPI Interworking Protocol (BIP) [6] is an example of an interworking standard.

2) Amendment items for the application-controlled GUI

2-1) Add the following text after 4.28:

4.29
interworking standards
standards that modify the behaviour of the BioAPI Framework (and BioAPI conformance requirements) to support the use of communications links to enable an application to interact with a remote BSP using a standardised protocol

4.30
test-verify
test-verification
one-to-one process of comparing a single biometric sample against a candidate of a biometric reference template at enrollment to determine whether the candidate template matches the test sample

4.31
enroll type
value denoting a pattern of suboperations performed by a BSP during an enroll operation

2-2) Add the following text after 6.1.8 d), after replacing "." at the end of c) with "; and":

e) providing graphical information to the application about an ongoing enrollment, verification, or identification operation.

2-3) Replace the heading of subclause 7.8 with the following text:

7.8 BioAPI_BIR_BIOMETRIC_TYPE (BioAPI 2.0)

This subclause applies only when the BioAPI version number in use is 2.0.

2-4) Add the following text after subclause 7.57 (BioAPI_VERSION):

7.58 BioAPI_BIR_BIOMETRIC_TYPE (BioAPI 2.1)

This subclause applies only when the BioAPI version number in use is 2.1.

A mask that describes the set of biometric types (factors) contained within a BioAPI BIR or supported by a BSP.

```
typedef uint32_t BioAPI_BIR_BIOMETRIC_TYPE;

    #define BioAPI_NO_BIOTYPE_AVAILABLE          (0x00000000)
    #define BioAPI_TYPE_MULTIPLE_BIOMETRIC_TYPES (0x00000001)
    #define BioAPI_TYPE_FACE                     (0x00000002)
    #define BioAPI_TYPE_VOICE                    (0x00000004)
    #define BioAPI_TYPE_FINGER                   (0x00000008)
    #define BioAPI_TYPE_IRIS                     (0x00000010)
    #define BioAPI_TYPE_RETINA                   (0x00000020)
    #define BioAPI_TYPE_HAND_GEOMETRY            (0x00000040)
    #define BioAPI_TYPE_SIGNATURE_SIGN           (0x00000080)
    #define BioAPI_TYPE_KEYSTROKE                (0x00000100)
    #define BioAPI_TYPE_LIP_MOVEMENT             (0x00000200)
    #define BioAPI_TYPE_GAIT                     (0x00001000)
    #define BioAPI_TYPE_VEIN                     (0x00002000)
    #define BioAPI_TYPE_DNA                      (0x00004000)
    #define BioAPI_TYPE_EAR                      (0x00008000)
    #define BioAPI_TYPE_FOOT                     (0x00010000)
    #define BioAPI_TYPE_SCENT                    (0x00020000)
    #define BioAPI_TYPE_OTHER                    (0x40000000)
    #define BioAPI_TYPE_PASSWORD                 (0x80000000)
```

NOTE 1: BioAPI_TYPE_MULTIPLE_BIOMETRIC_TYPES is used to indicate that the biometric samples contained within the BDB (BIR BiometricData) include biometric samples from more than one type of biometric sensor unit (e.g.,

fingerprint and facial data). Location of the individual samples within the BDB is specified by the Format Owner and identified by the value of the Format Type.

NOTE 2: The condition NO VALUE AVAILABLE is indicated by setting the value to zero. BIRs that are not originally created by BioAPI BSPs should use this value when transformed into a BioAPI BIR if Biometric Type information is not available in the original source record. Transformed BIRs whose biometric type does not correspond to one of the defined types shall use the value for OTHER.

NOTE 3: The BioAPI BIR Biometric Type corresponds to the “CBEFF_BDB_biometric_type” in ISO/IEC 19785-1.

NOTE 4: Although “password” is not a biometric characteristic, BioAPI_TYPE_PASSWORD is included as a valid BioAPI_BIR_BIOMETRIC_TYPE to support its use a) for development & test environments, and b) as an additional authentication factor within a BioAPI application.

