INTERNATIONAL STANDARD

ISO/IEC 19784-4

First edition
2011-03-01

Information technology — Biometric application programming interface — Part 4: Biometric sensor function provider interface

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

Partie 4: Interface du fournisseur de fonction du capteur biométrique

Reference number
ISO/IEC 19784-4:2011(E)

Contents

Page

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 shall not be held responsible for identifying any or all such patent rights.

ISO/IEC 19784-4 was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information technology*, Subcommittee SC 37, *Biometrics*.

ISO/IEC 19784 consists of the following parts, under the general title *Information technology — Biometric application programming interface*:

— *Part 1: BioAPI specification*

— *Part 2: Biometric archive function provider interface*

— *Part 4: Biometric sensor function provider interface*

Introduction

This part of ISO/IEC 19784 specifies a low-level interface that enables a Biometric Service Provider (BSP) module (see ISO/IEC 19784-1) to interact with a biometric sensor from a different vendor [a Biometric Sensor Function Provider (BSFP) module], using only the specification of the standardized interface. The biometric sensor is a physical device with a small amount of associated software. The interface enables a biometric sensor to interwork with any BSP that is designed to handle the technology supported by that sensor. It also enables the BSP module to be independent of details of the actual sensor, although it will remain dependent on the biometric technology it supports (finger, face, vein, etc.).

Biometric sensors can have very different working principles. Also the data formats vary very much depending on the biometric feature and the sensor type. To cover the resulting different requirements for a generic function provider, normative annexes contain the specific functions and data structures for typical sensor device classes or biometric modalities. The philosophy of this part of ISO/IEC 19784 is to add such normative annexes whenever the existing annexes do not cover the requirements for a typical sensor class.

Currently there are two normative annexes. Annex A provides type definitions and function calls designed to support biometric sensors designed to return images (either still images or a sequence of images). Annex B provides type definitions and function calls designed to support biometric sensors designed to return sequences of pen movements.

The major function of a BSP module performing a capture is to produce a complete Biometric Information Record (BIR) for returning to a biometric application (usually via a BioAPI framework).

The major function of a BSFP is to do a capture and to return data in an identified format, either as a single piece of data (use of BioSFPI_DataTransfer) or as a series of packets (use of BioSFPI_GetPackets) containing the captured information, or via a stream interface (see Annex C).

The BSP is responsible for turning this data into a Biometric Data Block (BDB) within a BIR for returning across the Service Provider Interface (SPI).

Additional (minor) functions relate to control of the biometric sensor and the parameters of its operation.

Information technology — Biometric application programming interface —

Part 4: Biometric sensor function provider interface

1 Scope

This part of ISO/IEC 19784 specifies a biometric sensor interface for a Biometric Service Provider (BSP, see ISO/IEC 19784-1). The interface supports a BSP wishing to provide the BioAPI Service Provider Interface (SPI) functions, whilst removing device handling activity from the BSP. This part of ISO/IEC 19784 provides an interface that can be used by all types of biometric sensor, including *inter alia* image streaming sensors (infrared, face, iris, finger, etc.), voice streaming sensors and digital tablets providing dynamic signature data.

It is not in the scope of this part of ISO/IEC 19784 to define security and privacy requirements for capturing and transferring of biometric data across the Sensor Function Provider Interface (SFPI).

2 Conformance

A BSFP shall support all the functions in Clause 8 of this part of ISO/IEC 19784 and all those in the two normative annexes. However, those functions that have a permissible error return of BioAPIERR_FUNCTION_NOT_SUPPORTED can always return this error. Functions that do not list this error as an allowed return are required to perform the function as described, subject to other error returns listed.

3 Normative references

The following referenced documents are indispensable for the application of this document. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

ISO/IEC 19784-1, *Information technology — Biometric application programming interface — Part 1: BioAPI specification*

ISO/IEC 19794-7, *Information technology — Biometric data interchange formats — Part 7: Signature/sign time series data*

4 Terms and definitions

For the purposes of this document, the terms and definitions given in ISO/IEC 19784-1 and the following apply.

NOTE Function names and data element names are not included here, but are defined within the body of this part of ISO/IEC 19784.

4.1
Biometric Sensor Function Provider
BSFP
BioAPI component, attached to a BSP via its interface, for managing and providing information from a biometric sensor

NOTE The biometric sensor can be a camera, a fingerprint device, an iris scanner, or a signature reader using images, pressure, or sound, or any other type of biometric sensor.

4.2
Biometric Sensor Function Provider Interface
BSFPI
BSP-to-BSFP interface that supports the functions needed to provide biometric data and to manage the BSFP itself and the associated device

4.3
packet
fragment of a BDB passing across the SFPI as the result of a single BioSFPI_GetPackets call

4.4
streaming data
data passing across an interface from a source that is operating continuously

5 Symbols, abbreviated terms, data structure definitions and error codes

5.1 Symbols and abbreviated terms

API – Application Programming Interface

BDB – Biometric Data Block

BFP – BioAPI Function Provider

BSP – Biometric Service Provider

BSFPI – Biometric Sensor Function Provider Interface

FPI – Function Provider Interface

ID – Identity/Identification/Identifier

SPI – Service Provider Interface

sRGB – Standard Red Green Blue (see ITU-R BT.709-5)

UUID – Universally Unique Identifier

5.2 Data structure definitions and error codes

For the purposes of this document, the data structure definitions and error codes defined in ISO/IEC 19784-1 apply.

6 Interface architecture

6.1 ISO/IEC 19784-1:2006 specifies the interface at the top of the BioAPI Framework, which a biometric application uses to interact with BSPs, and through that to biometric sensors either directly or through BFPs (see Figure 1 and Figure 2 of ISO/IEC 19784-1:2006).

6.2 The BSFPI contains two types of functions:

6.2.1 General BFP management functions, which are called by the BSP to manage the BFP and its associated BioAPI Units. These functions are not directly related to SPI functions.

— BioSFPI_BSFPLoad (see Clause 7)

— BioSFPI_UnitAttach (called in the context of BioAPI_BSPAttach and BioSPI_BSPAttach, see Clauses 8 and 9 of ISO/IEC 19784-1:2006)

— BioSFPI_UnitDetach (called in the context of BioAPI_BSPDetach and BioSPI_BSPDetach, see Clauses 8 and 9 of ISO/IEC 19784-1:2006)

— BioSFPI_QueryUnits (this function is directly related to the SPI function BioSPI_QueryUnits which is related to the API function BioAPI_QueryUnits, see Clauses 8 and 9 of ISO/IEC 19784-1:2006)

— BioSFPI_Free (the relation of this function to the corresponding SPI call depends on the behaviour of the BSP – when the BSP copies data that the BFP has allocated memory for, then it can immediately send BioSFPI_Free to the BFP, otherwise the BSP will call this function when itself is requested to free the memory for this data)

6.2.2 Unit management functions, which are called by the BSP to manage BioAPI Units directly. These functions are directly related to SPI functions.

— BioSFPI_ControlUnit (this function is directly related to the SPI function BioSPI_ControlUnit which is related to the API function BioAPI_ControlUnit, see Clauses 8 and 9 of ISO/IEC 19784-1:2006)

— BioSFPI_Cancel (this function can be called in relation to the corresponding calls of the SPI or the application, but can also be called when there is no corresponding request from the BioAPI framework or application, see Clauses 8 and 9 of ISO/IEC 19784-1:2006)

— BioSFPI_SetPowerMode (this function is directly related to the SPI function BioSPI_SetPowerMode which is related to the API function BioAPI_SetPowerMode, see Clauses 8 and 9 of ISO/IEC 19784-1:2006)

— BioSFPI_SetGUICallbacks (this function is directly related to the SPI function BioSPI_SetGUICallbacks which is related to the API function BioAPI_SetGUICallbacks, see Clauses 8 and 9 of ISO/IEC 19784-1:2006)

— BioSFPI_SetIndicatorStatus (this function is directly related to the SPI function BioSPI_SetIndicatorStatus which is related to the API function BioAPI_SetIndicatorStatus, see Clauses 8 and 9 of ISO/IEC 19784-1:2006)

— BioSFPI_GetIndicatorStatus (this function is directly related to the SPI function BioSPI_GetIndicatorStatus which is related to the API function BioAPI_GetIndicatorStatus, see Clauses 8 and 9 of ISO/IEC 19784-1:2006)

— BioSFPI_CalibrateSensor (this function is directly related to the SPI function BioSPI_CalibrateSensor which is related to the API function BioAPI_CalibrateSensor, see Clauses 8 and 9 of ISO/IEC 19784-1:2006)

The following functions are important for the interface between a BSP and an SFP, but do not map directly to any function across the SPI between a BSP and the framework (or an application for frameworkless operation).

- BioSFPI_QueryFormatInfo (this function is not related to the SPI)
- BioSFPI_ActivateSensor (this function is not related to the SPI)
- BioSFPI_GetPackets (this function is not related to the SPI)
- BioSFPI_DataTransfer (this function is not related to the SPI)
- BioSFPI_SubscribeToGUIEvents (this function is not related to the SPI)
- BioSFPI_UnsubscribeFromGUIEvents (this function is not related to the SPI)

7 Selecting and loading BFPs and BioAPI_Units

7.1 Obtaining information about BFPs and BioAPI_Units

The component registry contains information about the BioAPI framework, BSPs and BFPs (see 6.3 of ISO/IEC 19784-1:2006).

