INTERNATIONAL STANDARD

ISO/IEC 19784-1

First edition
2006-05-01

AMENDMENT 3
2010-08-01

Information technology — Biometric application programming interface —

Part 1: BioAPI specification

AMENDMENT 3: Support for interchange of certificates and security assertions, and other security aspects

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

Partie 1: Spécifications BioAPI

AMENDEMENT 3: Support pour interéchange de certificats et de déclarations de sécurité, et autres aspects de sécurité

Reference number
ISO/IEC 19784-1:2006/Amd.3:2010(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 3 to ISO/IEC 19784-1:2006 was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information technology*, Subcommittee SC 37, *Biometrics*.

This amendment to ISO/IEC 19784-1 defines a new version 2.2 of BioAPI which adds support for biometric fusion and security assertions to ISO/IEC 19784-1. It extends the API and the SPI of BioAPI by specifying new functions and new values for existing data types.

ISO/IEC 19784-1:2006 provides no direct support for biometric fusion. In addition, the use of FARs in the representation of matching scores is not suitable, in general, for performing score-level fusion (although it does allow some limited forms of fusion). This amendment adds support of biometric fusion to ISO/IEC 19784-1.

Information technology — Biometric application programming interface —

Part 1:
BioAPI specification

AMENDMENT 3: Support for interchange of certificates and security assertions, and other security aspects

1) General amendment items

1-1) Add the following at the end of the first paragraph of the Foreword:

", BioAPI 2.1, and BioAPI 2.2"

1-2) Replace the last paragraph of the Foreword with the following:

This is the first ISO/IEC standard on BioAPI. Previous versions were published by ANSI and the BioAPI Consortium. As the last official non-ISO release was designated Version 1.1, the version specified in this part of ISO/IEC 19784-1 is designated Version 2.0 onwards. This is to distinguish the versions of BioAPI products in the marketplace.

1-3) Replace the first paragraph of the Introduction with the following:

This part of ISO/IEC 19784 provides a high-level generic biometric authentication model suited to most forms of biometric technology. Support for multimodal biometrics and security assertions is also provided.

2) Amendment items for interchange of certificates and security assertions, and other security aspects

2-1) Replace the last paragraph of the Scope with the following:

This part of ISO/IEC 19784 specifies a version of the BioAPI specification that is defined to have a version number described as Major 2, Minor 0, or version 2.0. It also specifies a version number described as Major 2, Minor 1, or version 2.1 that provides an enhanced Graphical User Interface. It also specifies a version number described as Major 2, Minor 2, or version 2.2 that provides features supporting fusion and security. Some clauses and sub-clauses apply only to one of these versions, some to two or more. This is identified at the head of the relevant clauses and sub-clauses.

2-2) Remove the amended paragraph in Amd.2 after the last paragraph of the Scope and add the following paragraph after the last amended NOTEs of the Scope:

Conformance requirements are specified in Clause 2.

2-3) Add the following documents to Clause 3:

ISO/IEC 19785-4:2010, *Information technology — Common Biometric Exchange Formats Framework — Part 4: Security block format specifications*

ISO/IEC 24761:2009, *Information technology — Security techniques — Authentication context for biometrics*

2-4) Add the following term and definition before 4.1:

4.0
ACBio instance
report generated by a biometric processing unit (BPU) compliant to ISO/IEC 24761 to show the validity of the result of one or more subprocesses executed in the BPU

[ISO/IEC 24761]

2-5) Add the following term and definition after 4.3:

4.3bis
authentication context for biometrics
ACBio
International Standard that specifies the form and encoding of ACBio instances

[ISO/IEC 24761]

2-6) Add the following terms and definitions after 4.5:

4.5bis
biographic data (BioAPI 2.2)
non-biometric data that potentially affects a biometric operation

4.5ter
biographic BIR (BioAPI 2.2)
non-biometric BIR that potentially affects a biometric operation

2-7) Add the following term and definition after 4.10:

4.10bis
biometric processing unit
BPU
entity that executes one or more subprocesses that perform a biometric verification at a uniform level of security

[ISO/IEC 24761]

NOTE A sensor, a smart card, and a comparison device are examples of BPUs.

2-8) Add the following term and definition after 4.14:

4.14bis
BPU IO Index
integer assigned to each biometric data stream between BPUs by the subject, such as software, which utilizes the function of the BPU so that the validator can reconstruct the data flow among BPUs

[ISO/IEC 24761]

2-9) Add the following term and definition after 4.16:

4.16bis
decision BIR (BioAPI 2.2)
BIR which contains decision in the BDB

2-10) Replace the terms of 4.22 with the following terms:

4.22
score
score data
scoring

2-11) Add the following term and definition after 4.22:

4.22bis
score BIR (BioAPI 2.2)
BIR which contains score in the BDB

2-12) Add the following term and definition after 4.22bis:

4.22ter
secure BioAPI (BioAPI 2.2)
BioAPI API and SPI interfaces that include the security features defined for version 2.2 of BioAPI

2-13) Replace Clause 5 with the following:

ACBio – Authentication Context for Biometrics

API – Application Programming Interface

BDB – Biometric Data Block

BFP – BioAPI Function Provider

BIR – Biometric Information Record

BPU – Biometric Processing Unit

BSP – Biometric Service Provider

CBEFF – Common Biometric Exchange Formats Framework

FMR – False Match Rate

FPI – Function Provider Interface

GUI – Graphical User Interface

ID – Identity/Identification/Identifier

IRI – Internationalized Resource Identifier (see RFC 3987)

MAC – Message Authentication Code

MOC – Match on Card

PID – Product ID

SB – Security Block

SBH – Standard Biometric Header

NOTE This term and abbreviation is imported from ISO/IEC 19785-1.

SPI – Service Provider Interface

UUID – Universally Unique Identifier

2-14) Replace 6.6.6 as follows:

6.6.6 On BioAPI_BSPAttach/BioAPI_BSPAttachSecure (BioAPI 2.2 or greater), the application is required to select at most one BioAPI Unit of each category that is currently in an "inserted" state (or to select BioAPI_DONT_CARE) and is managed by that BSP or by an associated BFP. The BSP then either accesses that BioAPI Unit (for directly managed BioAPI Units), or else interacts with the associated BFP in order to access that BioAPI Unit.

2-15) Add the following text after 7.1:

7.1bis BioAPI_ACBio_PARAMETERS (BioAPI 2.2)

7.1bis.1 Structure giving information which is used to generate ACBio instances

```
typedef struct bioapi_acbio_parameters {

    uint32_t Challenge[4];

    uint32_t *InitialBPUIOIndexOutput;

    uint32_t *SupremumBPUIOIndexOutput;

} BioAPI_ACBio_PARAMETERS;
```


7.1bis.2 Definitions

Challenge – Challenge from the validator of a biometric verification when ACBio is used. This value shall be set to the field `controlValue` of type `ACBioContentInformation` in ACBio instances.

InitialBPUIOIndexOutput – The initial value of BPU IO index which is to be assigned to the output from the BioAPI Unit, BFP, or BSP when the ACBio instances are generated. The range between *InitialBPUIOIndexOutput* and *SupremumBPUIOIndexOutput* shall be divided into the number of BSP Units and BFPs which are attached to the BSP, and assigned to the BSP Units and BSPs.

SupremumBPUIOIndexOutput – The supremum of BPU IO indexes which are to be assigned to the output from the BioAPI Unit, BFP, or BSP when the ACBio instances are generated.

7.1ter BioAPI_ASN1_BIR (BioAPI 2.2)

A container for biometric data, (or non-biometric data) that may affect a biometric operation, encoded in ASN.1 DER. The BioAPI_ASN1_BIR structure is used when a BioAPI API with security functionality is used. The BioAPI_ASN1_BIR structure associates a length, in bytes, with the address of an arbitrary block of contiguous memory which contains an ASN.1 encoded BIR of type `BioAPI-BIR`, specified in Annex E. The ASN.1 encoded BIR consists of an SBH of type `BioAPIBIRHeader`, a BDB of type `BiometricData`, and an SB of type `CBEFFSecurityBlock`. The BDB may contain raw sample data, partially processed (intermediate) data, completely processed data, score data (resulting from a matching or fusion operation), decision data (resulting from a decision or fusion operation), or biographic data which can be provided as input to a biometric operation to modify its behavior. The BioAPI_ASN1_BIR may be used to enroll a user (thus being stored persistently), or may be used to verify or identify a user (thus being used transiently). It may also be stored and used later to provide feedback to subsequent biometric operations. Type BioAPI_ASN1_BIR is an alias of type BioAPI_DATA.

NOTE The ASN.1 type `BioAPI-BIR` corresponds to the C structured type BioAPI_BIR element by element.

```
typedef BioAPI_DATA BioAPI_ASN1_BIR;
```

7.1quater BioAPI_ASN1_ENCODED (BioAPI 2.2)

A container for ASN.1 DER encoded data. The BioAPI_ASN1_ENCODED structure is used to express information about cryptographic keys. The BioAPI _ASN1_ ENCODED structure associates a length, in bytes, with the address of an arbitrary block of contiguous memory which contains an ASN.1 encoded data. Type BioAPI_ASN1_ENCODED is an alias of type BioAPI_DATA.

```
typedef BioAPI_DATA BioAPI_ASN1_ENCODED;
```

2-16) In 7.4.1, modify the paragraph as follows:

A container for biometric data, or non-biometric data that may affect a biometric operation. A BioAPI_BIR consists of a BioAPI_BIR_HEADER, a BDB, and an optional SB. The BDB may contain raw sample data, partially processed (intermediate) data, completely processed data, score data (resulting from a matching or fusion operation), decision data (resulting from a decision or fusion operation), or biographic data which can be provided as input to a biometric operation to modify its behavior. The BioAPI_BIR may be used to enroll a user (thus being stored persistently), or may be used to verify or identify a user (thus being used transiently). It may also be stored and used later to provide feedback to subsequent biometric operations.

2-17) In 7.9.1, modify a) with the following:

a) it identifies the type of biometric sample (raw, intermediate, processed, score data, decision data or biographic data) that is contained in the BDB;

2-18) Replace 7.9.4 as follows:

7.9.4 The 'index' flag shall be set if an index is present in the BIR header and not set if no index is present in the BIR header.

```
typedef uint8_t BioAPI_BIR_DATA_TYPE;

   #define BioAPI_BIR_DATA_TYPE_RAW             (0x01)
   #define BioAPI_BIR_DATA_TYPE_INTERMEDIATE    (0x02)
   #define BioAPI_BIR_DATA_TYPE_PROCESSED       (0x04)
   #define BioAPI_BIR_DATA_TYPE_SCORE           (0x08)
   #define BioAPI_BIR_DATA_TYPE_DECISION        (0x09)
   #define BioAPI_BIR_DATA_TYPE_BIOGRAPHIC      (0x0A)
   #define BioAPI_BIR_DATA_TYPE_ENCRYPTED       (0x10)
   #define BioAPI_BIR_DATA_TYPE_SIGNED          (0x20)
```

NOTE 1 The BioAPI BIR Data Type corresponds to a combination of the "CBEFF_BDB_processed_level", "CBEFF_BDB_encryption_options", and "CBEFF_BIR_integrity_options" in ISO/IEC 19785-1.

NOTE 2 BioAPI_BIR_DATA_TYPE_DECISION (BioAPI 2.2 or greater) and BioAPI_BIR_DATA_TYPE_BIOGRAPHIC (BioAPI 2.2 or greater) have two bits on while others have only one bit on.

NOTE 3 BioAPI_BIR_DATA_TYPE_SCORE is used in BioAPI 2.2 or greater.

2-19) Replace the text of 7.10 with the following:

A handle to refer to a BioAPI BIR or an ASN.1 encoded BIR that exists within a BSP.