NOTE 5: The names of several biometric types are different in BioAPI 2.1 from the names used in BioAPI 2.0, with no change in semantics, and several new biometric types are present in BioAPI 2.1. Finally, the "thermal" types are present in BioAPI 2.0 but are absent from BioAPI 2.1. These differences are in order to align with ISO/IEC 19785-1.

NOTE 6: The names of the following values differ between BioAPI 2.0 and BioAPI 2.1, but their semantics is the same. The BioAPI 2.1 names are aligned with ISO/IEC 19785-1 (CBEFF).

Name in BioAPI 2.0	Name in BioAPI 2.1	Encoding
MULTIPLE	MULTIPLE_BIOMETRIC_TYPES	0x00000001
FACIAL_FEATURES	FACE	0x00000002
FINGERPRINT	FINGER	0x00000008
SIGNATURE_DYNAMICS	SIGNATURE_SIGN	0x00000080
KEYSTROKE_DYNAMICS	KEYSTROKE	0x00000100

NOTE 7: The following values are not present BioAPI 2.0 but are present in BioAPI 2.1 for consistency with ISO/IEC 19785-1 (CBEFF).

Added value	Encoding
NO_BIOTYPE_AVAILABLE	0x00000000
VEIN	0x00002000
DNA	0x00004000
EAR	0x00008000
FOOT	0x00010000
SCENT	0x00020000

2-5) Replace the heading of subclause 7.14 with the following text:

7.14 BioAPI_BIR_SUBTYPE (BioAPI 2.0)

This subclause applies only when the BioAPI version number in use is 2.0.

2-6) Add the following text after 7.58:

7.59 BioAPI_BIR_SUBTYPE (BioAPI 2.1)

7.59.1 This subclause applies only when the BioAPI version number in use is 2.1.

7.59.2 This identifies a subtype within the BDB type (specified in the BioAPI_BIR_BIOMETRIC_TYPE). Its values and their meaning are defined by the specifier of that BDB type.

NOTE: In the BioAPI 2.1 definition of this type, multiple bits can be set. This definition is aligned with ISO/IEC 19785-1.

```
typedef uint8_t BioAPI_BIR_SUBTYPE;

    #define BioAPI_BIR_SUBTYPE_VEIN_ONLY_MASK        (0x80)
    #define BioAPI_BIR_SUBTYPE_LEFT_MASK             (0x01)
    #define BioAPI_BIR_SUBTYPE_RIGHT_MASK            (0x02)

    #define BioAPI_BIR_SUBTYPE_THUMB                 (0x04)
    #define BioAPI_BIR_SUBTYPE_POINTERFINGER         (0x08)
    #define BioAPI_BIR_SUBTYPE_MIDDLEFINGER          (0x10)
    #define BioAPI_BIR_SUBTYPE_RINGFINGER        (0x20)
    #define BioAPI_BIR_SUBTYPE_LITTLEFINGER      (0x40)

    #define BioAPI_BIR_SUBTYPE_VEIN_PALM         (0x04)
    #define BioAPI_BIR_SUBTYPE_VEIN_BACKOFHAND   (0x08)
    #define BioAPI_BIR_SUBTYPE_VEIN_WRIST        (0x10)

    #define BioAPI_NO_SUBTYPE_AVAILABLE          (0x00)
```

7.59.3 This structure is a bitmask, with the bits defined as shown below. Bit 7 is used to indicate the interpretation of the lower-order bits. Bit positions zero (0) and one (1) always indicate left and right respectively. When bit position 7 is not set, these bit positions may apply to any biometric type; however, bit positions 2-6 are specific to the finger and vein biometric types. When bit position 7 is set, then the remaining bit positions apply to the vein biometric type only.

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0 (Any)	Little	Ring	Middle	Pointer	Thumb	Right	Left
1 (Vein only)	Reserved	Reserved	Wrist	Back of Hand	Palm	Right	Left

NOTE: Vein biometric data may be obtained from the fingers (Bit 7 set to zero), or from the wrists, palm, or backs of either hand (Bit 7 set to 1).