BFP information is stored in a structure called BioAPI_BFP_SCHEMA (see 7.3 of ISO/IEC 19784-1:2006). The BioAPI_BFP_SCHEMA contains data elements BFPPropertyID and BFPProperty. The BFPPropertyID is a UUID identifying the type of data structure of the BFPProperty data.

Each BFP that is compliant with ISO/IEC 19784-1 and this part of ISO/IEC 19784 has to provide a standardized BFPPropertyID and a related BFPPropertySchema. Two BSFPPropertyIDs and two related BSFPPropertySchemas are defined in Annexes A and B. Annex A defines a BSFPPropertySchema suitable for image devices, and Annex B defines one suitable for signature devices. These can be used in ISO/IEC 19784-1 calls as BFPPropertyIDs and BFPPropertySchemas.

The information given in the BFPPropertySchema is stored at BFP installation time (see 6.4 of ISO/IEC 19784-1:2006) in the component registry.

An application can obtain this information by calling BioAPI_EnumBFPs and BioAPI_QueryBFPs (see 8.1.10 and 8.1.11 of ISO/IEC 19784-1:2006).

A BSP can obtain this information by using the BioSPI_BFP_ENUMERATION_HANDLER (see 9.2.2 of ISO/IEC 19784-1:2006).

An application can retrieve information about all BioAPI_Units supported by a particular BSP by calling the function BioAPI_QueryUnits (see 8.1.9 of ISO/IEC 19784-1:2006). This function also works if no attach session is established yet.

A BSP can browse the component registry for all installed BFPs by using the callback mechanism of BioSPI_BFP_ENUMERATION_HANDLER. The BSP analyses the BFPPropertySchemas and decides which BFPs it can support.

Note: The criteria for that decision are out of the scope of this part of ISO/IEC 19784.

During the call BioSPI_QueryUnits the BSP may load the supported BFPs. A loaded BFP automatically detects the supported BioAPI_Units and reports the BioAPI_UNIT_SCHEMAs in the return of the BioSFPI_QueryUnits call to the BSP, which reports them to the application.

The BSP may create a list of all BFPUuids and the UnitIDs of the supported devices for each BFP. Additional information can be found in the parameter *UnitManagerUuid* of the BioAPI_UNIT_SCHEMA.

7.2 Loading BFPs

7.2.1 Loading BFPs when attaching to an unspecified BioAPI_Unit

This sub-clause describes how a BFP is loaded if the application does not choose a specific BioAPI_Unit in a call of BioAPI_Attach. (If the application selects a specific BioAPI_Unit, see 7.2.2).

In this case the application calls the function BioAPI_BSPAttach with the value BioAPI_DONT_CARE in the particular element of the parameter *UnitList*.

The BSP then performs the operation described in 7.1 using the callback mechanism of BioSPI_BFP_ENUMERATION_HANDLER and followed by a call of BioSFPI_QueryUnits. As a result, the BSP has a list of all units supported and their supporting BFPs. The BSP chooses one of the BioAPI_Units and then loads the appropriate BSFP and completes the operation with a BioSFPI_BSFPLoad function call followed by a BioSFPI_BSFPAttach function call.

Note: The criteria used to choose one of the supported BioAPI_Units are not defined in this part of ISO/IEC 19784.

7.2.2 Loading BFPs when attaching to a specific BioAPI_Unit

After issuing a BioAPI_BSPLoad, the framework will load the BSP (if supported) and issue a BioSPI_BSPLoad.

On return, the application calls the function BioAPI_QueryUnits to determine the units supported by that BSP, and then calls BioAPI_Attach giving the specific Unit it wishes to use. The application retrieves the UnitSchemaArray from the BioAPI_QueryUnits and can browse this for the BioAPI_Unit information. The application selects the *UnitId* it wants to attach to and calls the function BioAPI_BSPAttach. The BSP may decide to load an appropriate BFP at this time depending on either the list being built as described at the end of 7.1 or by analysing the parameter *UnitManagerUuid* of the BioAPI_UNIT_SCHEMA.

If the BSP can support at least one BSFP supporting the requested Unit, it proceeds as specified in 7.1. If it cannot, it will give an error return. If there are multiple choices of BSFP for the selected Unit, the BSP makes a choice (the criteria for doing this are not standardised).

8 BSFPI Definition

8.1 BSFPI data structures

8.1.1 Control codes

The function BioSFPI_ControlUnit uses 32bit values as codes to indicate a specific operation to be performed. To avoid conflicts between standardized and vendor-specific codes these control codes are structured.

The upper 16 bits are used to structure the purpose of control codes. The lower 16 bits are used to identify control codes of each category.

The following three categories are standardized:

```
#define BioSFPI_BFPControlCode (0x8000)
#define BioSFPI_BioAPIUnitControlCode (0x4000)
#define BioSFPI_VendorSpecificControlCode (0x1000)
```

All other category values are reserved for future use.

8.1.2 BioSFPI_EventHandler

This sub-clause defines the event handler interface to receive asynchronous notification of events of type BioAPI_EVENT from a BioAPI Unit. Example events include insertion or removal of a BioAPI Unit (e.g. insertion or withdrawal of a biometric sensor device).

This event handler is passed to the BSFP during BioSFPI_BFPLoad. This is the single event handler that all BioAPI Units managed by this BSFP should use to notify the BSP of event types that occur in a loaded BSFP.

```
typedef BioAPI_RETURN (BioAPI *BioSFPI_EventHandler)
        (const BioAPI_UUID *BSFPUuid,
        BioAPI_UNIT_ID UnitID,
        BioAPI_UNIT_SCHEMA *UnitSchema,
        BioAPI_ EVENT EventType);
```

BSFPUuid (input) - The UUID of the BSFP raising the event.

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

UnitSchema (input) – The schema of the BioAPI Unit raising the event.

EventType (input) - The BioAPI_ EVENT that has occurred.

8.1.3 BioSFPI_GUI_RESPONSE

An enumeration of the possible actions to be performed by a BFP after a GUI event notification callback made by the BFP has returned control to the BFP. Before returning from a notification callback, a BSP assigns a value of this type to an output parameter of the callback (Response).

```
typedef uint8_t BioSFPI_GUI_RESPONSE;

#define BioSFPI_GUI_RESPONSE_PROGRESS_CONTINUE (1)
#define BioSFPI_GUI_RESPONSE_PROGRESS_ABORT (2)
```

The value BioSFPI_GUI_RESPONSE_PROGRESS_CONTINUE can only be returned in response to a GUI progress event notification callback. It indicates that the BFP should continue performing the biometric capture.

The value BioSFPI_GUI_RESPONSE_PROGRESS_ABORT can only be returned in response to a GUI progress event notification callback. It indicates that the BFP should abort the biometric capture and produce an error code. After this response from the BSP, an error return from the original call is expected from the BFP.

If a BSP returns a response value other than those permitted in each situation, the BFP shall abort the biometric capture and cause the original function call to return an error code. No further GUI event notification callbacks are expected from the BFP for the same function.

8.1.4 BioSFPI_GUI_PROGRESS_EVENT_HANDLER

Callback function

```
typedef BioAPI_RETURN (BioAPI *BioSFPI_GUI_PROGRESS_EVENT_HANDLER)
     (BioAPI_UNIT_ID UnitID,
     const void *GUIProgressEventHandlerCtx,
     const BioAPI_GUI_BITMAP_ARRAY *Bitmaps,
     BioSFPI_GUI_RESPONSE *Response);
```

Description

This is a function pointer type for a BSP's GUI event handler function that is to handle GUI progress event notification callbacks coming from a BFP. In order to receive GUI progress event notifications, a BSP shall register a callback function of type BioSFPI_GUI_PROGRESS_EVENT_HANDLER by providing the callback address of the function, along with a context address, in a call to BioSFPI_SubscribeToGUIEvents (see 8.2.16).

The framework makes a callback to an application function of this type each time it receives an incoming callback to a function of type BioSPI_GUI_PROGRESS_EVENT_HANDLER which it exposes to a BFP. The parameters of the callback (except GUIProgressEventHandlerCtx) shall be set from the parameters of the incoming callback with the same names; the parameter GUIProgressEventHandlerCtx shall be set from the GUI progress context address originally provided by the BSP in its call to BioSFPI_SubscribeToGUIEvents; and the callback address shall be set from the GUI progress callback address originally provided by the BSP in the call to BioSFPI_SubscribeToGUIEvents.

A BFP can generate GUI progress events only during the execution of a call to BioSFPI_DataTransfer, BioSFPI_Play or any other function that involves a biometric capture. They can be generated at any time, even multiple times, during any biometric capture.

The main purpose of the GUI progress event notification is to send to the BSP streaming data (as a series of bitmaps) collected by the biometric sensor. One possible use of this information is for the BSPs to show the bitmaps to the subject or an operator. The BSP shall respond to each GUI progress notification by indicating whether the biometric capture should continue or abort (see 8.1.3).

The GUI progress event handler function and any functions invoked (directly or indirectly) by that function shall not call any BioSFPI function.