2-20) Replace 7.12.1 with the following:

7.12.1 A value which defines the purpose(s) or use(s) for which the BioAPI BIR is intended (when used as an input to a BioAPI function) or suitable (when used as an output from a BioAPI function or within the BIR header).

```
typedef uint8 BioAPI_BIR_PURPOSE;

   #define BioAPI_PURPOSE_VERIFY (1)
   #define BioAPI_PURPOSE_IDENTIFY (2)
   #define BioAPI_PURPOSE_ENROLL (3)
   #define BioAPI_PURPOSE_ENROLL_FOR_VERIFICATION_ONLY (4)
   #define BioAPI_PURPOSE_ENROLL_FOR_IDENTIFICATION_ONLY (5)
   #define BioAPI_PURPOSE_AUDIT (6)
```

```
#define BioAPI_PURPOSE_DECIDE (7)

#define BioAPI_NO_PURPOSE_AVAILABLE (0)
```

NOTE 1 The condition NO VALUE AVAILABLE is indicated by setting the value to zero. This value is used only for BIRs that are not originally generated by a BioAPI BSP, but originate from another source and have been transformed into a BioAPI BIR. BSPs shall not use this value.

NOTE 2 BioAPI_PURPOSE_DECIDE is used in BioAPI 2.2 or greater.

2-21) In 7.12.3, replace e) and f) as follows:

e) The BioAPI_Process, BioAPI_CreateTemplate, BioAPI_ProcessWithAuxBIR (BioAPI 2.1 or less), BioAPI_ProcessUsingAuxBIRs (BioAPI 2.2 or greater), BioAPI_Decide (BioAPI 2.2 or greater), and BioAPI_Fuse (BioAPI 2.2 or greater) functions do not have Purpose as an input parameter, but read the Purpose field from the BIR header of the input BIR.

f) The BioAPI_Process and BioAPI_ProcessUsingAuxBIRs functions may accept as input any intermediate BIR with a Purpose of BioAPI_PURPOSE_VERIFY or BioAPI_PURPOSE_IDENTIFY, and shall output only BIRs with the same purpose as the input BIR.

2-22) In 7.12.3, add i) as follows:

i) All score BIRs and decision BIRs must have the Decide purpose. Biographic BIRs may have any purpose. No other types of BIRs may have the Decide purpose.

2-23) Add the following text after 7.25:

7.25bis BioAPI_ENCRYPTION_ALG (BioAPI 2.2)

Identifies encryption algorithm supported by a BioAPI Unit. This BioAPI type shall be the XML value notation of ASN.1 identifier (see ISO/IEC 8824-1) assigned to encryption algorithms. Example to specify AES with 128 bit keys in CBC mode, the char would be “2.16.840.1.101.3.4.1.2” which is the XML value notation for.

```
typedef char *BioAPI_ENCRYPTION_ALG;

   #define BioAPI_ENCRYPTION_ALG_NOT_SUPPORTED NULL
```

7.25ter BioAPI_ENCRYPTION_INFO (BioAPI 2.2)

7.25ter.1 Structure giving information of the cryptographic algorithm and key(s) of a BioAPI Unit or a biometric application which is used to encrypt/decrypt biometric data

```
typedef struct bioapi_encryption_info {

    BioAPI_ENCRYPTION_ALG ENCAlg;

    BioAPI_KEY_INFO *ENCKeyInfo;

} BioAPI_ENCRYPTION_INFO;
```

7.25ter.2 Definitions

ENCAlg – Encryption algorithm.

ENCKeyInfo – An array of key information for encryption.

2-24) In 7.27, replace NOTE as follows:

NOTE: It may be impossible to mask an INSERT event coming from an attach session of a BSP, because the event may occur just after a ***BioAPI_BSPLoad*** call, before any ***BioAPI_EnableEvents*** call has had any chance to be processed. This is because ***BioAPI_EnableEvents*** requires a handle which is provided by ***BioAPI_BSPAttach***/***BioAPI_BSPAttachSecure*** (BioAPI 2.2 or greater), and ***BioAPI_BSPAttach***/***BioAPI_BSPAttachSecure*** (BioAPI 2.2 or greater) itself shall follow ***BioAPI_BSPLoad***. An INSERT event will be raised by the BSP on the ***BioAPI_BSPLoad*** call if a BioAPI Unit is already "inserted", and this event will reach the application before the application can call ***BioAPI_EnableEvents***.

2-25) Replace 7.38 with the following

A unique identifier, returned on ***BioAPI_BSPAttach***/***BioAPI_BSPAttachSecure*** (BioAPI 2.2 or greater), that identifies an attached BioAPI BSP session.

```
typedef uint32_t BioAPI_HANDLE;
```

2-26) Add the following text after 7.38:

7.38bis BioAPI_HASH_ALG (BioAPI 2.2)

Identifies the hash algorithm supported by a BioAPI Unit, which is used in the generation of ACBio instance. This BioAPI type shall be the XML value notation of ASN.1 identifier (see ISO/IEC 8824-1) assigned to hash algorithms. Example to specify the SHA-1 hash algorithm, the char would be “1.3.14.3.2.26” which is the XML value notation for.

```
typedef char *BioAPI_HASH_ALG;

   #define BioAPI_HASH_ALG_NOT_SUPPORTED NULL
```

2-27) Add the following text after 7.45:

7.45bis BioAPI_KEY_INFO (BioAPI 2.2)

7.45bis.1 Union giving information of cryptographic key of a BioAPI Unit or a biometric application which is used to encrypt/decrypt biometric data or to generate/validate MAC of BIR.

```
typedef union bioapi_key_info {

    BioAPI_KEY_TRANSPORT KTInfo;

    BioAPI_ASN1_ENCODED KEKInfo;

} BioAPI_KEY_INFO;
```


7.45bis.2 Definitions

KTInfo – Key information when key management technique of key transport is used

KEKInfo – Key information when key management technique of previously distributed symmetric key-encryption keys is used

NOTE: For details of key management, see RFC 3852.

7.45ter BioAPI_KEY_TRANSPORT (BioAPI 2.2)

7.45ter.1 Structure giving information of cryptographic key of a BioAPI Unit or a biometric application, which is used to encrypt/decrypt biometric data or to generate/.validate MAC of BIR, when key management technique of key transport is used.

```
typedef struct bioapi_key_transport {

    BioAPI_ASN1_ENCODED IssuerAndSerialNumber;

    BioAPI_ASN1_ENCODED Certificate;

} BioAPI_KEY_TRANSPROT;
```

7.45ter.2 Definitions

IssuerAndSerialNumber – ASN.1 encoded data of ASN.1 type `IssuerAndSerialNumber` which contains the information of issuer and serial number of the X.509 certificate for the public key.

Certificate – ASN.1 encoded data of ASN.1 type `Certificate` which contains X.509 certificate of the public key.

NOTE: For details of the definitions of the types **`IssuerAndSerialNumber`** and **`Certificate`**, see RFC 3852.

7.45quater BioAPI_MAC_ALG (BioAPI 2.2)

Identifies the MAC algorithm supported by a BioAPI Unit. Example to specify the HMAC algorithm with SHA-1, the char would be “1.3.6.1.5.5.8.1.2” which is the XML value notation for.

```
typedef char *BioAPI_MAC_ALG;

   #define BioAPI_MAC_ALG_NOT_SUPPORTED NULL
```

7.45quinquies BioAPI_MAC_INFO (BioAPI 2.2)

7.45quinquies.1 Structure giving information of the MAC algorithm and key(s) of a BioAPI Unit or a biometric application which is used to generate/validate MAC of BIR

```
typedef struct bioapi_mac_info {

    BioAPI_MAC_ALG MACAlg;

    BioAPI_KEY_INFO *MACKeyInfo;

} BioAPI_MAC_INFO;
```

7.45quinquies.2 Definitions

MACAlg – MAC algorithm.

MACKeyInfo – An array of key information for message authentication code.

2-28) In 7.46, add the following masks:

```
#define BioAPI_PROCESSUSINGAUXBIRS        (0x01000000)
#define BioAPI_VERIFYMATCHUSINGAUXBIRS    (0x02000000)
#define BioAPI_DECIDE                     (0x04000000)
#define BioAPI_FUSE                       (0x08000000)
#define BioAPI_SECURITY                   (0x10000000)
```

2-29) In 7.47, modify the description about BioAPI_PAYLOAD as follows:

If set, the BSP supports payload carry (accepts a payload during ***BioAPI_Enroll*** or ***BioAPI_CreateTemplate*** and returns the payload upon successful ***BioAPI_Verify***, ***BioAPI_VerifyMatch***, or ***VerifyMatchUsingAuxBIRs*** (BioAPI 2.2 or greater)).

2-30) In 7.47, modify the description about BioAPI_ADAPTATION as follows:

If set, the BSP supports BIR adaptation in the return parameters of a ***Verify***, ***VerifyMatch***, or ***VerifyMatchUsingAuxBIRs*** (BioAPI 2.2 or greater) operation.

2-31) Add the following text after 7.50:

7.50bis BioAPI_SECURITY_OPTIONS_MASK (BioAPI 2.2)

A mask that indicates what security options are supported by the BioAPI Unit.

```
typedef uint32_t BioAPI_SECURITY_OPTIONS_MASK;
#define BioAPI_ENCRYPTION (0x00000001)
```

If set, indicates that the BioAPI Unit supports encryption.

```
#define BioAPI_MAC (0x00000002)
```

If set, indicates that the BioAPI Unit supports MAC generation.

```
#define BioAPI_DIGITAL_SIGNATURE (0x00000004)
```

If set, indicates that the BioAPI Unit supports digital signature.

```
#define BioAPI_ACBio_GENERATION_WITH_MAC (0x00000010)
```

If set, indicates that the BioAPI Unit supports ACBio generation using MAC.

```
#define BioAPI_ACBio_GENERATION_WITH_DIGITAL_SIGNATURE (0x00000020)
```

If set, indicates that the BioAPI Unit supports ACBio generation using digital signature.

7.50ter BioAPI_SECURITY_PROFILE (BioAPI 2.2)

7.50ter.1 Structure giving information of the cryptographic algorithms and keys of a BioAPI Unit or a biometric application which is used to encrypt/decrypt biometric data or to generate/validate the MAC or digital signature of the BIR and also giving the information of the hash algorithm, the information about the MAC generation, and the digital signature used in ACBio generation. When this structure is used in the structure of BioAPI_UNIT_SCHEMA (BioAPI 2.2) as the output parameter of ***BioAPI_QueryUnits***, the parameters in this structure indicate the information supported in the BioAPI Unit. On the other hand, when this structure is used as the parameter of ***BioAPI_BSPAttachSecure***, the parameters in this structure indicates the information which is to be used in the execution of security operations.

```
typedef struct bioapi_security_profile {

    BioAPI_SECURITY_OPTIONS_MASK SupportedSecOption;

    BioAPI_ENCRYPTION_INFO **ENCInfo;

    BioAPI_MAC_INFO **MACInfo;

    BioAPI_DIGITAL_SIGNATURE_ALG *SIGNAlg;

    BioAPI_OPERATIONS_MASK ACBioOption;

    BioAPI_HASH_ALG **HASHAlgForACBio;

    BioAPI_MAC_INFO **MACInfoForACBio;

    BioAPI_DIGITAL_SIGNATURE_ALG *SIGNAlgForACBio;

} BioAPI_SECURITY_PROFILE;
```

7.50ter.2 Definitions

SupportedSecOption – A mask which indicates which security options are supported or to be executed by the BSP Unit.

ENCInfo – Encryption information used in the encryption of the BDB.

MACInfo – MAC information used to keep the integrity of the BIR.

SIGNAlg – Digital signature algorithm used to keep the integrity of the BIR.