7.59.4 Zero or more bits may be set.The abstract value NO VALUE AVAILABLE is indicated by setting all bits to zero.

NOTE The abstract value NO VALUE AVAILABLE can be used with BIRs that are not originally created by a BioAPI BSP but have been transformed from another data format. It can also be used when none of the other values are applicable or information is not available.

7.59.5 The bit positions BioAPI_BIR_SUBTYPE_THUMB through BioAPI_BIR_SUBTYPE_LITTLEFINGER can be used to identify the instance(s) of a biometric type related to the fingers.

NOTE: The BioAPI BIR Subtype corresponds to the “CBEFF_BDB_biometric_subtype” in ISO/IEC 19785-1.

7.59.6 If one or more of the bit positions BioAPI_BIR_SUBTYPE_THUMB through BioAPI_BIR_SUBTYPE_LITTLEFINGER are set, then either or both the bit positions BioAPI_BIR_SUBTYPE_LEFT_MASK and BioAPI_BIR_SUBTYPE_RIGHT_MASK shall also be set. When only the former is set, the subtype value designates one or more fingers of the left hand. When only the latter is set, the subtype value designates one or more fingers of the right hand. When both are set, the value designates one or more fingers of both hands (the same finger locations in both).

NOTE: It is not possible to designate, for example, a set of fingers consisting of the index finger of the left hand and the medium finger of the right hand.

7.59.7 When a value with multiple finger bits set occurs in a BIR header (or is used as an input parameter of a BioAPI function that performs capture), it is not strictly required that all the specified finger instances be actually present in the BDB of the BIR (or be actually captured). However, it is recommended that all the designated finger instances be present in the BDB except in the case of missing fingers and similar exceptional situations. Likewise, if the subtype field of a BIR designates multiple fingers, it is acceptable for the BDB of that BIR to include data about an extra finger if the subject happens to have six fingers in his or her hand.

7.59.8 If none of the finger bit positions BioAPI_BIR_SUBTYPE_THUMB through BioAPI_BIR_SUBTYPE_LITTLEFINGER are set, then the bit positions BioAPI_BIR_SUBTYPE_LEFT_MASK and BioAPI_BIR_SUBTYPE_RIGHT_MASK can be used to identify an instance of a biometric type for which there is one left instance and one right instance (for example, iris), or also the single instance of a biometric type for which there is only one instance (for example, face).

7.59.9 Vein data taken from one or more fingers is indicated by setting bit position 7 to zero, setting one or both of bit positions 0 and 1, and setting one or more of the finger bit positions (3-7). Vein data taken from one of the other sources (wrist, palm, or back of hand) is indicated by setting bit position 7 to one, setting one or both of bit positions 0 and 1, and setting one of bit positions 3-5 as appropriate).

NOTE 1: BIRs that are not originally created by a BioAPI BSP but have been transformed from another data format and for which subtype information is not available may use the NO VALUE AVAILABLE condition.

NOTE 2: The BioAPI BIR Subtype corresponds to the “CBEFF_BDB_biometric_subtype” in ISO/IEC 19785-1.

NOTE 3: This structure is primarily used within a BIR header; however, it is also used as an input parameter for functions that capture biometric data. The BioAPI_NO_SUBTYPE_AVAILABLE value is used in the BIR header when subtype information is not available. BioAPI_NO_SUBTYPE_AVAILABLE is also used as a function parameter when the application allows the BSP to determine which subtype is to be captured.

7.59.10 The use of the bit position BioAPI_BIR_SUBTYPE_MULTIPLE is not recommended when the BioAPI version number in use is 2.1.

2-7) Replace subclause 7.57 (BioAPI_VERSION) with the following text:

7.57 BioAPI_VERSION

This type is used to represent the version of the BioAPI or FPI specification to which components or data have been implemented. This type is used in the function BioAPI_Init, within the BIR header, and within schemas in the component registry.

The two following values are specified in this edition of this International Standard:

(1) 32 decimal (20 hex), corresponding to a Major value of 2 and a Minor value of 0, and representing BioAPI 2.0; and

(2) 33 decimal (21 hex), corresponding to a Major value of 2 and a Minor value of 1, and representing BioAPI 2.1.

```
typedef uint8_t BioAPI_VERSION;
```

NOTE 1: This type is not used for product versions, which are generally represented by strings.