If the BSP returns a value different from BioAPI_OK, the BFP shall cancel the current biometric capture and produce an error code.

Parameters

UnitID (input) – The unit ID of the biometric sensor unit of the BSP to which the GUI event is related.

GUIProgressEventHandlerCtx (input) – The context address originally provided by the subscriber application as part of the GUI event subscription.

Bitmaps (input) – An array of bitmaps containing image(s) of the samples captured in the biometric capture, which are intended for display to the subject or to an operator. This array usually contains zero or one bitmaps, but may contain multiple bitmaps if the BFP has the ability to capture multiple instances of a biometric modality in a single capture operation.

Response (output) – A value indicating the response from the BSP to the GUI select event notification (see 8.1.3).

Return Value
A BioAPI_RETURN value indicating success or specifying a particular error condition. The value BioAPI_OK indicates success. All other values represent an error condition.

Errors

```
BioAPIERR_USER_CANCELLED
BioAPIERR_FUNCTION_FAILED
BioAPIERR_FUNCTION_NOT_SUPPORTED
```

8.2 BSFP Functions

8.2.1 BioSFPI_BSFPLoad

```
BioAPI_RETURN BioAPI BioSFPI_BSFPLoad
    (const BioAPI_UUID *BSFPUuid,
     BioSFPI_EventHandler BioSFPINotifyCallback);
```

Description
This function initializes a BFP. Initialization includes registering the BSP event handler for the specified BFP and enabling all events. The BSP can choose to provide an event handler function to receive notification of events. Many BSP's can independently and concurrently load the same BFP, and each BSP can establish its own event handler. They will all receive notification of an event. The same or different event handlers can be used if a BSP loads multiple BFPs.

A BSP may establish as many event handlers as it wishes, for a given BFP, by calling **BioSFPI_BSFPLoad** one or more times for that BFP. An event handler is identified by the address of the notification.

When an event occurs in a BFP, the BFP may send an event notification to the BSP by calling the BSP's event handler.

If a BSP has set up multiple event handlers, they shall be called one at a time (in any order chosen by the BFP) rather than concurrently.

Event notification may occur at any time, either during a BioSPI call (related or unrelated to the event) or while there is no BioSPI call in execution. BSP writers should ensure that all callbacks are properly and safely handled by the BSP, no matter when the BSP receives them.

NOTE: This usually requires the use of thread synchronization techniques and discipline in the actions performed by the BSP code placed in event handlers.

When a BFP is loaded (**BioSFPI_BSFPLoad**), it shall raise an “insert” event immediately for each present BioAPI Unit. This will indicate to the BSP that it can go ahead and do a ***BioSFPI_UnitAttach***. If the hardware component for a specific functionality is not present, the “insert” event cannot be raised until the hardware component has been plugged in.

The function ***BioSFPI_UnitAttach*** can be invoked multiple times for each call to ***BioSFPI_BSFPLoad***. The *BSFPUuid* identifies the invoked BFP.

The *BioSFPINotifyCallback* defines a callback used to notify the BSP of events of type BioAPI_EVENT. The BSFP shall retain this information for later use.

Parameters

BSFPUuid (input) – The UUID of the invoked BSFP. Used to locate the component’s directory entry.

BioSFPINotifyCallback (input) – A function pointer for the event handler that manages events of type BioAPI_ EVENT.

Return Value

A BioAPI_RETURN value indicating success or specifying a particular error condition. The value BioAPI_OK indicates success. All other values represent an error condition.

Errors

```
BioAPIERR_MODULE_LOAD_FAIL
BioAPIERR_INVALID_UUID
```

8.2.2 BioSFPI_UnitAttach

```
BioAPI_RETURN BioAPI BioSFPI_UnitAttach
    (BioAPI_UNIT_ID UnitID);
```

Description
This function creates an Attach Session.

There is no requirement that the unit ID value provided by a BSP as input to this function be the same unit ID value that the framework provided to the BSP in the original call to BioSPI_BSPAttach (if any), provided that the two unit ID values identify the same BioAPI unit (see 8.2.4).

Parameters

UnitID (input) – The ID of that BioAPI Unit defined by a prior call to BioSFPI_QueryUnits. The UnitID has been filled into the UnitId field of each element of the BioAPI_UNIT_SCHEMA by the BSFP.

Return Value

A BioAPI_RETURN value indicating success or specifying a particular error condition. The value BioAPI_OK indicates success. All other values represent an error condition.

Errors

```
BioAPIERR_INVALID_UNIT_ID
BioAPIERR_UNIT_IN_USE
```

8.2.3 BioSFPI_UnitDetach

```
BioAPI_RETURN BioAPI BioSFPI_UnitDetach
    (BioAPI_UNIT_ID UnitID);
```

Description

This function terminates an Attach Session.

There is no requirement that the unit ID value provided by a BSP as input to this function be the same unit ID value that the framework provided to the BSP in the original call to BioSPI_BSPAttach (if any), provided that the two unit ID values identify the same BioAPI unit (see 8.2.4).

Parameters

UnitID (input) – The ID of that BioAPI Unit defined by a prior call to BioSFPI_QueryUnits. The UnitID has been filled into the UnitId field of each element of the BioAPI_UNIT_SCHEMA by the BSFP.

Return Value

A BioAPI_RETURN value indicating success or specifying a particular error condition. The value BioAPI_OK indicates success. All other values represent an error condition.

Errors

```
BioAPIERR_INVALID_UNIT_ID
```

8.2.4 BioSFPI_QueryUnits

```
BioAPI_RETURN BioAPI BioSFPI_QueryUnits
   (const BioAPI_UUID *BsfpUuid,
    BioAPI_UNIT_SCHEMA **UnitSchemaArray,
    uint32_t *NumberOfElements);
```

Description

This function returns an array of BioAPI Unit schemas (see 7.55 of ISO/IEC 19784-1:2006), which are managed by the given BSFP and are currently in the inserted state.

NOTE When the BSP calls the function ***BioSFPI_QueryUnits*** of a BFP, the BFP allocates memory for the data to be returned to the BSP. In some implementation architectures, the BSP will simply pass to the framework the data and the addresses exactly as returned by the BFP because the framework will interpret the addresses in the same way as the BFP and will be able to access the data that the BFP has placed at those addresses. In other implementation architectures, the BSP will need to move all the data returned by the BFP to newly allocated memory blocks accessible to the framework, and will call ***BioSFPI_Free*** after copying each memory block but before returning from the ***BioSPI_QueryUnits*** call. In the former case, when the framework calls ***BioSPI_Free***, the BSP will make a corresponding call to ***BioSFPI_Free***. In the latter case, the calls to ***BioSPI_Free*** will be handled internally by the BSP. However, such differences in the behaviour of the BSP are not visible to the Framework.

The memory block containing the array shall be freed by the BSP via a call to ***BioSFPI_Free*** when it is no longer needed by the BSP. The memory block pointed to by the *UnitProperty* member within each element of the array shall also be freed by the BSP via a call to ***BioSFPI_Free*** when it is no longer needed by the BSP.

This function shall only be called after ***BioSFPI_Load*** has been called for the specified BFP.

There is no requirement that a unit ID, returned by this function for a given BioAPI unit, be made available with the same unit ID value by the BSP to the framework. A BSP is free to translate any unit ID value (provided by a BFP) into a different unit ID value before providing it to the framework. The purpose of such a translation would be to avoid the existence of duplicate unit IDs within the scope of the BSP, which might happen when a BSP is using two or more BFPs of the same category, or when a BSP is using a BSFP while also directly managing biometric sensors.

Parameters

BsfpUuid (input) – The unique identifier for the BSFP for which the information is to be returned.

UnitSchemaArray (output) – A pointer to an address of the array of elements of type BioAPI_UNIT_SCHEMA (allocated by the BSP - but see note above) containing the unit schema information.

NumberOfElements (output) – A pointer to the number of elements in the array, which are managed by the given BSFP and are currently in the inserted state.

Return Value

A BioAPI_RETURN value indicating success or specifying a particular error condition. The value BioAPI_OK indicates success. All other values represent an error condition.

Errors

```
BioAPIERR_INVALID_UUID
BioAPIERR_MEMORY_ERROR
```

8.2.5 BioSFPI_Free

```
BioAPI_RETURN BioAPI BioSFPI_Free
    (void *Ptr);
```

Description

This function causes the space pointed to by *Ptr* to be de-allocated. If *Ptr* is NULL, no action occurs. Otherwise, if *Ptr* does not match a pointer earlier returned by the BioSFPI functions, or if the space already has been de-allocated by a call to ***BioSFPI_Free***, the behaviour is undefined.

Parameters

Ptr (*input*) – A pointer to the memory to free.

Return Value

A BioAPI_RETURN value indicating success or specifying a particular error condition. The value BioAPI_OK indicates success. All other values represent an error condition.

Errors

8.2.6 BioSFPI_ControlUnit

```
BioAPI_RETURN BioAPI BioSFPI_ControlUnit
    (BioAPI_UNIT_ID   UnitID,
     uint32_t ControlCode,
     const BioAPI_DATA *InputData,
    BioAPI_DATA **OutputData);
```

Description

This function sends control data to the BioAPI Unit and receives status or operation data from there. The content of the *ControlCode*, the send (input) data, and the receive (output) data will be specified in the related interface specification for this BioAPI Unit.