ACBioOption – A mask which indicates which security options of MAC or digital signature are supported or to be executed by the BSP Unit.

HASHAlgForACBio – Hash algorithm used to generate ACBio instances.

MACInfoForACBio – MAC information used to generate ACBio instances.

SIGNAlgForACBio – Digital signature algorithm used to generate ACBio instances.

7.50quater BioAPI_DIGITAL_SIGNATURE_ALG (BioAPI 2.2)

Identifies the digital signature algorithm supported by a BioAPI Unit. This BioAPI type shall be the XML value notation of ASN.1 identifier (see ISO/IEC 8824-1) assigned to digital signature algorithms. Example to specify the digital signature algorithm using SHA1 with RSA according to ISO/IEC 9796-2, the char would be “1.3.36.3.4.3.2.1” which is the XML value notation for.

```
typedef char *BioAPI_DIGITAL_SIGNATURE_ALG;

#define BioAPI_DIGITAL_SIGNATURE_ALG_NOT_SUPPORTED NULL
```

2-32) Add the following text after 7.55:

7.55bis BioAPI_UNIT_SCHEMA (BioAPI 2.2)

7.55bis.1 A BioAPI Unit schema indicates the specific characteristics of the BioAPI Unit.

```
typedef struct bioapi_unit_schema {
        BioAPI_UUID  BSPUuid;
        BioAPI_UUID  UnitManagerUuid;
        BioAPI_UNIT_ID  UnitId;
        BioAPI_CATEGORY  UnitCategory;
        BioAPI_UUID  UnitProperties;
        BioAPI_STRING  VendorInformation;
        uint32_t  SupportedEvents;
        BioAPI_UUID  UnitPropertyID;
        BioAPI_DATA  UnitProperty;
        BioAPI_STRING  HardwareVersion;
        BioAPI_STRING  FirmwareVersion;
        BioAPI_STRING  SoftwareVersion;
        BioAPI_STRING  HardwareSerialNumber;
        BioAPI_BOOL  AuthenticatedHardware;
        uint32_t  MaxBSPDbSize;
        uint32_t  MaxIdentify;
        BioAPI_SECURITY_PROFILE *SecurityProfile;
    } BioAPI_UNIT_SCHEMA;
```

NOTE: The BioAPI Unit Schema is used as a function parameter (by ***BioAPI_QueryUnits*** and ***BioAPI_EventHandler***), but is not stored in the component registry.

7.55bis.2 Definitions

BSPUuid – UUID of the BSP supporting this BioAPI Unit.

UnitManagerUuid – UUID of the software component directly managing the BioAPI Unit (may be either the BSP itself or a BFP).

UnitId – ID of the BioAPI Unit during this attach session. This will be created by the BSP uniquely.

UnitCategory – Defines the category of the BioAPI Unit.

UnitProperties – UUID indicating a set of properties of the BioAPI Unit. The indicated set can either be specified by each vendor or follow a related standard.

VendorInformation – Contains vendor proprietary information.

SupportedEvents – A mask indicating which types of events are supported by the hardware.

UnitPropertyID – UUID of the format of the following Unit property structure.

UnitProperty – Address and length of a memory buffer containing the Unit property describing the BioAPI Unit. The format and content of the Unit property can either be specified by the vendor or be specified in a related standard.

HardwareVersion – A NUL-terminated string containing the version of the hardware. Empty if not available.

FirmwareVersion – A NUL-terminated string containing the version of the firmware. Empty if not available.

SoftwareVersion – A NUL-terminated string containing the version of the software. Empty if not available.

HardwareSerialNumber – A NUL-terminated string containing the vendor defined unique serial number of the hardware component. Empty if not available.

AuthenticatedHardware – A boolean value that indicates whether the hardware component has been authenticated.

MaxBSPDbSize – Maximum size database supported by the BioAPI Unit. If zero, no database exists.

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

SecurityProfile – Security profile of the BioAPI Unit.

2-33) In 8.1.5.1, replace the seventh and eighth paragraph as follows:

There is a "use count" on the establishment of event handlers; they have to be de-established (by ***BioAPI_BSPUnload***) as many times as they were established. When a BSP is loaded (***BioAPI_BSPLoad***), it shall raise an "insert" event immediately for each present BioAPI Unit. This will indicate to the biometric application that it can go ahead and do a ***BioAPI_BSPAttach***/***BioAPI_BSPAttachSecure*** (BioAPI 2.2 or greater). 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.

This function shall only be called if there is at least one call to ***BioAPI_Init*** for which a corresponding call to ***BioAPI_Terminate*** has not yet been made. The function ***BioAPI_BSPAttach***/***BioAPI_BSPAttachSecure*** (BioAPI 2.2 or greater) can be invoked multiple times for each call to ***BioAPI_BSPLoad***.

2-34) In 8.1.6.1, replace the fourth paragraph as follows:

This function should not be called by the application if there is a call to ***BioAPI_BSPAttach***/***BioAPI_BSPAttachSecure*** (BioAPI 2.2 or greater) for which a corresponding call to ***BioAPI_BSPDetach*** (for a given BSP handle) has not yet been made. However, should this function be called while the BSP is still attached, then for each call to ***BioAPI_BSPAttach***/***BioAPI_BSPAttachSecure*** (BioAPI 2.2 or greater) without a corresponding call to ***BioAPI_BSPDetach***, the actions relative to the missing corresponding call to ***BioAPI_BSPDetach*** shall be implicitly performed by the BioAPI Framework (as though the corresponding function had been called at that time), followed by the actions relative to ***BioAPI_BSPUnload***, (i.e., the Framework will detach the BSP prior to unloading it).

2-35) Add the following text after 8.1.7:

8.1.7bis BioAPI_BSPAttachSecure (BioAPI 2.2)

```
BioAPI_RETURN BioAPI BioAPI_BSPAttachSecure
    (const BioAPI_UUID *BSPUuid,
    BioAPI_VERSION Version,
    const BioAPI_ACBio_PARAMETERS *ACBioParameters,
    const BioAPI_UNIT_LIST_ELEMENT *UnitList,
    const BioAPI_SECURITY_PROFILE *SecurityProfileList,
    uint32_t NumUnits,
    BioAPI_HANDLE *NewBSPHandle);
```

8.1.7bis.1 Description

This function initiates a BSP attach secure session and verifies that the version of the BSP expected by the application is compatible with the version on the system. The caller shall specify a list of zero or more BioAPI

Units that the BSP is to use in the attach session being created, and also specify the security profile information to be used in biometric operations afterwards.

This function shall only be called (for a given BSP UUID) if there is at least one call to ***BioAPI_BSPLoad*** (for that BSP UUID) for which a corresponding call to ***BioAPI_BSPUnload*** has not yet been made. The function ***BioAPI_BSPAttachSecure*** can be invoked multiple times for each call to ***BioAPI_BSPLoad*** (prior to a call to ***BioAPI_BSPUnload***), for the same BSP, creating multiple invocations of that BSP.

8.1.7bis.2 Parameters

BSPUuid (input) – a pointer to the UUID structure containing the global unique identifier for the BSP.

Version (input) – the major and minor version number of the BioAPI specification that the application is expecting the BSP to support. The BSP shall determine whether it is compatible with the required version.

ACBioParameters (input/optional*)* – Parameters about ACBio generation. A NULL pointer shall be set if no generation of ACBio instances is necessary.

UnitList (input) – a pointer to a buffer containing a list of BioAPI_UNIT_LIST_ELEMENT structures indicating to the BSP which BioAPI Units (supported by the BSP) it is to use for this attach session. The structures contain the ID and category of each BioAPI Unit. The application may specify one of the following for each category of BioAPI Unit:

a. Select a specific BioAPI Unit: To specify a particular BioAPI Unit to use for this attach session, the ID and category of that BioAPI Unit will be provided for that element.

b. Select any BioAPI Unit: When the UnitID is set to BioAPI_DONT_CARE in a particular element, the BSP will choose which BioAPI Unit of that category to use, or will give an error return if it does not support any BioAPI Units of that category. If a particular category is not listed, the BSP will likewise choose a BioAPI Unit of that category to use if it supports a BioAPI Unit of that category (however, there is no error return if it does not).

c. Select no BioAPI Unit: When the UnitID is set to BioAPI_DONT_INCLUDE, the BSP will explicitly not attach a BioAPI Unit of the given category, even if it supports one of that category.

NOTE: Any subsequent calls requiring use of a BioAPI Unit of this category will fail with an error return.

SecurityProfileList (input) – a pointer to a buffer containing a list of BioAPI_SECURITY_PROFILE structures which contain security information to be used in security operations by BioAPI Units. The order of this list shall keep that of the *UnitList.*

NumUnits (input) – The number of BioAPI Unit elements in the list that the pointer *UnitList* is pointing to.

NOTE: Only one BioAPI Unit of each category can be attached by a biometric application for each BSP attach secure session at any time.

NewBSPHandle (output) – a new handle that can be used to interact with the requested biometric service provider. The value will be set to BioAPIERR_FRAMEWORK_INVALID_BSP_HANDLE if the function fails.

8.1.7bis.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.

8.1.7bis.4 Errors

```
BioAPIERR_INCOMPATIBLE_VERSION
BioAPIERR_BSP_NOT_LOADED
```

```
BioAPIERR_INVALID UNIT_ID
BioAPIERR_INVALID_UUID
BioAPIERR_UNIT_IN_USE
BioAPIERR_INVALID_CATEGORY
BioAPIERR_SECURITY_CHALLENGE_NOT_SET
BioAPIERR_SECURITY_BPUIOINDEX_NOT_SET
BioAPIERR_SECURITY_SUPREMUM_BPUIOINDEX_NOT_SET
BioAPIERR_SECURITY_PROFILE_NOT_SET
BioAPIERR_SECURITY_ENCRYPTION_ALG_NOT_SUPPORTED
BioAPIERR_SECURITY_ENCRYPTION_ALG_NOT_SET
BioAPIERR_SECURITY_ENCKEY_NOT_SET
BioAPIERR_SECURITY_MAC_ALG_NOT_SUPPORTED
BioAPIERR_SECURITY_MACKEY_NOT_SET
BioAPIERR_SECURITY_MAC_ALG_NOT_SET
BioAPIERR_SECURITY_DIGITAL_SIGNATURE_ALG_NOT_SUPPORTED
BioAPIERR_SECURITY_DIGITAL_SIGNATURE_ALG_NOT_SET
BioAPIERR_SECURITY_HASH_ALG_ACBIO_NOT_SUPPORTED
BioAPIERR_SECURITY_HASH_ALG_ACBIO_NOT_SET
BioAPIERR_SECURITY_MAC_ALG_ACBIO_NOT_SUPPORTED
BioAPIERR_SECURITY_MACKEY_ACBIO_NOT_SET
BioAPIERR_SECURITY_MAC_ALG_ACBIO_NOT_SET
BioAPIERR_SECURITY_DIGITAL_SIGNATURE_ALG_ACBIO_NOT_SUPPORTED
BioAPIERR_SECURITY_DIGITAL_SIGNATURE_ALG_ACBIO_NOT_SET
```

See also BioAPI Error Handling (Clause 11).

2-36) In 8.1.8.1, replace the third paragraph as follows:

This function shall only be called after ***BioAPI_BSPAttach***/***BioAPI_BSPAttachSecure*** (BioAPI 2.2 or greater) has been called, and shall not be called more than once for the same BSP handle created by the call to ***BioAPI_BSPAttach***/ ***BioAPI_BSPAttachSecure*** (BioAPI 2.2 or greater).

2-37) Add the following NOTE at the end of 8.1.9.2:

NOTE:The structure of BioAPI_UNIT_SCHEMA is dependent on the version of BioAPI. BioAPI_UNIT_SCHEMA of BioAPI 2.2 contains security information while that of BioAPI 2.1 or less does not.