NOTE 2: The BioAPI Version used within the BIR header corresponds to the “CBEFF_patron_header_version” in ISO/IEC 19785-1.

The BioAPI Version is a concatenation of the major and minor version values such that first hex digit represents the major version and the second hex digit represents the minor version:

BioAPI_VERSION 0xnm

where n = MajorVersion and m=MinorVersion:

2-8) Replace the heading of subclause 7.16 (BioAPI_BSP_SCHEMA) with the following text:

7.16 BioAPI_BSP_SCHEMA (BioAPI 2.0)

This subclause applies only when the BioAPI version number in use is 2.0.

2-9) Add the following text after 7.59:

7.60 BioAPI_BSP_SCHEMA (BioAPI 2.1)

7.60.1 This subclause applies only when the BioAPI version number in use is 2.1.

7.60.2 Information about a BSP, maintained in the component registry.


```
typedef struct bioapi_bsp_schema {
        BioAPI_UUID BSPUuid;
        BioAPI_STRING BSPDescription;
        uint8_t *Path;
        BioAPI_VERSION SpecVersion;
        BioAPI_STRING ProductVersion;
        BioAPI_STRING Vendor;
        BioAPI_BIR_BIOMETRIC_DATA_FORMAT *BSPSupportedFormats;
        uint32_t NumSupportedFormats;
        BioAPI_BIR_BIOMETRIC_TYPE FactorsMask;
        BioAPI_OPERATIONS_MASK Operations;
        BioAPI_OPTIONS_MASK Options;
        BioAPI_FMR PayloadPolicy;
        uint32_t MaxPayloadSize;
        int32_t DefaultVerifyTimeout;
        int32_t DefaultIdentifyTimeout;
        int32_t DefaultCaptureTimeout;
        int32_t DefaultEnrollTimeout;
        int32_t DefaultCalibrateTimeout;
        uint32_t MaxBSPDbSize;
        uint32_t MaxIdentify;
        uint32_t  MaxNumEnrollInstances;
        uint8_t   *HostingEndpointIRI;
        BioAPI_UUID  BSPAccessUUID;
    }BioAPI_BSP_SCHEMA;
```


7.60.3 Definitions

BSPUuid – UUID of the BSP.

BSPDescription – A NUL-terminated string containing a text description of the BSP.

Path – A pointer to a NUL-terminated string containing the path of the file containing the BSP executable code, including the filename. The path may be a URL. This string shall consist of ISO/IEC 10646 characters encoded in UTF-8 (see ISO/IEC 10646, Annex D).

NOTE: When BioAPI_BSP_SCHEMA is used within a function call, the component that receives the call allocates the memory for the *Path* schema element and the calling component frees the memory.

SpecVersion – Major/minor version number of the BioAPI specification to which the BSP was implemented.

ProductVersion – The version string of the BSP software.

Vendor – A NUL-terminated string containing the name of the BSP vendor.

BSPSupportedFormats – A pointer to an array of BioAPI_BIR_BIOMETRIC_DATA_FORMAT structures specifying the supported BDB formats.

NumSupportedFormats – Number of supported formats contained in BSPSupportedFormats.
FactorsMask – A mask which indicates which biometric types are supported by the BSP.
Operations – A mask which indicates which operations are supported by the BSP.

Options – A mask which indicates which options are supported by the BSP.

PayloadPolicy – Threshold setting (maximum FMR value) used to determine when to release the payload after successful verification.

MaxPayloadSize – Maximum payload size (in bytes) that the BSP can accept.

DefaultVerifyTimeout – Default timeout value in milliseconds used by the BSP for ***BioAPI_Verify*** operations when no timeout is specified by the application.

DefaultIdentifyTimeout – Default timeout value in milliseconds used by the BSP for ***BioAPI_Identify*** and ***BioAPI_IdentifyMatch*** operations when no timeout is specified by the application.