The function allocates a memory block large enough to contain the output data that are to be returned to the application, fills the memory block with the data, and sets the fields *Length* and *Data* of the *OutputData* structure to the size and address of the memory block (respectively).

The memory block returned by the BioSFPI function call shall be freed by the application using ***BioSFPI_Free***.

This function is a transparent channel to the BioAPI Unit and can be used for any purposes that cannot be realized by standardized functions (e.g. manage the contrast settings, manage the focus etc.).

There is no requirement that the unit ID value provided by a BSP as input to this function be the same unit ID value that the framework provided to the BSP in the original call to BioSPI_BSPAttach (if any), provided that the two unit ID values identify the same BioAPI unit (see 8.2.4).

Parameters

UnitID (input) – The ID of that BioAPI Unit previously defined by the BSFP.

ControlCode (input) – The function code in the BioAPI unit to be called.

InputData (input) – Address and length of a buffer containing the data to be send to the BioAPI unit related to the given *ControlCode*.

OutputData (output) – Pointer to a BioAPI_DATA structure. On output, this shall contain the address and length of a data buffer containing the data received from the BioAPI unit after processing the function indicated by the *ControlCode*. If no memory block has been allocated by the function, the address shall be set to NULL and the length shall be set to zero.

Return Value

A BioAPI_RETURN value indicating success or specifying a particular error condition. The value BioAPI_OK indicates success. All other values represent an error condition.

Errors

```
BioAPIERR_MEMORY_ERROR
BioAPIERR_INVALID_POINTER
BioAPIERR_FUNCTION_FAILED
BioAPIERR_FUNCTION_NOT_SUPPORTED
```

8.2.7 BioSFPI_Cancel

```
BioAPI_RETURN BioAPI BioSFPI_Cancel
    (BioAPI_UNIT_ID   UnitID);
```

Description

This function shall cancel the presently blocked call to the BioAPI Unit. The function shall not return until the blocking call has been cancelled.

The implementation of the SFP shall ensure that this function can be called from the BSP at any time. A presently blocked call to the BioAPI Unit shall not prevent calling this function.

There is no requirement that the unit ID value provided by a BSP as input to this function be the same unit ID value that the framework provided to the BSP in the original call to BioSPI_BSPAttach (if any), provided that the two unit ID values identify the same BioAPI unit (see 8.2.4).

Parameters

UnitID (input) – The ID of that BioAPI Unit where the blocked call shall be cancelled.

Return Value

A BioAPI_RETURN value indicating success or specifying a particular error condition. The value BioAPI_OK indicates success. All other values represent an error condition.

Errors

```
BioAPIERR_UNIT_IN_USE - indicates that underlying layers could not cancel
                        the presently blocked call properly. Either further
                             proprietary software or hardware actions are
                             needed to reset the related components.
BioAPIERR_INVALID_UNIT_ID
BioAPIERR_UNIT_NOT_INSERTED
```

8.2.8 BioSFPI_SetPowerMode

```
BioAPI_RETURN BioAPI BioSFPI_SetPowerMode
    (BioAPI_UNIT_ID  UnitId,
     BioAPI_POWER_MODE  PowerMode);
```

Description

This function sets the currently attached BioAPI Unit to the requested power mode if the BioAPI Unit supports it.

There is no requirement that the unit ID value provided by a BSP as input to this function be the same unit ID value that the framework provided to the BSP in the original call to BioSPI_BSPAttach (if any), provided that the two unit ID values identify the same BioAPI unit (see 8.2.4).

Parameters

UnitId (input) – The ID of the BioAPI Unit for which the power mode is to be set. The BioAPI_DONT_CARE option is not valid for this function.

PowerMode (input) – A 32-bit value indicating the power mode to set the BioAPI Unit to.

Return Value

A BioAPI_RETURN value indicating success or specifying a particular error condition. The value BioAPI_OK indicates success. All other values represent an error condition.

Errors

```
BioAPIERR_FUNCTION_NOT_SUPPORTED
```

8.2.9 BioSFPI_SetGUICallbacks

```
BioAPI_RETURN BioAPI BioSFPI_SetGUICallbacks
    (BioAPI_UNIT_ID  UnitId,
    BioAPI_GUI_STREAMING_CALLBACK GuiStreamingCallback,
    Void   *GuiStreamingCallbackCtx,
    BioAPI_GUI_STATE_CALLBACK  GuiStateCallback,
    Void   *GuiStateCallbackCtx);
```

Description
This function allows the BSP to establish callbacks so that the BSP may get data information by receiving from the BSFP a sequence of bit-map images, called streaming data, as well as state information.

NOTE: Not all BSFPs support the provision of streaming data.

Parameters:

UnitId (input) – The ID of the BioAPI Unit, which may return status and data callback information.

GuiStreamingCallback (input) – A pointer to a BSP callback to deal with the presentation of biometric streaming data.

GuiStreamingCallbackCtx (input) – A generic pointer to context information that was provided by the BSP and will be presented on the callback.

GuiStateCallback (input) – A pointer to a BSP callback to deal with GUI state changes.

GuiStateCallbackCtx (input) – A generic pointer to context information that was provided by the BSP and will be presented on the callback.

NOTE 1 – Function sub-types for BioAPI_GUI_STATE_CALLBACK and BioAPI_GUI_STREAMING_CALLBACK are defined in 7.36 and 7.37 of ISO/IEC 19784-1:2006, respectively.

NOTE 2 – See also clause C.7 of ISO/IEC 19784-1:2006 on User Interface Considerations.

Return Value
A BioAPI_RETURN value indicating success or specifying a particular error condition. The value BioAPI_OK indicates success. All other values represent an error condition.

Errors

```
BioAPIERR_INVALID_POINTER
BioAPIERR_FUNCTION_NOT_SUPPORTED
```

8.2.10 BioSFPI_QueryFormatInfo

```
BioAPI_RETURN BioAPI BioSFPI_QueryFormatInfo
    (BioAPI_UNIT_ID  UnitId,
     BioAPI_UUID  *FormatInfo);
```

Description
This function is used to query a biometric sensor unit to determine what type of data format it supports. The UUID received equals one of those specified in an annex of this standard. (see A.4, B.4).

There is no requirement that the unit ID value provided by a BSP as input to this function be the same unit ID value that the framework provided to the BSP in the original call to BioSPI_BSPAttach (if any), provided that the two unit ID values identify the same BioAPI unit (see 8.2.4).

Parameters

UnitId (input) – The ID of the BioAPI Unit being addressed for capturing biometric data.

FormatInfo (output) – A pointer to a buffer retrieving the UUID describing the data format supported by the biometric sensor unit.

Return Value

A BioAPI_RETURN value indicating success or specifying a particular error condition. The value BioAPI_OK indicates success. All other values represent an error condition.

Errors

```
BioAPIERR_FUNCTION_NOT_SUPPORTED
```

8.2.11 BioSFPI_GetPackets

```
BioAPI_RETURN BioAPI BioSFPI_GetPackets
    (BioAPI_UNIT_ID UnitId,
     BIOAPI_DATA *MemoryForTransfer,
     uint32_t *PacketLength,
     BioAPI_BIR_BIOMETRIC_DATA_FORMAT    *BDBFormat);
```

Description

This function requests the BSFP to fill the BioAPI_DATA structure referenced by parameter MemoryForTransfer, and to return with the length of the inserted packet in PacketLength. The memory is allocated by the BSP with a size as derived from BioSFPI_UNIT_IMAGE_PROPERTY_SCHEMA in Annex A or BioSFPI_UNIT_SIGNATURE_PROPERTY_SCHEMA in Annex B.

This function identifies the BioAPI_Unit and the BDB format that will be returned by one or a sequence of calls of BioSFPI_GetPackets. The BSFP can overwrite the BDBFormat parameter and return a different format than requested by the BSP. It is a decision of the BSP whether to accept that format change or to cancel this operation by not further sending BioSFPI_GetPackets calls.

The BDBFormat is an identifier as listed in ISO/IEC SC 37 Standing Document 9 (SD 9).

The BDB format selected is constrained by the capabilities of the BSFP for that UnitId (see also BioSFPI_UNIT_IMAGE_PROPERTY_SCHEMA in Annex A or BioSFPI_UNIT_SIGNATURE_PROPERTY_SCHEMA in Annex B).

NOTE 1 – In case of requesting multiple consecutive packets for obtaining all data for e.g. a capture process the BSFP should return packets in formats and fractions which can be processed by the BSP without detailed knowledge of BSFP or BioAPI_Unit specifics. The BSFP architecture is intended to help to develop less BioAPI_Unit specific and more universal BSPs.

The BSP will make repeated calls of BioSFPI_GetPackets until "last packet" is signalled by the BSFP.

A multiple buffer principle to increase the performance and data rate of BSP and BSFP can easily be implemented by placing a first call BioSFPI_GetPackets to the BSFP and while this is being processed the BSP can prepare or allocate the next piece of transfer memory. When the first BioSFPI_GetPackets returns the BSP immediately can place the second call to BioSFPI_GetPackets using the previously prepared memory in the MemoryForDataTransfer data structure. While this second call is being processed in the BSFP the returned memory of the first call can be processed by the BSP and again a new piece of memory can be prepared for the third call and so on.