2-38) In 8.2.1, replace the second paragraph as follows:

This function shall only be called after ***BioAPI_BSPAttach/BioAPI_BSPAttachSecure*** (BioAPI 2.2 or greater) has been called, and shall not be called after ***BioAPI_BSPDetach*** has been called for the BSP handle created by the call to ***BioAPI_BSPAttach/BioAPI_BSPAttachSecure*** (BioAPI 2.2 or greater).

2-39) Replace the text of 8.2.2 as follows:

This function returns the BIR associated with a BIR handle returned by a BSP. If ***BioAPI_BSPAttachSecure*** (BioAPI 2.2 or greater)has been called***, the returned BIR shall contain the security block.*** The BIR handle is freed by the BSP before the function returns.

2-40) Replace 8.4.1.1 by the following:

8.4.1.1 Description

This function captures samples for the purpose specified, and the BSP returns either an "intermediate" type BIR (if the ***BioAPI_Process, BioAPI_ProcessWithAuxBIR***, or ***BioAPI_ProcessUsingAuxBIRs*** (BioAPI 2.2 or greater) function needs to be called), or a "processed" BIR (if not). The Purpose is recorded in the header of the *CapturedBIR*. If *AuditData* is non-NULL, a BIR of type "raw" may be returned. The function returns handles to whatever data is collected, and all local operations can be completed through use of the handles. If the application needs to acquire the data either to store it in a BIR database or to send it to a server, the application can retrieve the data with the ***BioAPI_GetBIRFromHandle*** function.

By default, the BSP is responsible for providing the user interface associated with the capture operation. The application may, however, request control of the GUI "look-and-feel" by providing a GUI callback pointer in ***BioAPI_SetGUICallbacks***. See clause C.7 for additional explanation of user interface features.

Capture serializes use of the sensor device. If two or more biometric applications are racing for the sensor, the losers will wait until the operation completes or the timeout expires. This serialization takes place in all functions that capture data. The BSP is responsible for serializing. It may do this by either returning 'busy' (BioAPI_UNIT_IN_USE) or by queuing requests.

If BioAPI_BSPAttachSecure (BioAPI 2.2 or greater) has been called, security operations shall be done according to the value set in the member of the parameter *SecurityProfileList* of ***BioAPI_BSPAttachSecure*** (BioAPI 2.2 or greater) corresponding to the BioAPI Unit which executes this function. If ACBio instance generation is requested, the challenge given as *Challenge* in the parameter *ACBioParameters* of ***BioAPI_BSPAttachSecure*** (BioAPI 2.2 or greater) shall be set into the field `controlValue` of type `ACBioContentInformation` in the ACBio instance. The current BPU IO index for the BSP Unit shall be set into the field `bpuIOIndex` of `bpuOuputExecutionInformationList` in the `biometricProcess` of `ACBioContentInformation` in the ACBio instance and also set into `bpuIOIndex` in `subBlockForACBio` in the security block. After that the current BPU IO index shall be incremented. The BIR shall be stored in type BioAPI_ASN1_BIR.

The BIR Handle returned by the function shall be released by the application (via ***BioAPI_FreeBIRHandle***) when it is no longer needed.

2-41) Replace 8.4.1.2 as follows:

8.4.1.2 Parameters:

BSPHandle (input) – The handle of the attached biometric service provider.

Purpose (input) – A value indicating the purpose of the biometric data capture.

Subtype (input/optional) – Specifies which subtype (e.g., left/right eye) to capture. A value of BioAPI_NO_SUBTYPE_AVAILABLE (0x00) indicates that the BSP is to select the subtype(s).

NOTE: Not all BSPs support the capture of specific *Subtypes*. Actual *Subtypes* captured will be reflected in the header of the returned *CapturedBIR*.

OutputFormat (input/optional) – Specifies which BDB format to use for the returned *CapturedBIR*, if the BSP supports more than one format. A NULL pointer value indicates that the BSP is to select the format.

CapturedBIR (output) – A handle to a BIR containing captured data. This data is either an "intermediate" type BIR (which can only be used by either the ***BioAPI_Process***, ***BioAPI_ProcessWithAuxBIR***, ***BioAPI_ProcessUsingAuxBIRs*** (BioAPI 2.2 or greater), or ***BioAPI_CreateTemplate*** functions, depending on the purpose), or a "processed" BIR (which can be used directly by ***BioAPI_VerifyMatch***,

BioAPI_VerifyMatchUsingAuxBIRs (BioAPI 2.2 or greater), or ***BioAPI_IdentifyMatch***, depending on the purpose).

Timeout (input) – An integer specifying the timeout value (in milliseconds) for the operation. If this timeout is reached, the function returns an error, and no results. This value can be any positive number. A '-1' value means the BSP's default timeout value will be used.

AuditData (output/optional) – A handle to a BIR containing raw biometric data. This data may be used to provide human-identifiable data of the person at the sensor unit. If the pointer is NULL on input, no audit data is collected. Not all BSPs support the collection of audit data. A BSP may return a BIR handle value of BioAPI_UNSUPPORTED_BIR_HANDLE to indicate *AuditData* is not supported, or a value of BioAPI_INVALID_BIR_HANDLE to indicate that no audit data is available.

2-42) Add the following at the end of errors in 8.4.1.4:

```
BioAPIERR_SECURITY_ENCRYPTION_FAILURE

BioAPIERR_SECURITY_MAC_GENERATION_FAILURE

BioAPIERR_SECURITY_DIGITAL_SIGNATURE_GENERATION_FAILURE
```

2-43) Add the following at the end of 8.4.2.1:

If BioAPI_BSPAttachSecure (BioAPI 2.2 or greater) has been called, security operations shall be done according to the value set in the member of the parameter *SecurityProfileList* of ***BioAPI_BSPAttachSecure*** (BioAPI 2.2 or greater) corresponding to the BioAPI Unit which executes this function. If ACBio instance generation is requested, the challenge given as *Challenge* in the parameter *ACBioParameters* of ***BioAPI_BSPAttachSecure*** (BioAPI 2.2 or greater) shall be set into the field `controlValue` of type `ACBioContentInformation` in the ACBio instance. The current BPU IO index for the BSP Unit shall be set into the field `bpuIOIndex` of `bpuOuputExecutionInformationList` in the `biometricProcess` of `ACBioContentInformation` in the ACBio instance and also set into `bpuIOIndex` in `subBlockForACBio` in the security block. After that the current BPU IO index shall be incremented. The BIR shall be stored in type BioAPI_ASN1_BIR.

2-44) Add the following after 8.4.2.4:

```
BioAPIERR_SECURITY_ENCRYPTION_FAILURE

BioAPIERR_SECURITY_DECRYPTION_FAILURE

BioAPIERR_SECURITY_MAC_GENERATION_FAILURE

BioAPIERR_SECURITY_MAC_VERIFICATION_FAILURE

BioAPIERR_SECURITY_DIGITAL_SIGNATURE_GENERATION_FAILURE

BioAPIERR_SECURITY_DIGITAL_SIGNATURE_VERIFICATION_FAILURE
```

2-45) Replace 8.4.3.1 by the following:

8.4.3.1 Description

This function processes the intermediate data captured via a call to ***BioAPI_Capture*** for the purpose of either verification or identification. If the processing capability is supported by the attached BSP invocation, the BSP builds a "processed biometric sample" BIR; otherwise, *ProcessedBIR* is set to NULL, and this function returns BioAPIERR_BSP_FUNCTION_NOT_SUPPORTED.

If BioAPI_BSPAttachSecure (BioAPI 2.2 or greater) has been called, security operations shall be done according to the value set in the member of the parameter *SecurityProfileList* of ***BioAPI_BSPAttachSecure*** (BioAPI 2.2 or greater) corresponding to the BioAPI Unit or BFP which executes this function. If ACBio instance generation is requested, the challenge given as *Challenge* in the parameter *ACBioParameters* of ***BioAPI_BSPAttachSecure*** (BioAPI 2.2 or greater) shall be set into the field `controlValue` of type `ACBioContentInformation` in the ACBio instance. The current BPU IO index for the BSP Unit shall be set into the field `bpuIOIndex` of `bpuOuputExecutionInformationList` in the `biometricProcess` of `ACBioContentInformation` in the ACBio instance and also set into `bpuIOIndex` in `subBlockForACBio` in the security block. After that the current BPU IO index shall be incremented. The BIR shall be stored in type BioAPI_ASN1_BIR.

This function results in the creation of a BIR by the BSP. The application can retrieve the BIR using the BIR handle through a call to ***BioAPI_GetBIRFromHandle***, which also frees the handle, or can release the memory associated with the BIR handle only through a call to ***BioAPI_FreeBIRHandle***.

2-46) Add the following at the end of errors in 8.4.3.4:

```
BioAPIERR_SECURITY_ENCRYPTION_FAILURE

BioAPIERR_SECURITY_DECRYPTION_FAILURE

BioAPIERR_SECURITY_MAC_GENERATION_FAILURE

BioAPIERR_SECURITY_MAC_VERIFICATION_FAILURE

BioAPIERR_SECURITY_DIGITAL_SIGNATURE_GENERATION_FAILURE

BioAPIERR_SECURITY_DIGITAL_SIGNATURE_VERIFICATION_FAILURE
```

2-47) Add NOTE after 8.4.4.1:

NOTE:This function is used only in BioAPI 2.1 or less. In BioAPI 2.2 or greater, ***BioAPI_ProcessUsingAuxBIRs*** (BioAPI 2.2) is used instead of this function.

2-48) Add the following after 8.4.4:

8.4.4bis BioAPI_ProcessUsingAuxBIRs (BioAPI 2.2)

```
BioAPI_RETURN BioAPI BioAPI_ProcessUsingAuxBIRs

  (BioAPI_HANDLE BSPHandle,

  const BioAPI_INPUT_BIR *CapturedBIR,

  uint32_t NumberOfAuxBIRs,
```

```
const BioAPI_INPUT_BIR *AuxBIRs,

const BioAPI_BIR_BIOMETRIC_DATA_FORMAT *OutputFormat,

BioAPI_BIR_HANDLE *ProcessedBIR);
```

8.4.4bis.1 Description

This function processes the intermediate data previously captured via a call to ***BioAPI_Capture*** in conjunction with auxiliary data, creating processed biometric samples for the purpose of subsequent verification or identification. It enables implementations that require the input of auxiliary data to the process operation.

In addition to the intermediate BIR, this function takes as input a list of auxiliary BIRs. Each auxiliary BIR in the list may be an intermediate, processed, score, decision, or biographic BIR. The biometric service provider may modify the way it performs the processing operation based on the auxiliary BIRs provided as input and on the order in which they occur in the list. However, such modifications to the processing operation are not specified in this standard.

The main purposes of the auxiliary BIRs are:

a) to provide feedback to the biometric processing operation from prior processing, matching, fusion, or decision operations; and

b) to modify the biometric processing operation based on statistical information (either specific to the subject, or specific to a population, or both).

If BioAPI_BSPAttachSecure (BioAPI 2.2 or greater) has been called, security operations shall be done according to the value set in the member of the parameter *SecurityProfileList* of ***BioAPI_BSPAttachSecure*** (BioAPI 2.2 or greater) corresponding to the BioAPI Unit or BFP which executes this function. If ACBio instance generation is requested, the challenge given as *Challenge* in the parameter *ACBioParameters* of ***BioAPI_BSPAttachSecure*** (BioAPI 2.2 or greater) shall be set into the field `controlValue` of type `ACBioContentInformation` in the ACBio instance. The current BPU IO index for the BSP Unit shall be set into the field `bpuIOIndex` of `bpuOutputExecutionInformationList` in the `biometricProcess` of `ACBioContentInformation` in the ACBio instance and also set into `bpuIOIndex` in `subBlockForACBio` in the security block. After that the current BPU IO index shall be incremented. The BIR shall be stored in type BioAPI_ASN1_BIR.