DefaultCaptureTimeout – Default timeout value in milliseconds used by the BSP for ***BioAPI_Capture*** operations when no timeout is specified by the application.

DefaultEnrollTimeout – Default timeout value in milliseconds used by the BSP for ***BioAPI_Enroll*** operations when no timeout is specified by the application.

DefaultCalibrateTimeout **–** Default timeout value in milliseconds used by the BSP for sensor calibration operations when no timeout is specified by the application.

MaxBSPDbSize – Maximum size of a BSP-controlled BIR database.

NOTE 1: Applies only when a BSP is only capable of directly managing a single archive unit.
NOTE 2: A value of zero means that no information about the database size is being provided for one of the following three reasons:

a) databases are not supported,

b) it is capable of managing multiple units (either directly or through a BFP interface), each of which may have a different “maximum size” and information about these units will be provided as part of the insert notification (part of Unit Schema), or

c) one archive unit is supported, but the information is not given here – it will be provided in the insert notification.

MaxIdentify – Largest population supported by the identify function. Unlimited = FFFFFFFF.

MaxNumEnrollInstances – The maximum number of distinct instances that a BSP can create reference templates for in one enroll operation. This information can be useful to an application that uses the application-controlled GUI feature.

HostingEndpointIRI – An IRI identifying the framework whose component registry contains a registration of the BSP. This parameter shall be ignored by frameworks conforming to this part of this International Standard and shall be set to NULL by an application. It is provided to support interworking standards, which may specify the use of identical BSPs present on multiple computers from within an application running on the same or a different computer.

BSPAccessUUID – A UUID, unique within the scope of an application, which the application may use to refer to the BSP as an alternative to the BSP product UUID. This parameter shall be ignored by frameworks conforming to this part of this International Standard and can be set to any UUID value by an application. It is provided to support interworking standards, which may specify the use of identical BSPs present on multiple computers from within an application running on the same or a different computer.

NOTE: The "BSPAccess_UUID" and the "HostingendpointIRI" are part of the definition of the C type **`BioAPI_BSP_SCHEMA`**, but are not part of the BSP schema information stored in the component registry (see Table 3).

7.60.4 See subclause 10.1.2 and 10.2.1 for further explanation of schema elements and for BSP insertion of information into the component registry.

2-10) Replace the parameter definitions in 7.28.2 with the following text:

`BSPUuid` (input) – The UUID of the BSP raising the event.

`UnitID` (input) – The unit ID of the BioAPI Unit associated with the event.

`AppNotifyCallbackCtx` (input) – A generic pointer to context information that was provided in the call to **`BioAPI_BSPLoad`** that established the event handler.

`UnitSchema` (input) – A pointer to the unit schema of the BioAPI Unit associated with the event.

`EventType` (input) – The **`BioAPI_EVENT`** that has occurred.

2-11) Replace the heading of subclause 7.31 (BioAPI_GUI_BITMAP) with the following text:

7.31 BioAPI_GUI_BITMAP (BioAPI 2.0)

This subclause applies only when the BioAPI version number in use is 2.0.

2-12) Add the following text after 7.60:

7.61 BioAPI_GUI_BITMAP (BioAPI 2.1)

7.61.1 This subclause applies only when the BioAPI version number in use is 2.1.

7.61.2 This structure provides a graphic for display by the application.

```
typedef struct bioapi_gui_bitmap {
   BioAPI_BIR_SUBTYPE_MASK SubtypeMask;
      /* The subtype (or subtypes) of the sample represented
      by the bitmap */
   uint32_t  Width;
      /* Width of bitmap in pixels (number of pixels for each line) */
   uint32_t  Height;
      /* Height of bitmap in pixels (number of lines) */
   BioAPI_DATA  Bitmap;
} BioAPI_GUI_BITMAP;
```

7.61.3 Definitions

`Bitmap` – A series of 8-bit grayscale pixels (where '00' = black and 'FF' = white), read from left to right, top to bottom, as specified in Table Amd.1-1.