There is no requirement that the unit ID value provided by a BSP as input to this function be the same unit ID value that the framework provided to the BSP in the original call to BioSPI_BSPAttach (if any), provided that the two unit ID values identify the same BioAPI unit (see 8.2.4).

Parameters

UnitId (input) – The ID of the BioAPI Unit being addressed to capture biometric streaming data.

MemoryForTransfer (input/output) – This is a constant pointer that references a BIOAPI_DATA structure which will be used to pass data. The BSFP will place the next data in the MemoryForTransfer, subject to the length limit in that structure, returning the data length. (See also Annex C).

PacketLength (output) – The size (in octets) of the packet placed in the BioAPI_DATA structure.

BDBFormat (input/output) – The Format Owner and Format Type of the BDB which will be returned.

Return Value

A BioAPI_RETURN value indicating success or specifying a particular error condition. The value BioAPI_OK indicates success. All other values represent an error condition.

Errors

```
BioAPIERR_FUNCTION_NOT_SUPPORTED
```

8.2.12 BioSFPI_DataTransfer

```
BioAPI_RETURN BioAPI BioSFPI_DataTransfer
    (BioAPI_UNIT_ID  UnitId,
     uint32_t *StreamID,
     BioAPI_DATA  *SensorData,
     BioAPI_BIR_BIOMETRIC_DATA_FORMAT    *BDBFormat);
```

Description

This function retrieves a BDB resulting from a capture. This function identifies the BioAPI_Unit and the BDB format that will be returned.

This function requests the BSFP to fill the BioAPI_DATA structure referenced by parameter SensorData. The memory is allocated by the BSP with a size as derived from BioSFPI_UNIT_IMAGE_PROPERTY_SCHEMA in Annex A or BioSFPI_UNIT_SIGNATURE_PROPERTY_SCHEMA in Annex B.

The BSFP can overwrite the BDBFormat parameter and return a different format than requested by the BSP. It is a decision of the BSP whether to accept that format change or to cancel this operation by sending BioSFPI_Cancel.

The BDBFormat is an identifier as listed in ISO/IEC SC 37 Standing Document 9 (SD 9).

The BDB format selected is constrained by the capabilities of the BSFP for that UnitId (see also BioSFPI_UNIT_IMAGE_PROPERTY_SCHEMA in Annex A or BioSFPI_UNIT_SIGNATURE_PROPERTY_SCHEMA in Annex B).

There is no requirement that the unit ID value provided by a BSP as input to this function be the same unit ID value that the framework provided to the BSP in the original call to BioSPI_BSPAttach (if any), provided that the two unit ID values identify the same BioAPI unit (see 8.2.4).

Parameters

UnitId (input) – The ID of the BioAPI Unit being addressed for capturing biometric data.

StreamID (input) – The ID for access to a stream function provided by the system integrator on any given platform that enables streaming data to be transferred between the BSFP and the BSP.

SensorData (output) – A pointer to a structure retrieving the biometric data.

BDBFormat (input/output) – The Format Owner and Format Type of the BDB which will be returned.

Return Value

A BioAPI_RETURN value indicating success or specifying a particular error condition. The value BioAPI_OK indicates success. All other values represent an error condition.

Errors

```
BioAPIERR_FUNCTION_NOT_SUPPORTED
```

8.2.13 BioSFPI_SetIndicatorStatus

```
BioAPI_RETURN BioAPI BioSFPI_SetIndicatorStatus
    (BioAPI_UNIT_ID              UnitId,
    BioAPI_INDICATOR_STATUS      IndicatorStatus);
```

Description

This function sets the selected BioAPI Unit to the requested indicator status if the BioAPI Unit supports it. After BioAPI_INDICATOR_ACCEPT or

If BioAPI_INDICATOR_REJECT is set in the IndicatorStatus parameter, the status will not be changed until the application sets another value.

There is no requirement that the unit ID value provided by a BSP as input to this function be the same unit ID value that the framework provided to the BSP in the original call to BioSPI_BSPAttach (if any), provided that the two unit ID values identify the same BioAPI unit (see 8.2.4).

Parameters

UnitId (input) - The ID of the BioAPI Unit for which the indicator status is to be set. The BioAPI_DONT_CARE option is not valid for this function.

IndicatorStatus (input) - A value to which to set the indicator status of the BioAPI Unit.

Return Value

A BioAPI_RETURN value indicating success or specifying a particular error condition. The value BioAPI_OK indicates success. All other values represent an error condition.

Errors

```
BioAPIERR_FUNCTION_NOT_SUPPORTED
BioAPIERR_INVALID_UNIT_ID
BioAPIERR_UNIT_NOT_INSERTED
BioAPIERR_UNIT_IN_USE
```

8.2.14 BioSFPI_GetIndicatorStatus

```
BioAPI_RETURN BioAPI BioSFPI_GetIndicatorStatus
    (BioAPI_UNIT_ID              UnitId,
     BioAPI_INDICATOR_STATUS       *IndicatorStatus);
```

Description

This function returns the indicator status of the BioAPI Unit if that BioAPI Unit supports it.

There is no requirement that the unit ID value provided by a BSP as input to this function be the same unit ID value that the framework provided to the BSP in the original call to BioSPI_BSPAttach (if any), provided that the two unit ID values identify the same BioAPI unit (see 8.2.4).

Parameters

UnitId (input) - The ID of the BioAPI Unit for which the indicator status is to be obtained. The BioAPI_DONT_CARE option is not valid for this function.

IndicatorStatus (output) - A value for the indicator status of the BioAPI Unit.

Return Value

A BioAPI_RETURN value indicating success or specifying a particular error condition. The value BioAPI_OK indicates success. All other values represent an error condition.

Errors

```
BioAPIERR_FUNCTION_NOT_SUPPORTED
BioAPIERR_INVALID_UNIT_ID
BioAPIERR_UNIT_NOT_INSERTED
BioAPIERR_UNIT_IN_USE
```

8.2.15 BioSFPI_CalibrateSensor

```
BioAPI_RETURN BioAPI BioSFPI_CalibrateSensor
    (BioAPI_UNIT_ID     UnitId,
     int32_t            *Timeout);
```

Description

This function performs a calibration of the attached BioAPI Unit.

There is no requirement that the unit ID value provided by a BSP as input to this function be the same unit ID value that the framework provided to the BSP in the original call to BioSPI_BSPAttach (if any), provided that the two unit ID values identify the same BioAPI unit (see 8.2.4).

Parameters

UnitId (input) - The ID of the BioAPI Unit being addressed for capturing biometric data.

Timeout (input/output) – A pointer to an integer specifying the timeout value (in milliseconds) for the operation. If this timeout is reached, the function returns an error. This value can be any positive number. If the value is set to 0 by the BSP this function immediately returns with the value set to the required calibration timeout value of the attached BioAPI_Unit if that biometric sensor supports calibration. If the biometric sensor does not support calibration the timeout value integer returns 0.

Return Value

A BioAPI_RETURN value indicating success or specifying a particular error condition. The value BioAPI_OK indicates success. All other values represent an error condition.

Errors

```
BioAPIERR_FUNCTION_NOT_SUPPORTED
BioAPIERR_UNIT_IN_USE
BioAPIERR_INVALID_UNIT_ID
BioAPIERR_UNIT_NOT_INSERTED
BioAPIERR_CALIBRATION_NOT_SUCCESSFUL
BioAPIERR_TIMEOUT_EXPIRED
```

8.2.16 BioSFPI_SubscribeToGUIEvents

```
BioAPI_RETURN BioAPI BioSFPI_SubscribeToGUIEvents
    (BioAPI_UNIT_ID                      UnitId,
    BioSFPI_GUI_PROGRESS_EVENT_HANDLER   GUIProgressEventHandler,
    const void                           *GUIProgressEventHandlerCtx);
```

Description
This optional function creates a "GUI event subscription" for the current BSP. GUI event subscriptions are maintained by the BSP at runtime.

The BSP shall provide the callback addresses of a GUI progress event handler. All the parameter values provided in a call to this function shall become part of the GUI event subscription maintained by the BSP.

GUI event subscriptions specifying a BFP attach session shall be destroyed automatically by the BSP when the attach session is destroyed.

The context addresses provided in the call shall be passed back by the BFP to the BSP in subsequent callbacks to the event handlers. This International Standard does not assign any meaning to the context addresses, but specifies them to satisfy the needs of some BSPs.

Multiple calls to this function shall not result, at any time, in multiple GUI progress event handlers, established for the same BSP Unit Id. This is to ensure that, for any incoming GUI event, there will be no ambiguity as to which GUI event handler (if any) is to be called by the BSP.

NOTE – This implies that two subscriptions with the same parameters are not allowed.

This function shall only be called (for a given BSP Unit Id) if there is at least one call to BioSFPI_BFPLoad (for that BSFP UUID).

Parameters

UnitId (input) – The ID of the BioAPI Unit, which may return status and data callback information.

GUIProgressEventHandler (input) – The callback address of a BSP function that is to receive GUI progress event notifications from the BFP.