This function results in the creation of a BIR by the BSP. The application can retrieve the BIR using the BIR handle through a call to ***BioAPI_GetBIRFromHandle***, which also frees the handle, or can release the memory associated with the BIR handle only through a call to ***BioAPI_FreeBIRHandle***.

8.4.4bis.2 Parameters

BSPHandle (input) – The handle of the attached biometric service provider.

CapturedBIR (input) – A BIR obtained from the BioAPI_Capture function previously called.

NumberOfAuxBIRs (input) – The number of elements (auxiliary BIRs) in the array pointed to by the parameter AuxBIRs.

AuxBIRs (input) – A pointer to an array of elements of type BioAPI_INPUT_BIR. The number of elements of the array shall be *NumberOfAuxBIRs*. Each auxiliary BIR may be an intermediate, processed, score, decision, or biographic BIR, and any combination of such BIR types is allowed. The order of the elements is significant. If *NumberOfAuxBIRs* is zero, *AuxiliaryBIRs* may be a NULL pointer.

NOTE: The content and format of the auxiliary data are specified by the BIR Biometric Data Format field in the auxiliary BIR header and may be specific to a BSP.

OutputFormat (input/optional) – Specifies which BDB format to use for the returned *ProcessedBIR*, if the BSP supports more than one format. A NULL pointer value indicates that the BSP is to select the format.

ProcessedBIR (output) – A handle for the newly constructed "processed" BIR.

8.4.4bis.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.

8.4.4bis.4 Errors

```
BioAPIERR_INVALID_BIR_HANDLE

BioAPIERR_INVALID_BIR

BioAPIERR_BIR_SIGNATURE_FAILURE

BioAPIERR_TOO_MANY_HANDLES

BioAPIERR_INCONSISTENT_PURPOSE

BioAPIERR_PURPOSE_NOT_SUPPORTED

BioAPIERR_UNSUPPORTED_FORMAT

BioAPIERR_RECORD_NOT_FOUND

BioAPIERR_FUNCTION_NOT_SUPPORTED

BioAPIERR_SECURITY_ENCRYPTION_FAILURE

BioAPIERR_SECURITY_DECRYPTION_FAILURE

BioAPIERR_SECURITY_MAC_GENERATION_FAILURE

BioAPIERR_SECURITY_MAC_VERIFICATION_FAILURE

BioAPIERR_SECURITY_DIGITAL_SIGNATURE_GENERATION_FAILURE

BioAPIERR_SECURITY_DIGITAL_SIGNATURE_VERIFICATION_FAILURE
```

See also **BioAPI Error Handling** (Clause 11).

2-49) Replace 8.4.5.1 by the following:

8.4.5.1 Description

This function performs a verification (1-to-1) match between two BIRs: the *ProcessedBIR* and the *ReferenceTemplate*. The *ProcessedBIR* is the "processed" BIR constructed specifically for this verification. The *ReferenceTemplate* was created at enrollment.

The application shall request a maximum FMR value criterion (threshold) for a successful match. The Boolean *Result* indicates whether verification was successful or not, and the *FMRAchieved* is a FMR value (score) indicating how closely the BIRs actually matched.

NOTE: See clause C.4 for information on the use of the FMR concept for normalized scoring and thresholding.

By setting the *AdaptedBIR* pointer to an address other than NULL, the application can request that a BIR be constructed by adapting the *ReferenceTemplate* using the *ProcessedBIR*. A new handle is returned to the *AdaptedBIR*. If the match is successful, an attempt may be made to adapt the *ReferenceTemplate* with information taken from the *ProcessedBIR*. (Not all BSPs perform adaptation). The resulting *AdaptedBIR* should now be considered an optimal enrollment template, and be saved in the BIR database. (It is up to the application whether it uses or discards this data.) It is important to note that adaptation may not occur in all cases. In the event of an adaptation, this function stores the handle to the new BIR in the memory pointed to by the *AdaptedBIR* parameter.

If a *Payload* is associated with the *ReferenceTemplate*, the *Payload* may be returned upon successful verification if the *FMRAchieved* is sufficiently stringent; this is controlled by the policy of the BSP and specified in its schema.

NOTE 1: Not all BSPs support return of payloads.

NOTE 2: See clauses A.4.6.2.6 and C.5 for additional information regarding use of payloads.

If BioAPI_BSPAttachSecure (BioAPI 2.2 or greater) has been called, security operations shall be done according to the value set in the member of the parameter *SecurityProfileList* of ***BioAPI_BSPAttachSecure*** (BioAPI 2.2 or greater) corresponding to the BioAPI Unit or BFP which executes this function. If ACBio instance generation is requested, the challenge given as *Challenge* in the parameter *ACBioParameters* of ***BioAPI_BSPAttachSecure*** (BioAPI 2.2 or greater) shall be set into the field `controlValue` of type `ACBioContentInformation` in the ACBio instance. The current BPU IO index for the BSP Unit shall be set into the field `bpuIOIndex` of `bpuOuputExecutionInformationList` in the `biometricProcess` of `ACBioContentInformation` in the ACBio instance and also set into `bpuIOIndex` in `subBlockForACBio` in the security block. After that the current BPU IO index shall be incremented. The BIR shall be stored in type BioAPI_ASN1_BIR.

The memory block returned by the BioAPI function call shall be freed by the application as soon as it is no longer needed using ***BioAPI_Free*** (see clause 8.7.2). If an adapted BIR is returned, its handle can be released through a call to ***BioAPI_FreeBIRHandle***.

2-50) Add the following at the end of errors in 8.4.5.4:

```
    BioAPIERR_SECURITY_ENCRYPTION_FAILURE

    BioAPIERR_SECURITY_DECRYPTION_FAILURE

    BioAPIERR_SECURITY_MAC_GENERATION_FAILURE

    BioAPIERR_SECURITY_MAC_VERIFICATION_FAILURE

    BioAPIERR_SECURITY_DIGITAL_SIGNATURE_GENERATION_FAILURE

BioAPIERR_SECURITY_DIGITAL_SIGNATURE_VERIFICATION_FAILURE
```

2-51) Add the following text after 8.4.5:

8.4.5bis BioAPI_VerifyMatchUsingAuxBIRs (BioAPI 2.2)

```
BioAPI_RETURN BioAPI BioAPI_VerifyMatchUsingAuxBIRs

  (BioAPI_HANDLE BSPHandle,

  BioAPI_FMR MaxFMRRequested,
```

```
  const BioAPI_INPUT_BIR *ProcessedBIR,

  const BioAPI_INPUT_BIR *ReferenceTemplate,

  uint32_t NumberOfAuxBIRs,

  const BioAPI_INPUT_BIR *AuxBIRs,

  BioAPI_BIR_HANDLE *AdaptedBIR,

  BioAPI_BOOL *Result,

  BioAPI_FMR *FMRAchieved,

  BioAPI_BIR_HANDLE *ResultBIR,

  BioAPI_DATA *Payload);
```

8.4.5bis.1 Description

This function performs a verification (1-to-1) match between two BIRs: the *ProcessedBIR* and the *ReferenceTemplate*. The *ProcessedBIR* is the "processed" BIR constructed specifically for this verification. The *ReferenceTemplate* was created at enrollment.

The application shall request a maximum FMR value criterion (threshold) for a successful match.

In addition to the processed BIR and the reference template BIR, this function takes as input a list of auxiliary BIRs. Each auxiliary BIR in the list may be an intermediate, processed, score, decision, or biographic BIR.

The biometric service provider may modify the way it performs the verification operation based on the auxiliary BIRs provided as input and on the order in which they occur in the list. However, such modifications to the verification operation are not specified in this standard.

The main purposes of the auxiliary BIRs are:

a) to provide feedback to the biometric matching operation from prior processing, matching, fusion, or decision operations; and

b) to modify the biometric matching operation based on statistical information (either specific to the subject, or specific to a population, or both).

By setting the *AdaptedBIR* pointer to an address other than NULL, the application can request that a BIR be constructed by adapting the *ReferenceTemplate* using the *ProcessedBIR*. A new handle is returned to the *AdaptedBIR*. If the match is successful, an attempt may be made to adapt the *ReferenceTemplate* with information taken from the *ProcessedBIR*. (Not all BSPs perform adaptation). The resulting AdaptedBIR should now be considered an optimal enrollment template, and be saved in the BIR database. (It is up to the application whether it uses or discards this data.) It is important to note that adaptation may not occur in all cases. In the event of an adaptation, this function stores the handle to the new BIR in the memory pointed to by the *AdaptedBIR* parameter.

The Boolean *Result* indicates whether verification was successful or not, and the *FMRAchieved* is a FMR value (score) indicating how closely the BIRs actually matched.

NOTE: See clause C.4 for information on the use of the FMR concept for normalized scoring and thresholding.

In addition to *Result*, this function returns a *ResultBIR*, which must be either a score BIR or a decision BIR containing a BDB whose format is chosen by the BSP. If a score BIR is returned, it must contain the score of the matching operation as specified by the particular "score" BDB format, along with any additional information specified for that BDB format. If a decision BIR is returned, it must contain the boolean result of the matching

operation (whether verification was successful or not) as specified by the particular "decision" BDB format, along with any additional information specified for that BDB format.

It is recommended that BSP return a score BIR rather than a decision BIR whenever possible. While a decision BIR can only be used in decision-level fusion (thus preventing the use of score-level fusion), a score BIR can either be used in score-level fusion or passed as input to ***BioAPI_Decide*** to produce a decision BIR, which can then be used in decision-level fusion. Return of score BIRs results therefore in greater flexibility for applications that use biometric fusion.

If a Payload is associated with the *ReferenceTemplate*, the Payload may be returned upon successful verification if the *FMRAchieved* is sufficiently stringent; this is controlled by the policy of the BSP and specified in its schema.

NOTE 1: Not all BSPs support return of payloads.

NOTE 2: See clauses A.4.6.2.6 and C.5 for additional information regarding use of payloads.

If BioAPI_BSPAttachSecure (BioAPI 2.2 or greater) has been called, security operations shall be done according to the value set in the member of the parameter *SecurityProfileList* of ***BioAPI_BSPAttachSecure*** (BioAPI 2.2 or greater) corresponding to the BioAPI Unit or BFP which executes this function. If ACBio instance generation is requested, the challenge given as *Challenge* in the parameter *ACBioParameters* of ***BioAPI_BSPAttachSecure*** (BioAPI 2.2 or greater) shall be set into the field `controlValue` of type `ACBioContentInformation` in the ACBio instance. The current BPU IO index for the BSP Unit shall be set into the field `bpuIOIndex` of `bpuOuputExecutionInformationList` in the `biometricProcess` of `ACBioContentInformation` in the ACBio instance and also set into `bpuIOIndex` in `subBlockForACBio` in the security block. After that the current BPU IO index shall be incremented. The BIR shall be stored in type BioAPI_ASN1_BIR.

The memory block returned by the BioAPI function call shall be freed by the application as soon as it is no longer needed using ***BioAPI_Free*** (see clause 8.7.2). If an adapted BIR is returned, its handle can be released through a call to ***BioAPI_FreeBIRHandle***.

8.4.5bis.2 Parameters

BSPHandle (input) – The handle of the attached biometric service provider.

MaxFMRRequested (input) – The requested FMR criterion for successful verification (i.e., the matching threshold).

ProcessedBIR (input) – The BIR to be verified, or its handle.

ReferenceTemplate (input) – The BIR to be verified against, or its key in a BIR database, or its handle.