Table Amd.1-1 — GUI bitmap format

Byte position	Meaning	Description
0	line 0, pixel 0	first pixel of first line
1	line 0, pixel 1	second pixel of first line
…	…	
`Width` -1	line 0, pixel (**`Width`** -1)	last pixel of first line
`Width`	line 1, pixel 0	first pixel of second line
…	…	
(**`Width`** * **`Height`**) - 1	line (**`Width`** - 1), pixel (**`Height`** - 1)	last line, last pixel

NOTE: This is used with the application-controlled GUI option. See **`BioAPI_GUI_STATE_EVENT_HANDLER`** and **`BioAPI_GUI_PROGRESS_EVENT_HANDLER`** descriptions in 7.71.

2-13) Replace the heading of subclause 7.32 (BioAPI_GUI_MESSAGE) with the following text:

7.32 BioAPI_GUI_MESSAGE (BioAPI 2.0)

This subclause applies only when the BioAPI version number in use is 2.0.

2-14) Add the following text after 7.61:

7.62 BioAPI_GUI_ENROLL_TYPE (BioAPI 2.1)

This subclause applies only when the BioAPI version number in use is 2.1.

The type BioAPI_GUI_ENROLL_TYPE indicates the enroll type of a BSP, and is defined as follows:

```
typedef uint32_t BioAPI_GUI_ENROLL_TYPE;

#define BioAPI_GUI ENROLL_TYPE TEST_VERIFY      (0x00000001)

#define BioAPI_GUI ENROLL_TYPE MULTIPLE_CAPTURE (0x00000002)
```

The enroll type of a BSP denotes a pattern of suboperations performed by the BSP during an enroll operation.

The bit position BioAPI_GUI ENROLL_TYPE TEST_VERIFY indicates, when set, that the BSP supports test verification at enrollment. During an enroll operation, an application receives GUI state event notification callbacks for two capture suboperations, GUI state event notification callbacks for a process suboperation, GUI state event notification callbacks for a create template suboperation, and GUI state event notification callbacks for a verify match suboperation;

The bit position BioAPI_GUI ENROLL_TYPE MULTIPLE_CAPTURE indicates, when set, that the BSP supports multiple capture suboperations at enrollment. During an enroll operation, an application receives GUI state event notification callbacks for multiple capture suboperations followed by GUI state event notification callbacks for a create template suboperation.

The two bit positions shall not be set at the same time.

NOTE: This restriction is imposed because an application, in order to support a combination of Test-Verify and Multiple-Capture, would need more information about the timing of the suboperations in order to decide the layout of its screen during enrollment.

7.63 BioAPI_GUI_BITMAP_ARRAY (BioAPI 2.1)

This subclause applies only when the BioAPI version number in use is 2.1.

The type **BioAPI_GUI_BITMAP_ARRAY** is defined as follows:


```
typedef struct bioapi_gui_bitmap_array {
    uint32_t          NumberOfMembers;
    BioAPI_GUI_BITMAP    *Bitmaps;
        /* A pointer to an array of BioAPI_GUI_BITMAPs */
} BioAPI_GUI_BITMAP_ARRAY;
```

7.64 BioAPI_BIR_SUBTYPE_MASK (BioAPI 2.1)

This subclause applies only when the BioAPI version number in use is 2.1.