GUIProgressEventHandlerCtx (input) – A context address that is to be passed back by the BFP to the BSP on each callback to the GUI progress event handler.

Return Value
A BioAPI_RETURN value indicating success or specifying a particular error condition. The value BioAPI_OK indicates success. All other values represent an error condition.

Errors
```
BioAPIERR_INVALID_POINTER
BioAPIERR_FUNCTION_NOT_SUPPORTED
```

8.2.17 BioSFPI_UnsubscribeFromGUIEvents

```
BioAPI_RETURN BioAPI BioSFPI_UnsubscribeFromGUIEvents
    (BioAPI_UNIT_ID UnitId,
    BioSFPI_GUI_PROGRESS_EVENT_HANDLER GUIProgressEventHandler
    const void *GUIProgressEventHandlerCtx);
```

Description
This optional function destroys a GUI event subscription previously created by a call to BioSFPI_SubscribeToGUIEvents. The particular subscription that is destroyed is the one whose definition exactly matches the parameters of the call.

NOTE – A call to this function will not destroy a GUI event subscription, even if there is only one subscription, unless the parameters of the call have exactly the same values as those of the call to BioSFPI_SubscribeToGUIEvents that created the subscription. The BSP needs to remember the value of those parameters.

Usually, BSPs will not need to call this function, because all GUI event subscriptions for a given Unit Id are automatically destroyed by the BSP when the attach session of the BFP is destroyed. This function can be used when the BSP needs to modify the current GUI event subscriptions but does not want to destroy an attach session.

This function shall only be called (for a given Unit Id) if there is at least one call to BioSFPI_BSFPLoad (for that BSFP UUID).

Parameters
The parameters of this function have the same meaning as the parameters of the function BioSFPI_SubscribeToGUIEvents with the same names, and are used by the BFP to identify the existing GUI event subscription that is to be destroyed.

Return Value
A BioAPI_RETURN value indicating success or specifying a particular error condition. The value BioAPI_OK indicates success. All other values represent an error condition.

Errors
```
BioAPIERR_INVALID_POINTER
BioAPIERR_FUNCTION_NOT_SUPPORTED
```

Annex A
(normative)

Imaging biometric sensor units

A.1 General

This annex describes biometric sensor units that create still or streaming images.

A.1.1 Bit and Byte ordering

Images are represented in Big-Endian format. That is, the more significant bytes of any multi-byte quantity are stored at lower addresses in memory than less significant bytes. The order for transmission shall also be the most significant byte first and least significant byte last. Within a byte, the order of transmission shall be the most significant bit first and the least significant bit last. All numeric values are fixed-length unsigned integer quantities.

A.1.2 Scan sequence

Each image as presented in accordance with this annex shall appear to have been captured in an upright position. The scanning sequence (and recorded data) shall appear to have been from left-to-right, progressing from top-to-bottom of the biometric feature. For the purpose of describing the position of each pixel within an image to be exchanged, a pair of reference axes shall be used. The origin of the axes, pixel location (0, 0), shall be located at the upper left-hand corner of each image. The x-coordinate (horizontal) position shall increase positively from the origin to the right side of the image. The y-coordinate (vertical) position shall increase positively from the origin to the bottom of the image.

A.1.3 Pixel aspect ratio

Images shall be represented using pixels. The ratio of the horizontal and vertical dimensions of the pixels are given in or returned from function calls where needed.

A.1.4 Greyscale pixel depth

The greyscale precision of the pixel data shall be specified in terms of the pixel depth or the number of bits used to represent the greyscale value of a pixel. A pixel depth of 3 provides 8 levels of greyscale; a depth of 8 provides up to 256 levels of gray. For greyscale data, the minimum value that can be assigned to a "black" pixel shall be zero. The maximum value that can be assigned to a "white" pixel shall be the greyscale value with all of its bits of precision set to "1". However, the 'blackest" pixel in an image may have a value greater than "0" and the "whitest" pixel may have a value less than its maximum value. This implies that the maximum value for a "white" pixel with 5 bits of precision shall be 31 or less. The maximum value for the "whitest" pixel using 8 bits of precision shall be 255 or less. The pixel depth may range from 1 to 16 bits.

A.1.5 Colour space

The 24-bit RGB colour space where for every pixel, eight (8) bits will be used to represent each of the Red, Green, and Blue components. To achieve device-independence, the RGB values from the camera or scanner should be converted to values in a defined standard RGB space, such as sRGB, using the device's colour profile and colour management processing. Information regarding device profiling and colour management can be downloaded from the International Colour Consortium URL: http://www.color.org/.

A.1.6 Video interlacing

Interlaced video frames are not allowed for the streamed images. All interlacing shall be absent (not simply removed, but absent).

A.2 BioSFPI_BSFPImagePropertyID

Defines the UUID of the format of the BioSFPI_BSFPPropertySchema. When a BSFP claims to be conformant to this annex of this part of ISO/IEC 19784, it has to use a PropertySchema as defined within this annex of this part of ISO/IEC 19784. The value of this UUID will be used in the BFP Property ID element of the BFP schema as described in ISO/IEC 19784-1:2006 Clause 10.1.3.

```
#define    (BioSFPI_BSFPImagePropertyID, 0x4011dde0, 0x8008, 0x11de, 0xa393,
       0x0002a5d5c51b);
```

NOTE – This UUID 4011dde0-8008-11de-a393-0002a5d5c51b has been generated and registered in accordance with ISO/IEC 9834-8 using the generator provided by the International Telecommunication Union.

A.3 BioSFPI_BSFPImagePropertySchema

The BioSFPI_BSFPPropertySchema contains information about the function provider's capabilities, which are presented to the BioAPI component registry and in the related function calls. The function provider capabilities can differ from the capabilities of a biometric sensor unit. The capabilities useable by a BSP or BioAPI application are the intersection of the BSFP and biometric sensor unit capabilities.

```
typedef struct _biosfpi_bsfp_image_property_schema  {
       BioAPI_VERSION                      BSFPImageVersion;
       BioAPI_POWER_MODE                   SupportedPowerModes;
       BioAPI_BOOL                         DataTransferSupported;
       BioAPI_BOOL                         GetPacketsSupported;
       BioAPI_BIR_BIOMETRIC_DATA_FORMAT    *BDBFormatsSupportedList;
       uint32_t                            NumberOfElements;
       void                                *AdditionalParameters;
       uint32_t                            AdditionalParametersSize;
   } BioSFPI_BSFP_IMAGE_PROPERTY_SCHEMA, *BioSFPI_BSFP_IMAGE_PROPERTY_SCHEMA_PTR;
```

BSFPImageVersion – This value indicates the version of the SFPI. The version corresponding to this International Standard has an integer value of (decimal) 16, or (hex) 10, corresponding to a Major value of 1 and a Minor value of zero.

SupportedPowerModes – An enumeration that specifies the types of power modes the BSP and its attached devices (BioAPI_Units) will try to use. Handling of a power mode means to pass the related function call to the BioAPI Unit, when this supports the mode and to return the error code related to the result of the reaction of the BioAPI Unit.

DataTransferSupported – If true, the BFP supports calls of BioSFPI_DataTransfer.

GetPacketsSupported – If true, the BFP supports calls of BioSFPI_GetPackets.

BDBFormatsSupportedList – This a pointer to a list of pairs of format owner and format type identifying the BDB formats supported.

NumberOfElements – Gives the number of elements in the BDBFormatsSupportedList.

AdditionalParameters – Is a pointer to a data field which is described by the manufacturer of the BSFP or by other standards.

AdditionalParametersSize – Gives the size of the structure AdditionalParameters in octets.

NOTE – Any registered BDB formats (standard or vendor specific) can be used. Restriction of BDB formats can be done by profiling.

A.4 BioSFPI_UnitImagePropertyID

Defines the UUID of the format of the BioSFPI_UnitPropertySchema for imaging biometric sensor units. When an imaging biometric sensor unit claims to be conformant to this part of ISO/IEC 19784, it has to use a PropertySchema as defined within this part of ISO/IEC 19784.

```
#define (BioSFPI_UnitPropertyID, 0xa3f70a60, 0x800d, 0x11de, 0xba, 0xe3, 0x00, 0x02, 0xa5,
        0xd5, 0xc5, 0x1b);
```

NOTE – This UUID has been generated and registered in accordance with ISO/IEC 9834-8 using the generator provided by the International Telecommunication Union.