NumberOfAuxBIRs (input) – The number of elements (auxiliary BIRs) in the array pointed to by the parameter AuxBIRs.

AuxBIRs (input) – A pointer to an array of elements of type BioAPI_INPUT_BIR. The number of elements of the array shall be NumberOfAuxBIRs. Each auxiliary BIR may be an intermediate, processed, score, decision, or biographic BIR, and any combination of such BIR types is allowed. The order of the elements is significant. If NumberOfAuxBIRs is zero, AuxBIRs may be a NULL pointer.

AdaptedBIR (output/optional) – A pointer to the handle of the adapted BIR. This parameter can be NULL if an adapted BIR is not desired. Not all BSPs support the adaptation of BIRs. The function may return a handle value of BioAPI_UNSUPPORTED_BIR_HANDLE to indicate that adaptation is not supported or a value of BioAPI_INVALID_BIR_HANDLE to indicate that adaptation was not possible.

Result (output) – A pointer to a Boolean value indicating (BioAPI_TRUE/BioAPI_FALSE) whether the BIRs matched or not according to the specified criteria.

FMRAchieved (output) – A pointer to an FMR value indicating the closeness of the match (i.e., the match score).

ResultBIR (output) – A handle for the newly constructed "score" or "decision" BIR.

Payload (output/optional) – If a payload is associated with the ReferenceTemplate, it is returned in an allocated BioAPI_DATA structure if the FMRAchieved satisfies the payload policy of the BSP.

8.4.5bis.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.

8.4.5bis.4 Errors

```
BioAPIERR_INVALID_BIR_HANDLE

BioAPIERR_INVALID_BIR

BioAPIERR_BIR_SIGNATURE_FAILURE

BioAPIERR_INCONSISTENT_PURPOSE

BioAPIERR_BIR_NOT_FULLY_PROCESSED

BioAPIERR_RECORD_NOT_FOUND

BioAPIERR_QUALITY_ERROR

BioAPIERR_SECURITY_ENCRYPTION_FAILURE

BioAPIERR_SECURITY_DECRYPTION_FAILURE

BioAPIERR_SECURITY_MAC_GENERATION_FAILURE

BioAPIERR_SECURITY_MAC_VERIFICATION_FAILURE

BioAPIERR_SECURITY_DIGITAL_SIGNATURE_GENERATION_FAILURE

BioAPIERR_SECURITY_DIGITAL_SIGNATURE_VERIFICATION_FAILURE
```

See also **BioAPI Error Handling** (Clause 11).

2-52) Add the following text at the end of 8.4.6.1:

If BioAPI_BSPAttachSecure (BioAPI 2.2 or greater) has been called, security operations shall be done according to the value set in the member of the parameter *SecurityProfileList* of ***BioAPI_BSPAttachSecure*** (BioAPI 2.2 or greater) corresponding to the BioAPI Unit which executes this function.

2-53) Add the following at the end of errors in 8.4.6.4:

```
BioAPIERR_SECURITY_ENCRYPTION_FAILURE

BioAPIERR_SECURITY_DECRYPTION_FAILURE
```

```
BioAPIERR_SECURITY_MAC_GENERATION_FAILURE

BioAPIERR_SECURITY_MAC_VERIFICATION_FAILURE

BioAPIERR_SECURITY_DIGITAL_SIGNATURE_GENERATION_FAILURE

BioAPIERR_SECURITY_DIGITAL_SIGNATURE_VERIFICATION_FAILURE
```

2-54) Add the following text after 8.4.6:

8.4.6bis BioAPI_Decide (BioAPI 2.2)

```
BioAPI_RETURN BioAPI BioAPI_Decide

  (BioAPI_HANDLE BSPHandle,

  const BioAPI_INPUT_BIR *ScoreBIR,

  uint32_t NumberOfAuxBIRs,

  const BioAPI_INPUT_BIR *AuxBIRs,

  BioAPI_BIR_HANDLE *DecisionBIR);
```

8.4.6bis.1 Description

This function processes a score BIR produced by a call to ***BioAPI_VerifyMatchUsingAuxBIRs*** or ***BioAPI_Fuse*** and builds a decision BIR.

In addition to the score BIR, this function takes as input a list of auxiliary BIRs. Each auxiliary BIR in the list may be an intermediate, processed, score, decision, or biographic BIR. The biometric service provider may modify the way it performs the decision operation based on the auxiliary BIRs provided as input and on the order in which they occur in the list. However, such modifications to the decision operation are not specified in this standard.

The main purposes of the auxiliary BIRs are:

a) to provide feedback to the biometric decision operation from prior processing, matching, fusion, or decision operations; and

b) to modify the biometric decision operation based on statistical information (either specific to the subject, or specific to a population, or both).

If BioAPI_BSPAttachSecure (BioAPI 2.2 or greater) has been called, security operations shall be done according to the value set in the member of the parameter *SecurityProfileList* of ***BioAPI_BSPAttachSecure*** (BioAPI 2.2 or greater) corresponding to the BioAPI Unit or BFP which executes this function. If ACBio instance generation is requested, the challenge given as *Challenge* in the parameter *ACBioParameters* of ***BioAPI_BSPAttachSecure*** (BioAPI 2.2 or greater) shall be set into the field `controlValue` of type `ACBioContentInformation` in the ACBio instance. The current BPU IO index for the BSP Unit shall be set into the field `bpuIOIndex` of `bpuOuputExecutionInformationList` in the `biometricProcess` of `ACBioContentInformation` in the ACBio instance and also set into `bpuIOIndex` in `subBlockForACBio` in the security block. After that the current BPU IO index shall be incremented. The BIR shall be stored in type BioAPI_ASN1_BIR.

8.4.6bis.2 Parameters

BSPHandle (input) – The handle of the attached BioAPI service provider.

ScoreBIR (input) – The score BIR or its handle.

NumberOfAuxBIRs (input) – The number of elements (auxiliary BIRs) in the array pointed to by the parameter AuxBIRs.

AuxBIRs (input) – A pointer to an array of elements of type BioAPI_INPUT_BIR. The number of elements of the array shall be *NumberOfAuxBIRs*. Each auxiliary BIR may be an intermediate, processed, score, decision, or biographic BIR, and any combination of such BIR types is allowed. The order of the elements is significant. If *NumberOfAuxBIRs* is zero, *AuxBIRs* may be a NULL pointer.

DecisionBIR (output) – A handle for the newly constructed decision BIR, or zero if the BSP does not support this function.

8.4.6bis.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.

8.4.6bis.4 Errors

```
BioAPIERR_INVALID_BIR_HANDLE

BioAPIERR_INVALID_BIR

BioAPIERR_BIR_SIGNATURE_FAILURE

BioAPIERR_INCONSISTENT_PURPOSE

BioAPIERR_BIR_NOT_FULLY_PROCESSED

BioAPIERR_RECORD_NOT_FOUND

BioAPIERR_QUALITY_ERROR

BioAPIERR_SECURITY_ENCRYPTION_FAILURE

BioAPIERR_SECURITY_DECRYPTION_FAILURE

BioAPIERR_SECURITY_MAC_GENERATION_FAILURE

BioAPIERR_SECURITY_MAC_VERIFICATION_FAILURE

BioAPIERR_SECURITY_DIGITAL_SIGNATURE_GENERATION_FAILURE

BioAPIERR_SECURITY_DIGITAL_SIGNATURE_VERIFICATION_FAILURE
```

See also **BioAPI Error Handling** (Clause 11).

8.4.6ter BioAPI_Fuse (BioAPI 2.2)

```
BioAPI_RETURN BioAPI BioAPI_Fuse

  (BioAPI_HANDLE BSPHandle,

  uint32_t NumberOfSourceBIRs,
```

```
const BioAPI_INPUT_BIR *SourceBIRs,

uint32_t NumberOfAuxBIRs,

const BioAPI_INPUT_BIR *AuxBIRs,

BioAPI_BIR_HANDLE *FusionBIR);
```

8.4.6ter.1 Description

This function performs a biometric fusion operation on the source BIRs provided as input, which shall be two or more BIRs of the same type (processed, score, or decision BIRs), and produces a fusion BIR. The order of the source BIRs is significant.

If the source BIRs are processed BIRs, the fusion BIR shall be a processed BIR. If the source BIRs are score BIRs, the fusion BIR shall be either a score BIR or a decision BIR. If the source BIRs are decision BIRs, the fusion BIR shall be a decision BIR.

In addition to the source BIRs, this function takes as input a list of auxiliary BIRs. Each auxiliary BIR in the list may be an intermediate, processed, score, decision, or biographic BIR. The biometric service provider may modify the way it performs the fusion operation based on the auxiliary BIRs provided as input and on the order in which they occur in the list. However, such modifications to the fusion operation are not specified in this standard.

The main purposes of the auxiliary BIRs are:

a) to provide feedback to the biometric fusion operation from prior processing, matching, fusion, or decision operations; and

b) to modify the biometric fusion operation based on statistical information (either specific to the subject, or specific to a population, or both).

If BioAPI_BSPAttachSecure (BioAPI 2.2 or greater) has been called, security operations shall be done according to the value set in the member of the parameter *SecurityProfileList* of ***BioAPI_BSPAttachSecure*** (BioAPI 2.2 or greater) corresponding to the BioAPI Unit or BFP which executes this function. If ACBio instance generation is requested, the challenge given as *Challenge* in the parameter *ACBioParameters* of ***BioAPI_BSPAttachSecure*** (BioAPI 2.2 or greater) shall be set into the field `controlValue` of type `ACBioContentInformation` in the ACBio instance. The current BPU IO index for the BSP Unit shall be set into the field `bpuIOIndex` of `bpuOuputExecutionInformationList` in the `biometricProcess` of `ACBioContentInformation` in the ACBio instance and also set into `bpuIOIndex` in `subBlockForACBio` in the security block. After that the current BPU IO index shall be incremented. The BIR shall be stored in type BioAPI_ASN1_BIR.

8.4.6ter.2 Parameters

BSPHandle (input) – The handle of the attached BioAPI service provider.

NumberOfSourceBIRs (input) – The number of elements (source BIRs) in the array pointed to by the parameter SourceBIRs. This number shall be greater than or equal to two.

SourceBIRs (input) – A pointer to an array of elements of type BioAPI_INPUT_BIR. The number of elements of the array shall be *NumberOfSourceBIRs*, and the BIRs shall all be of the same type (processed, score, or decision BIRs). The order of the elements is significant.

NumberOfAuxBIRs (input) – The number of elements (auxiliary BIRs) in the array pointed to by the parameter AuxBIRs.

AuxBIRs (input) – A pointer to an array of elements of type BioAPI_INPUT_BIR. The number of elements of the array shall be *NumberOfAuxBIRs*. Each auxiliary BIR may be an intermediate, processed, score,

decision, or personal BIR, and any combination of such BIR types is allowed. The order of the elements is significant. If *NumberOfAuxBIRs* is zero, *AuxBIRs* may be a NULL pointer.

FusionBIR (output) – A handle for the newly constructed fusion BIR, or zero if the BSP does not support fusion. The fusion BIR shall be a processed, score, or a decision BIR.

8.4.6ter.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.

8.4.6ter.4 Errors

`BioAPIERR_INVALID_BIR_HANDLE`

`BioAPIERR_INVALID_BIR`

`BioAPIERR_BIR_SIGNATURE_FAILURE`

`BioAPIERR_INCONSISTENT_PURPOSE`

`BioAPIERR_BIR_NOT_FULLY_PROCESSED`

`BioAPIERR_RECORD_NOT_FOUND`

`BioAPIERR_QUALITY_ERROR`

`BioAPIERR_SECURITY_ENCRYPTION_FAILURE`

`BioAPIERR_SECURITY_DECRYPTION_FAILURE`

`BioAPIERR_SECURITY_MAC_GENERATION_FAILURE`

`BioAPIERR_SECURITY_MAC_VERIFICATION_FAILURE`

`BioAPIERR_SECURITY_DIGITAL_SIGNATURE_GENERATION_FAILURE`

`BioAPIERR_SECURITY_DIGITAL_SIGNATURE_VERIFICATION_FAILURE`

See also **BioAPI Error Handling** (Clause 11).