The type **BioAPI_BIR_SUBTYPE_MASK** is defined as follows:

```
typedef uint32_t BioAPI_BIR_SUBTYPE_MASK;

#define BioAPI_BIR_SUBTYPE_LEFT_BIT               (0x0001)
#define BioAPI_BIR_SUBTYPE_RIGHT_BIT              (0x0002)
#define BioAPI_BIR_SUBTYPE_LEFT_THUMB_BIT         (0x0004)
#define BioAPI_BIR_SUBTYPE_LEFT_POINTERFINGER_BIT (0x0008)
#define BioAPI_BIR_SUBTYPE_LEFT_MIDDLEFINGER_BIT  (0x0010)
#define BioAPI_BIR_SUBTYPE_LEFT_RINGFINGER_BIT    (0x0020)
#define BioAPI_BIR_SUBTYPE_LEFT_LITTLEFINGER_BIT  (0x0040)
#define BioAPI_BIR_SUBTYPE_RIGHT_THUMB_BIT        (0x0080)
#define BioAPI_BIR_SUBTYPE_RIGHT_POINTERFINGER_BIT    (0x0100)
#define BioAPI_BIR_SUBTYPE_RIGHT_MIDDLEFINGER_BIT     (0x0200)
#define BioAPI_BIR_SUBTYPE_RIGHT_RINGFINGER_BIT       (0x0400)
#define BioAPI_BIR_SUBTYPE_RIGHT_LITTLEFINGER_BIT     (0x0800)
#define BioAPI_BIR_SUBTYPE_LEFT_VEIN_PALM_BIT         (0x00001000)
```

```
#define BioAPI_BIR_SUBTYPE_LEFT_VEIN_BACKOFHAND_BIT   (0x00002000)
#define BioAPI_BIR_SUBTYPE_LEFT_VEIN_WRIST_BIT        (0x00004000)
#define BioAPI_BIR_SUBTYPE_RIGHT_VEIN_PALM_BIT        (0x00008000)
#define BioAPI_BIR_SUBTYPE_RIGHT_VEIN_BACKOFHAND_BIT  (0x00010000)
#define BioAPI_BIR_SUBTYPE_RIGHT_VEIN_WRIST_BIT       (0x00020000)
```

NOTE 1: The VEIN values are not present in BioAPI 2.0.

The bit positions **BioAPI_BIR_SUBTYPE_LEFT_BIT** and **BioAPI_BIR_SUBTYPE_RIGHT_BIT** can be used to identify the instance of a biometric modality for which there is one "left" instance and one "right" instance.

NOTE 2: Iris and hand geometry are examples of such modalities.

The other bit positions (from **BioAPI_BIR_SUBTYPE_LEFT_THUMB_BIT** through **BioAPI_BIR_SUBTYPE_RIGHT_LITTLEFINGER_BIT**) can be used to identify the instance of a biometric modality related to the fingers.

NOTE 3: Fingerprint is one such modality.

The value zero (no bit positions set) can be used with biometric modalities for which there is only one instance.

NOTE 4: Signature/sign is one such modality.

7.65 BioAPI_GUI_EVENT_SUBSCRIPTION (BioAPI 2.1)

This subclause applies only when the BioAPI version number in use is 2.1.

This type carries information about an existing named GUI event subscription. It identifies the subscriber application and the named subscription, and indicates which types of GUI events are in the scope of the subscription. This type is specified for use in the **BioAPI_QueryGUIEventSubscriptions** function.

A named GUI event subscription is one that was created by a call to **BioAPI_SubscribeToGUIEvents** specifying a non-**NULL** GUI event subscription UUID. The framework calls the event handlers specified in a named subscription to notify BSP-generated GUI events that have been redirected to that named subscription (see 8.3.7), and to notify application-generated GUI events (see 8.3.3, 8.3.4, and 8.3.5) directed to that named subscription.

```
typedef struct _bioapi_gui_event_subscription  {
    const uint8_t *SubscriberEndpointIRI;
    BioAPI_UUID GUIEventSubscriptionUuid;
    BioAPI_BOOL GUISelectEventSubscribed;
    BioAPI_BOOL GUIStateEventSubscribed;
    BioAPI_BOOL GUIProgressEventSubscribed;
} BioAPI_GUI_EVENT_SUBSCRIPTION;
```

SubscriberEndpointIRI – An IRI -see RFC3987- (originally provided by the framework) that identifies the application that created the named GUI event subscription. This shall be **NULL** if the subscriber application is the same as the current application.

GUIEventSubscriptionUuid – A UUID (originally provided by the subscriber application) that identifies the named GUI event subscription.

GUISelectEventSubscribed – Indicates whether GUI select events are in the scope of the subscription (a non-**NULL** callback address was originally provided by the subscriber application for the GUI select event handler).

GUIStateEventSubscribed – Indicates whether GUI state events are in the scope of the subscription (a non-**NULL** callback address was originally provided by the subscriber application for the GUI state event handler).