A.5 BioSFPI_UnitImagePropertySchema

The BioSFPI_UnitPropertySchema contains the information to be presented in the related function calls. Presenting an imaging biometric sensor unit this schema contains information to identify the unit itself (SensorUuid and SensorName) and capability information as well.

```
typedef struct _biosfpi_unit_image_property_schema  {
        BioAPI_UUID                             SensorType;
        BioAPI_UUID                             SensorUuid;
        BioAPI_STRING                           SensorName;
        BioAPI_STRING                           SensorSerialNumber;
        uint32_t                                MaxResponseTime;
        BioAPI_POWER_MODE                       SupportedPowerModes;
        uint32_t                                PixelsHeight;
        uint32_t                                PixelsWidth;
        uint32_t                                AspectRatio;
        uint32_t                                MonochromePixelDepth;
        uint32_t                                ColourPixelDepth;
        uint32_t                                Resolution;
        BioAPI_BOOL                             DataTransferSupported;
        BioAPI_BOOL                             GetPacketsSupported;
        BioAPI_BIR_BIOMETRIC_DATA_FORMAT        *BDBFormatsSupportedList;
        uint32_t                                NumberOfElements;
        uint32_t                                EstimatedDataRate;
        uint32_t                                MaxPacketsPerSecond;
        uint32_t                                MaxFramesPerSecond;
        uint32_t                                SuggestedMemoryForTransferSize;
    } BioSFPI_UNIT_IMAGE_PROPERTY_SCHEMA, *BioSFPI_UNIT_IMAGE_PROPERTY_SCHEMA_PTR;
```

SensorType –	The UUID of the type of the biometric sensor unit. The value is equal to the UUID defined in Clause A.2.
SensorUuid –	The UUID of the biometric sensor unit given by the vendor.
SensorName –	The name of the biometric sensor.
SensorSerialNumber –	A device identifier given by the manufacturer. Setting this value to zero is allowed and indicates that there is no device identifier supported.
MaxResponseTime –	The value defines the maximum time in milliseconds a caller of any function to the BioAPI Unit has to wait until the function returns. A '0' means that no information about the maximum time until the BioAPI Unit returns is given.
SupportedPowerModes –	An enumeration that specifies the types of power modes the BSP and its attached devices (BioAPI_Units) will try to use. Handling of a power mode means to pass

	the related function call to the BioAPI Unit, when this supports the mode and to return the error code related to the result of the reaction of the BioAPI Unit.
PixelsHeight –	The value defines the number of vertical pixels being supported by the biometric sensor unit.
PixelsWidth –	The value defines the number of horizontal pixels being supported by the biometric sensor unit.
AspectRatio –	The value defines the ratio of PixelsHeight to PixelsWidth. The ratio is calculated by the higher 16 bit of AspectRatio divided by the lower 16 bit. For example an AspectRatio of 1:1 uses a coding of 0x00010001.
MonochromePixelDepth –	The value defines the pixel depth for greyscale images. If this value is 0 the biometric sensor unit does not support greyscale images.
ColourPixelDepth –	The value defines the pixel depth for sRGB images. If this value is 0 the biometric sensor unit does not support colour images. The conversion from RGB to sRGB can be either done by the biometric sensor unit itself or by the related BSFP.
Resolution –	The value defines the number of pixels per cm of the active sensor area.
DataTransferSupported –	If true, the biometric sensor supports calls of BioSFPI_DataTransfer.
GetPacketsSupported –	If true, the biometric sensor supports calls of BioSFPI_GetPackets.
BDBFormatsSupportedList –	This a pointer to a list of pairs of format owner and format type identifying the BDB formats supported.
NumberOfElements –	Gives the number of elements in the BDBFormatsSupportedList.
EstimatedDataRate –	Gives the number of kilobytes per second on average from the device between an activation and subsequent de-activation of that interface. If this parameter is set to 0 no information is given.
MaxPacketsPerSecond –	Gives the maximum number of packets per second the BioAPI_Unit can create. This parameter helps the BSP to implement a sufficient memory management (e.g. multiple buffers). If this parameter is set to 0 no information is given.
MaxFramesPerSecond –	Gives the maximum number of frames per second the BioAPI_Unit can create in streaming mode (using BioSFPI_DataTransfer). If this parameter is set to 0 no information is given.
SuggestedMemoryForTransferSize –	Gives the SFPs best estimate of an optimal amount of memory for transfer using BioSFPI_GetPackets or BioSFPI_DataTransfer streaming data.

A.6 Image sensor functions

A.6.1 BioSFPI_GetImageParameters

```
BioAPI_RETURN BioAPI BioSFPI_GetImageParameters
    (BioAPI_UNIT_ID  UnitId,
     uint32_t        *PixelsHeight,
     uint32_t        *PixelsWidth,
     uint32_t        *Colours,
     uint32_t        *PixelDepth);
```

Description
This function is used to retrieve the actual biometric sensor parameter.

There is no requirement that the unit ID value provided by a BSP as input to this function be the same unit ID value that the framework provided to the BSP in the original call to BioSPI_BSPAttach (if any), provided that the two unit ID values identify the same BioAPI unit (see 8.2.4).

Parameters

UnitId (input) – The ID of the BioAPI Unit being addressed for getting the capture parameters.

PixelsHeight (output) – A pointer to retrieve the vertical size in pixels of the image.

PixelsWidth (output) – A pointer to retrieve the horizontal size in pixels of the image.

Colours (output) – A pointer to retrieve the colour mode of the biometric sensor unit. If set to 0 the biometric sensor unit will return greyscale data until advised otherwise. If set to 1 the biometric sensor unit will return sRGB data until reset.

PixelDepth (output) – A pointer to retrieve the number of levels of the colour channel(s). In case of a greyscale image it represents the number of levels of gray. In case of a sRGB image it represents the levels of each colour (red, green, blue). For the three colours the number of levels is equal in any case.

Return Value

A BioAPI_RETURN value indicating success or specifying a particular error condition. The value BioAPI_OK indicates success. All other values represent an error condition.

Errors

```
BioAPIERR_FUNCTION_NOT_SUPPORTED
```

A.6.2 BioSFPI_SetImageParameters

```
BioAPI_RETURN BioAPI BioSFPI_SetImageParameters
    (BioAPI_UNIT_ID  UnitId,
     uint32_t        PixelsHeight,
     uint32_t        PixelsWidth,
     uint32_t        Colours,
     uint32_t        PixelDepth);
```

Description
This function is used to set the biometric sensor parameter.

There is no requirement that the unit ID value provided by a BSP as input to this function be the same unit ID value that the framework provided to the BSP in the original call to BioSPI_BSPAttach (if any), provided that the two unit ID values identify the same BioAPI unit (see 8.2.4).

Parameters

UnitId (input) – The ID of the BioAPI Unit being addressed for setting the capture parameters.

PixelsHeight (input) – Sets the vertical size in pixels of the image to be retrieved. If the vertical size to be set differs from the default value of the biometric sensor and the new vertical size is not supported by the sensor the function call will be returned with BioAPIERR_FUNCTION_NOT_SUPPORTED.

PixelsWidth (input) – Sets the horizontal size in pixels of the image to be retrieved. If the horizontal size to be set differs from the default value of the biometric sensor and the new horizontal size is not supported by the biometric sensor the function call will be returned with BioAPIERR_FUNCTION_NOT_SUPPORTED.

Colours (input) – If set to 0 the biometric sensor is requested to return greyscale data until advised otherwise. If set to 1 the biometric sensor is requested to return sRGB data until reset. If the requested data format is not supported by the biometric sensor the function call will be returned with BioAPIERR_FUNCTION_NOT_SUPPORTED.

PixelDepth (input) – Defines as binary value the number of levels for the colour channel. In case of a greyscale image it represents the number of levels of gray. In case of a sRGB image it represents the levels of each colour (red, green, blue). For the three colours the number of levels is equal in any case. If the requested pixel depth is not supported by the biometric sensor the function call will be returned with BioAPIERR_FUNCTION_NOT_SUPPORTED.

Return Value

A BioAPI_RETURN value indicating success or specifying a particular error condition. The value BioAPI_OK indicates success. All other values represent an error condition.

Errors

```
BioAPIERR_FUNCTION_NOT_SUPPORTED
```

A.6.3 BioSFPI_Pause

```
BioAPI_RETURN BioAPI BioSFPI_Pause
    (BioAPI_UNIT_ID UnitId);
```

Description

This function pauses the retrieving of streaming data.

There is no requirement that the unit ID value provided by a BSP as input to this function be the same unit ID value that the framework provided to the BSP in the original call to BioSPI_BSPAttach (if any), provided that the two unit ID values identify the same BioAPI unit (see 8.2.4).

Parameters

UnitId (input) – The ID of the BioAPI Unit being addressed to pause until advised otherwise.

Return Value

A BioAPI_RETURN value indicating success or specifying a particular error condition. The value BioAPI_OK indicates success. All other values represent an error condition.

Errors

```
BioAPIERR_FUNCTION_NOT_SUPPORTED
```

A.6.4 BioSFPI_Play

```
BioAPI_RETURN BioAPI BioSFPI_Play
    (BioAPI_UNIT_ID  UnitId);
```

Description

This function restarts capturing streaming data.

There is no requirement that the unit ID value provided by a BSP as input to this function be the same unit ID value that the framework provided to the BSP in the original call to BioSPI_BSPAttach (if any), provided that the two unit ID values identify the same BioAPI unit (see 8.2.4).

Parameters

UnitId (input) – The ID of the BioAPI Unit being addressed to capture biometric streaming data.

Return Value

A BioAPI_RETURN value indicating success or specifying a particular error condition. The value BioAPI_OK indicates success. All other values represent an error condition.

Errors

```
BioAPIERR_FUNCTION_NOT_SUPPORTED
```

Annex B
(normative)

Signature sensor units

B.1 General

This annex describes biometric sensors that capture signature/sign data.