2-55) Add the following at the end of 8.4.7.1:

If BioAPI_BSPAttachSecure (BioAPI 2.2 or greater) has been called, security operations shall be done according to the value set in the member of the parameter *SecurityProfileList* of ***BioAPI_BSPAttachSecure*** (BioAPI 2.2 or greater) corresponding to the BioAPI Unit which executes this function. If ACBio instance generation is requested, the challenge given as *Challenge* in the parameter *ACBioParameters* of ***BioAPI_BSPAttachSecure*** (BioAPI 2.2 or greater) shall be set into the field `controlValue` of type `ACBioContentInformation` in the ACBio instance. The current BPU IO index for the BSP Unit shall be set into the field `bpuIOIndex` of `bpuOuputExecutionInformationList` in the `biometricProcess` of `ACBioContentInformation` in the ACBio instance and also set into `bpuIOIndex` in `subBlockForACBio` in the security block. After that the current BPU IO index shall be incremented. The BIR shall be stored in type BioAPI_ASN1_BIR.

2-56) Add the following at the end of errors in 8.4.7.4:

`BioAPIERR_SECURITY_ENCRYPTION_FAILURE`

`BioAPIERR_SECURITY_MAC_GENERATION_FAILURE`

`BioAPIERR_SECURITY_DIGITAL_SIGNATURE_GENERATION_FAILURE`

2-57) Replace 8.4.8.1 by the following:

8.4.8.1 Description

This function captures biometric data from the attached device (sensor unit), and compares it against the *ReferenceTemplate*.

The application shall request a maximum FMR value criterion (threshold) for a successful match. The Boolean *Result* indicates whether verification was successful or not, and the *FMRAchieved* is a FMR value (score) indicating how closely the BIRs actually matched.

NOTE: See clause C.4 for information on the use of the FMR concept for normalized scoring and thresholding.

By setting the *AdaptedBIR* pointer to non-NULL, the application can request that a BIR be constructed by adapting the *ReferenceTemplate* using the *ProcessedBIR*. A new handle is returned to the *AdaptedBIR.* If the match is successful, an attempt may be made to adapt the *ReferenceTemplate* with information taken from the *ProcessedBIR*. (Not all BSPs perform adaptation). The resulting *AdaptedBIR* should now be considered an optimal enrollment template, and be saved in the BIR database. (It is up to the application whether it uses or discards this data). It is important to note that adaptation may not occur in all cases. In the event of an adaptation, this function stores the handle to the new BIR in the memory pointed to by the *AdaptedBIR* parameter.

If a *Payload* is associated with the *ReferenceTemplate*, the *Payload* may be returned upon successful verification if the *FMRAchieved* is sufficiently stringent; this is controlled by the policy of the BSP and specified in its schema.

NOTE 1: Not all BSPs support return of payloads.

NOTE 2: See clauses A.4.6.2.6 and C.5 for additional information regarding use of payloads.

The BSP is responsible for providing the user interface associated with the verify operation as a default. The application may request control of the GUI “look-and-feel” by providing a GUI callback pointer in ***BioAPI_SetGUICallbacks***. See clause C.7 for additional explanation of user interface features.

Since the ***BioAPI_Verify*** operation includes a capture, it serializes use of the sensor device. If two or more applications are racing for the device, the losers will wait until the operation completes or the timeout expires. This serialization takes place in all functions that capture data. The BSP is responsible for serializing. It may do this by either returning ‘busy’ (BioAPI_UNIT_IN_USE) or by queuing requests.

If BioAPI_BSPAttachSecure (BioAPI 2.2 or greater) has been called, security operations shall be done according to the value set in the member of the parameter *SecurityProfileList* of ***BioAPI_BSPAttachSecure*** (BioAPI 2.2 or greater) corresponding to the BioAPI Units or BFP which execute this function. If ACBio instance generation is requested, the challenge given as *Challenge* in the parameter *ACBioParameters* of ***BioAPI_BSPAttachSecure*** (BioAPI 2.2 or greater) shall be set into the field `controlValue` of type `ACBioContentInformation` in the ACBio instance(s). The current BPU IO index for the BSP Unit shall be set into the field `bpuIOIndex` of `bpuOuputExecutionInformationList` in the `biometricProcess` of `ACBioContentInformation` in the ACBio instance(s) and also set into `bpuIOIndex` in `subBlockForACBio` in the security block. After that the current BPU IO index shall be incremented. The BIR shall be stored in type BioAPI_ASN1_BIR.

The memory block returned by the BioAPI function call shall be freed by the application as soon as it is no longer needed using ***BioAPI_Free*** (see clause 8.7.2). Output BIRs can be retrieved by a call to ***BioAPI_GetBIRFromHandle***, which releases the handle, or the handle can be released without retrieving the BIR through ***BioAPI_FreeBIRHandle***.

2-58) Add the following at the end of errors in 8.4.8.4:

```
BioAPIERR_SECURITY_ENCRYPTION_FAILURE

BioAPIERR_SECURITY_DECRYPTION_FAILURE

BioAPIERR_SECURITY_MAC_GENERATION_FAILURE

BioAPIERR_SECURITY_MAC_VERIFICATION_FAILURE

BioAPIERR_SECURITY_DIGITAL_SIGNATURE_GENERATION_FAILURE

BioAPIERR_SECURITY_DIGITAL_SIGNATURE_VERIFICATION_FAILURE
```

2-59) Add the following text at the end of 8.4.9.1:

If BioAPI_BSPAttachSecure (BioAPI 2.2 or greater) has been called, security operations shall be done according to the value set in the member of the parameter *SecurityProfileList* of ***BioAPI_BSPAttachSecure*** (BioAPI 2.2 or greater) corresponding to the BioAPI Unit which executes this function.

2-60) Add the following at the end of errors in 8.4.9.4:

```
BioAPIERR_SECURITY_ENCRYPTION_FAILURE

BioAPIERR_SECURITY_MAC_GENERATION_FAILURE

BioAPIERR_SECURITY_DIGITAL_SIGNATURE_GENERATION_FAILURE
```

2-61) Add the following text at the end of 8.4.10.1:

If BioAPI_BSPAttachSecure (BioAPI 2.2 or greater) has been called, security operations shall be done according to the value set in the member of the parameter *SecurityProfileList* of ***BioAPI_BSPAttachSecure*** (BioAPI 2.2 or greater) corresponding to the BioAPI Unit which executes this function.

2-62) Add the following at the end of errors in 8.4.10.4:

```
BioAPIERR_SECURITY_DECRYPTION_FAILURE

BioAPIERR_SECURITY_MAC_VERIFICATION_FAILURE

BioAPIERR_SECURITY_DIGITAL_SIGNATURE_VERIFICATION_FAILURE
```

2-63) Add the following text after 8.4.10.4:

8.4.10bis BioAPI_Export (BioAPI 2.2)

```
BioAPI_RETURN BioAPI BioAPI_Export
   (BioAPI_HANDLE BSPHandle,
   BioAPI_BIR_HANDLE Handle,
   const BioAPI_BIR_BIOMETRIC_DATA_FORMAT *OutputBDBFormat,
   const BioAPI_BIR_SECURITY_BLOCK_FORMAT *OutputSBFormat,
   BioAPI_BIR *BIR);
```

8.4.10bis.1 Description

This function returns the BIR of the BDB format identified by OutputBDBFormat and the SB format identified by OutputSBFormat, associated with a BIR handle and stored in BioAPI_ASN1_BIR form, returned by a BSP.

If BioAPI_BSPAttachSecure (BioAPI 2.2 or greater) has been called, security operations shall be done according to the value set in the member of the parameter *SecurityProfileList* of ***BioAPI_BSPAttachSecure*** (BioAPI 2.2 or greater) corresponding to the BioAPI Unit which executes this function.

8.4.10bis.2 Parameters

BSPHandle (input) – The handle of the attached biometric service provider.

Handle (input) – The handle of the BIR to be exported.

OutputBDBFormat (input) – Specifies which BDB format to use for the returned BIR, if the BSP supports more than one format.

OutputSBFormat (input/optional) – Specifies which SB format to use for the returned BIR, if the BSP supports more than one format. A NULL pointer value indicates that SB is not used for the returned BIR.

BIR (output) – Pointer to the exported BIR.

8.4.10bis.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.

8.4.10bis.4 Errors

```
BioAPIERR_INVALID_BIR_HANDLE
BioAPIERR_INVALID_BSP_HANDLE
BioAPIERR_FUNCTION_NOT_SUPPORTED
BioAPIERR_SECURITY_ENCRYPTION_FAILURE
BioAPIERR_SECURITY_MAC_GENERATION_FAILURE
BioAPIERR_SECURITY_DIGITAL_SIGNATURE_GENERATION_FAILURE
```

See also **BioAPI Error Handling** (Clause 11).

2-64) Add the following at the end of 8.5.7.1:

If BioAPI_BSPAttachSecure (BioAPI 2.2 or greater) has been called, security operations shall be done according to the value set in the member of the parameter *SecurityProfileList* of ***BioAPI_BSPAttachSecure*** (BioAPI 2.2 or greater) corresponding to the BioAPI Unit which executes this function. If ACBio instance generation is requested, the challenge given as *Challenge* in the parameter *ACBioParameters* of ***BioAPI_BSPAttachSecure*** (BioAPI 2.2 or greater) shall be set into the field `controlValue` of type `ACBioContentInformation` in the ACBio instance. The current BPU IO index for the BSP Unit shall be set into the field `bpuIOIndex` of `bpuOuputExecutionInformationList` in the `biometricProcess` of `ACBioContentInformation` in the ACBio instance and also set into `bpuIOIndex` in `subBlockForACBio` in the security block. After that the current BPU IO index shall be incremented. The BIR shall be stored in type BioAPI_ASN1_BIR.

2-65) Add the following at the end of errors in 8.5.7.4:

```
BioAPIERR_SECURITY_ENCRYPTION_FAILURE

BioAPIERR_SECURITY_DECRYPTION_FAILURE

BioAPIERR_SECURITY_MAC_GENERATION_FAILURE

BioAPIERR_SECURITY_MAC_VERIFICATION_FAILURE

BioAPIERR_SECURITY_DIGITAL_SIGNATURE_GENERATION_FAILURE

BioAPIERR_SECURITY_DIGITAL_SIGNATURE_VERIFICATION_FAILURE
```

2-66) Add the following at the end of 8.5.8.1:

If BioAPI_BSPAttachSecure (BioAPI 2.2 or greater) has been called, security operations shall be done according to the value set in the member of the parameter *SecurityProfileList* of ***BioAPI_BSPAttachSecure*** (BioAPI 2.2 or greater) corresponding to the BioAPI Unit which executes this function. If ACBio instance generation is requested, the challenge given as *Challenge* in the parameter *ACBioParameters* of ***BioAPI_BSPAttachSecure*** (BioAPI 2.2 or greater) shall be set into the field `controlValue` of type `ACBioContentInformation` in the ACBio instance. The current BPU IO index for the BSP Unit shall be set into the field `bpuIOIndex` of `bpuOuputExecutionInformationList` in the `biometricProcess` of `ACBioContentInformation` in the ACBio instance and also set into `bpuIOIndex` in `subBlockForACBio` in the security block. After that the current BPU IO index shall be incremented. The BIR shall be stored in type BioAPI_ASN1_BIR.