B.1.1 Bit and byte ordering

Signature shall use Big-Endian format. That is, the more significant bytes of any multibyte quantity are stored at lower addresses in memory than less significant bytes. The order for transmission shall also be the most significant byte first and least significant byte last. Within a byte, the order of transmission shall be the most significant bit first and the least significant bit last. All numeric values are fixed-length unsigned integer quantities.

B.1.2 Coordinate system

The coordinate system used to express the pen position shall be a three-dimensional Cartesian coordinate system. There are 3 channels defined for recording pen position data in the three-dimensional space. The x axis shall be the horizontal axis of the writing plane, with x coordinates increasing to the right. The y axis shall be the vertical axis of the writing plane, with y coordinates increasing upwards. The z axis shall be the axis perpendicular to the writing plane, with z coordinates increasing upwards out of the writing plane starting from 0. The horizontal and vertical pen position in the writing plane is recorded in the X and Y channels, respectively. The height of the pen above the writing plane is recorded in the Z channel.

The horizontal and vertical pen velocity in the writing plane is recorded in the VX and VY channels, respectively.

The horizontal and vertical pen acceleration in the writing plane is recorded in the AX and AY channels, respectively.

The T channel is defined for recording time data.

The DT channel is defined for recording time data relative to the previous sample.

The F channel is defined for recording pen forces (pressure) data.

The S channel is defined for recording whether the pen touches the writing plane or not. The data values shall be 0 in case of non-touching and 1 in case of touching.

There are 5 channels defined for recording pen orientation data, Az, El, TY, TY and R. Implementers may choose to use both azimuth and elevation or tilt angles. The third degree of freedom in orientation is defined as the rotation of the pen about its axis.

Inclusion of the X and Y channels is mandatory. Either the T channel or the DT channel shall be present, or uniform sampling (constant time difference between adjacent sample points) shall be indicated. Inclusion of the other channels is optional.

(see ISO/IEC 19794-7:2007)

B.1.3 Channel inclusion field

The channel inclusion field shall indicate the presence or absence of channels. The channel inclusion field shall consist of 2 octets. Each bit shall correspond to a channel as shown in Table B.1. A bit value of 1 shall encode the presence of the corresponding channel; a bit value of 0 shall encode the absence of the corresponding channel.

Table B.1 — Format of the channel inclusion field

Channel	Interpretation	Octet	Bit position
X	x coordinate (horizontal pen position)	1	8 (MSB)
Y	y coordinate (vertical pen position)	1	7
Z	z coordinate (height of pen above the writing plane)	1	6
VX	Velocity in x direction	1	5
VY	Velocity in y direction	1	4
AX	Acceleration in x direction	1	3
AY	Acceleration in y direction	1	2
T	Time	1	1 (LSB)
DT	Time difference	2	8 (MSB)
F	Pen tip force (pressure)	2	7
S	Tip switch state (touching/not touching the writing plane)	2	6
TX	Tilt along the x axis	2	5
TY	Tilt along the y axis	2	4
AZ	Azimuth angle of the pen (yaw)	2	3
EL	Elevation angle of the pen (pitch)	2	2
R	Rotation (rotation about the pen axis)	2	1 (LSB)

As an example, Figure B.1 shows the channel inclusion field for signature/sign data including the channels X, Y, T, F, S, Az, El and R.

Figure B.1 — Example of a channel inclusion field

Octet 1								Octet 2							
1	1	0	0	0	0	0	1	0	1	1	0	0	1	1	1

(see ISO/IEC 19794-7:2007)

B.1.4 Sequence of sample points

For each sample point, the field shall begin with a value for the mandatory X channel, followed by a value for the mandatory Y channel and a sequence of optional channel values as indicated by the channel inclusion field. The order of the channel values is determined by the order of indicated inclusion within the channel inclusion field (Table B.1).

For the Z, T, DT, F, Az, El, and R channels, integer values in the range from 0 to 65 535 are allowed. These values shall be encoded in 2 octets as unsigned integers.

For the X, Y, VX, VY, AX, AY, TX, and TY channels, integer values in the range from –32 768 to 32 767 are allowed. These values shall be encoded in 2 octets as unsigned integers, after adding 32 768 to each value. Hence, for non-negative numbers, bit 8 of the most significant octet has the value 1; for negative numbers, bit 8 of the most significant octet has the value 0.

For the S channel, integer values in the range from 0 to 1 are allowed. These values shall be encoded in one octet as unsigned integers.

(see ISO/IEC 19794-7:2007)

B.1.5 Sample Rate

The sample Rate shall indicate the frequency of sampling used by signature biometric sensors.

B.1.6 ChannelDescriptions

The UnitSignaturePropertySchema contains information called channel description. As each physical scanner device may have different value outputs e.g. at the same force or rotation angle the unit property schema needs to hold information about e.g. minimum and maximum values or calibration factors.

The content of the ChannelDescriptions is similar to ISO/IEC 19794-7. The ChannelInclusion field indicates the presence of each channel description in descending order from msb to lsb (see B.1.3 and ISO/IEC 19794-7:2007, Clause 7.2.6.1).

Each ChannelDescription (when present) shall start with a channel description preamble as prescribed in ISO/IEC 19794-7:2007, Clause 7.2.6.2. The preamble will be followed by the values as indicated in the preamble. The value coding and data type shall be exactly as in ISO/IEC 19794-7.

As a result the ChannelDescriptions are a data structure of variable length which is stored in the BSFP_UnitSignaturePropertySchema.

B.2 BioSFPI_BSFPSignaturePropertyID

Defines the UUID of the format of the BioSFPI_BSFPSignaturePropertySchema. When a BSFP claims to be conformant to this annex of this part of ISO/IEC 19784, it has to use a PropertySchema as defined within this annex of this part of ISO/IEC 19784. The value of this UUID will be used in the BFP Property ID element of the BFP schema as described in ISO/IEC 19784-1:2006, Clause 10.1.3.

```
#define (BioSFPI_BSFPSignaturePropertyID, 0xcd02bde0, 0x800e, 0x11de, 0xb83, 0x26,
                                  0x00, 0x02, 0xa5, 0xd5, 0xc5, 0x1b);
```

NOTE – This UUID cd02bde0-800e-11de-8326-0002a5d5c51b has been generated and registered in accordance with ISO/IEC 9834-8 using the generator provided by the International Telecommunication Union.

B.3 BioSFPI_BSFPSignaturePropertySchema

The BioSFPI_BSFPSignaturePropertySchema contains information about the function provider's capabilities, which are presented to the BioAPI component registry and in the related function calls. The function provider capabilities can differ from the capabilities of a biometric sensor. The capabilities useable by a BSP or BioAPI application are the intersection of the BSFP and biometric sensor capabilities.

```
typedef struct _biosfpi_bsfp_signature_property_schema {
       BioAPI_VERSION                       BSFPSignatureVersion;
       BioAPI_POWER_MODE                    SupportedPowerModes;
       BioAPI_BOOL                          DataTransferSupported;
       BioAPI_BOOL                          GetPacketsSupported;
       BioAPI_BIR_BIOMETRIC_DATA_FORMAT     *BDBFormatsSupportedList;
       uint32_t                             NumberOfElements;
       void                                 *AdditionalParameters;
       uint32_t                             AdditionalParametersSize;
    } BioSFPI_BSFP_SIGNATURE_PROPERTY_SCHEMA, *BioSFPI_BSFP_SIGNATURE_PROPERTY_SCHEMA_PTR;
```

BSFPSignatureVersion –	This value indicates the version of the SFPI. The version corresponding to this International Standard has an integer value of (decimal) 16, or (hex) 10, corresponding to a Major value of 1 and a Minor value of zero.
SupportedPowerModes –	An enumeration that specifies the types of power modes the BSP and its attached devices (BioAPI_Units) will try to use. Handling of a power mode means to pass the related function call to the BioAPI Unit, when this supports the mode and to return the error code related to the result of the reaction of the BioAPI Unit.
DataTransferSupported –	If true, the BFP supports calls of BioSFPI_DataTransfer.
GetPacketsSupported –	If true, the BFP supports calls of BioSFPI_GetPackets.
BDBFormatsSupportedList –	This a pointer to a list of pairs of format owner and format type identifying the BDB formats supported.
NumberOfElements –	Gives the number of elements in the BDBFormatsSupportedList.
AdditionalParameters –	Is a pointer to a data field which is described by the manufacturer of the BSFP or by other standards.
AdditionalParametersSize –	Gives the size of the structure AdditionalParameters in bytes.

NOTE – Any registered BDB formats (standard or vendor specific) can be used. Restriction of BDB formats can be done by profiling.

B.4 BioSFPI_UnitSignaturePropertyID

Defines the UUID of the format of the BioSFPI_UnitSignaturePropertySchema for signature biometric sensors. When a BSFP supporting a digital tablet biometric sensor claims to be conformant to this part of ISO/IEC 19784, it has to use a PropertySchema as defined within this part of ISO/IEC 19784.

```
#define    (BioSFPI_UnitSignaturePropertyID, 0x84a898c0, 0x800f, 0x11de, 0xbd81,
       0x0002a5d5c51b);
```

NOTE – This UUID 84a898c0-800f-11de-bd81-0002a5d5c51b has been generated and registered in accordance with ISO/IEC 9834-8 using the generator provided by the International Telecommunication Union.