2-67) Add the following at the end of errors in 8.5.8.4:

```
BioAPIERR_SECURITY_ENCRYPTION_FAILURE

BioAPIERR_SECURITY_DECRYPTION_FAILURE

BioAPIERR_SECURITY_MAC_GENERATION_FAILURE

BioAPIERR_SECURITY_MAC_VERIFICATION_FAILURE

BioAPIERR_SECURITY_DIGITAL_SIGNATURE_GENERATION_FAILURE

BioAPIERR_SECURITY_DIGITAL_SIGNATURE_VERIFICATION_FAILURE
```

2-68) Add the following at the end of 8.5.9.1:

If BioAPI_BSPAttachSecure (BioAPI 2.2 or greater) has been called, security operations shall be done according to the value set in the member of the parameter *SecurityProfileList* of ***BioAPI_BSPAttachSecure*** (BioAPI 2.2 or greater) corresponding to the BioAPI Unit which executes this function. If ACBio instance generation is requested, the challenge given as *Challenge* in the parameter *ACBioParameters* of ***BioAPI_BSPAttachSecure*** (BioAPI 2.2 or greater) shall be set into the field `controlValue` of type `ACBioContentInformation` in the ACBio instance. The current BPU IO index for the BSP Unit shall be set into the field `bpuIOIndex` of `bpuOuputExecutionInformationList` in the `biometricProcess` of `ACBioContentInformation` in the ACBio instance and also set into `bpuIOIndex` in `subBlockForACBio` in the security block. After that the current BPU IO index shall be incremented. The BIR shall be stored in type BioAPI_ASN1_BIR.

2-69) Add the following at the end of errors in 8.5.9.4:

```
BioAPIERR_SECURITY_ENCRYPTION_FAILURE

BioAPIERR_SECURITY_DECRYPTION_FAILURE

BioAPIERR_SECURITY_MAC_GENERATION_FAILURE

BioAPIERR_SECURITY_MAC_VERIFICATION_FAILURE

BioAPIERR_SECURITY_DIGITAL_SIGNATURE_GENERATION_FAILURE

BioAPIERR_SECURITY_DIGITAL_SIGNATURE_VERIFICATION_FAILURE
```

2-70) Add the following text after 9.3.1.3:

9.3.1.3bis BioSPI_BSPAttachSecure (BioAPI 2.2)

```
BioAPI_RETURN BioAPI BioSPI_BSPAttachSecure
    (const BioAPI_UUID *BSPUuid,
    BioAPI_VERSION Version,
    const BioAPI_ACBio_PARAMETERS *ACBioParameters,
    const BioAPI_UNIT_LIST_ELEMENT *UnitList,
    const BioAPI_SECURITY_PROFILE *SecurityProfileList,
    uint32_t NumUnits,
    BioAPI_HANDLE BSPHandle);
```

9.3.1.3bis.1 Description

This function is invoked by the Framework once for each invocation of ***BioAPI_BSPAttachSecure*** (BioAPI 2.2 or greater) specifying the BSP identified by *BSPUuid*.

The biometric service provider shall verify compatibility with the version level specified by *Version*. If the version is not compatible, then this function fails. The BSP should perform all initializations required to support the new BSP invocation.

The BSP shall attach the specified BioAPI Units if they are supported.

NOTE: This is a sister function to ***BioAPI_BSPAttachSecure*** (BioAPI 2.2 or greater. See clause 8.1.7bis).

9.3.1.3bis.2 Parameters

BSPUuid (input) – a pointer to the UUID of the invoked biometric service provider.

Version (input) – The major and minor version number of the BioAPI specification that the application is expecting the BSP to support. The BSP shall determine whether it is compatible with the required version.

ACBioParameters (input/optional*)* – Parameters about ACBio generation. A NULL pointer shall be set if no generation of ACBio instances is necessary.

UnitList (input) – a pointer to a buffer containing a list of BioAPI_UNIT_LIST_ELEMENT structures indicating to the BSP which BioAPI Units (supported by the BSP) it is to use for this attach session. The structures contain the ID and category of each BioAPI Unit. One of the following will be specified for each category of BioAPI Unit, :

a. Selection of a specific BioAPI Unit: The particular BioAPI Unit to be used in this attach session is specified by inclusion of its ID and category.

b. Selection of any BioAPI Unit: When the UnitID is set to BioAPI_DONT_CARE in a particular element, the BSP will choose which BioAPI Unit of that category to use, or will give an error return if it does not support any BioAPI Units of that category. If a particular category is not listed, the BSP will likewise choose a BioAPI Unit of that category to use if it supports a BioAPI Unit of that category (however, there is no error return if it does not).

c. Selection of no BioAPI Unit: When the UnitID is set to BioAPI_DONT_INCLUDE, the BSP will explicitly not attach a BioAPI Unit of the given category, even if it supports one of that category.

NOTE: Any subsequent calls requiring use of a BioAPI Unit of this category will fail with an error return.

SecurityProfileList (input) – a pointer to a buffer containing a list of BioAPI_SECURITY_PROFILE structures which contain security information to be used in security operations by BioAPI Units. The order of this list shall keep that of the *UnitList*:

NumUnits (input) – The number of BioAPI Unit elements in the list that the pointer *UnitList* is pointing to. If this parameter contains "0", the BSP selects the BioAPI Unit for all categories of BioAPI Units that the BSP manages directly or indirectly.

BSPHandle (input) – The BioAPI_HANDLE value assigned by the Framework and associated with the attach session being created by this function.

NOTE: Only one BioAPI Unit of each category can be attached for each attach session at any time.

9.3.1.3bis.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.

9.3.1.3bis.4 Errors

See ***BioAPI_BSPAttachSecure*** (clause 8.1.7bis).

2-71) Replace the NOTE after 9.3.1.5 with the following:

NOTE 1: Details of the function definition are located in clause 8.1.9, ***BioAPI_QueryUnits***.

NOTE 2:The structure of BioAPI_UNIT_SCHEMA is dependent on the version of BioAPI. BioAPI_UNIT_SCHEMA of BioAPI 2.2 contains security information while that of BioAPI 2.1 or less does not.

2-72) Add the following after 9.3.4.4:

9.3.4.4bis BioSPI_ProcessUsingAuxBIRs (BioAPI 2.2)

```
BioAPI_RETURN BioAPI BioSPI_ProcessUsingAuxBIRs
    (BioAPI_HANDLE BSPHandle,
    const BioAPI_INPUT_BIR *CapturedBIR,
    uint32_t NumberOfAuxBIRs,
    const BioAPI_INPUT_BIR *AuxBIRs,
    const BioAPI_BIR_BIOMETRIC_DATA_FORMAT *OutputFormat,
    BioAPI_BIR_HANDLE *ProcessedBIR);
```

NOTE: Details of the function definition are located in clause 8.4.4bis, ***BioAPI_ProcessUsingAuxBIRs*** (BioAPI 2.2 or greater).

2-73) Add the following text after 9.3.4.5:

9.3.4.5bis BioSPI_ VerifyMatchUsingAuxBIRs (BioAPI 2.2)

```
BioAPI_RETURN BioAPI BioSPI_VerifyMatchUsingAuxBIRs
    (BioAPI_HANDLE BSPHandle,
    BioAPI_FMR MaxFMRRequested,
    const BioAPI_INPUT_BIR *ProcessedBIR,
    const BioAPI_INPUT_BIR *ReferenceTemplate,
    uint32_t NumberOfAuxBIRs,
    const BioAPI_INPUT_BIR *AuxBIRs,
    BioAPI_BIR_HANDLE *AdaptedBIR,
    BioAPI_BOOL *Result,
    BioAPI_FMR *FMRAchieved,
    BioAPI_BIR_HANDLE *ResultBIR,
    BioAPI_DATA *Payload);
```

NOTE: Details of the function definition are located in clause 8.4.5bis, ***BioAPI_ VerifyMatchUsingAuxBIRs*** (BioAPI 2.2 or greater).

9.3.4.5ter BioSPI_Decide (BioAPI 2.2)

```
BioAPI_RETURN BioAPI BioSPI_Decide
    (BioAPI_HANDLE BSPHandle,
    const BioAPI_INPUT_BIR *ScoreBIR,
    uint32_t NumberOfAuxBIRs,
    const BioAPI_INPUT_BIR *AuxBIRs,
    BioAPI_BIR_HANDLE *DecisionBIR);
```

NOTE: Details of the function definition are located in clause 8.4.6bis, ***BioAPI_Decide*** (BioAPI 2.2 or greater).

9.3.4.5quater BioSPI_Fuse (BioAPI 2.2)

```
BioAPI_RETURN BioAPI BioSPI_Fuse
    (BioAPI_HANDLE BSPHandle,
    uint32_t NumberOfSourceBIRs,
    const BioAPI_INPUT_BIR *SourceBIRs,
    uint32_t NumberOfAuxBIRs,
    const BioAPI_INPUT_BIR *AuxBIRs,
    BioAPI_BIR_HANDLE *FusionBIR);
```

NOTE: Details of the function definition are located in clause 8.4.6ter, ***BioAPI_Fuse*** (BioAPI 2.2 or greater).

2-74) Add the following after 9.3.4.10:

9.3.4.10bis BioSPI_Export (BioAPI 2.2)

```
BioAPI_RETURN BioAPI BioSPI_Export
   (BioAPI_HANDLE BSPHandle,
   BioAPI_BIR_HANDLE Handle,
   const BioAPI_BIR_BIOMETRIC_DATA_FORMAT *OutputBDBFormat,
   const BioAPI_BIR_SECURITY_BLOCK_FORMAT *OutputSBFormat,
   BioAPI_BIR *BIR);
```

NOTE: Details of the function definition are located in clause 8.4.9bis, ***BioAPI_Export*** (BioAPI 2.2 or greater).

2-75) Add the following NOTE before 11.1:

NOTE: There are additional error codes defined in ISO/IEC 29141 – Tenprint Capture Using BioAPI.

2-76) Add the following text after 11.2.7:

11.2.8 Security Error Codes (BioAPI 2.2)

```
#define    BioAPIERR_SECURITY_PROFILE_NOT_SET    (0x000601)
```

The parameter of the type `BioAPI_SECURITY_PROFILE` is NULL when ***BioAPI_BSPAttachSecure*** (BioAPI 2.2 or greater) is called.

```
#define    BioAPIERR_SECURITY_ENCRYPTION_ALG_NOT_SUPPORTED    (0x000610)
```

The specified encryption algorithm is not supported.

```
#define    BioAPIERR_SECURITY_ENCRYPTION_ALG_NOT_SET          (0x000611)
```

The encryption algorithm is not set though encryption is specified.

```
#define    BioAPIERR_SECURITY_ENCKEY_NOT_SET                  (0x000612)
```

The encryption key is not set though the encryption is specified.

```
#define    BioAPIERR_SECURITY_MAC_ALG_NOT_SUPPORTED       (0x000620)
```

The specified MAC algorithm is not supported.

```
#define    BioAPIERR_SECURITY_MAC_ALG_NOT_SET             (0x000621)
```

The MAC key is not set though the MAC is specified.

```
#define    BioAPIERR_SECURITY_MACKEY_NOT_SET              (0x000622)
```

The MAC algorithm is not set though MAC is specified.

```
#define    BioAPIERR_SECURITY_DIGITAL_SIGNATURE_ALG_NOT_SUPPORTED   (0x000640)
```

The specified digital signature algorithm is not supported.

```
#define    BioAPIERR_SECURITY_DIGITAL_SIGNATURE_ALG_NOT_SET         (0x000641)
```

The digital signature algorithm is not set though digital signature is specified.

```
#define    BioAPIERR_SECURITY_CHALLENGE_NOT_SET (0x000701)
```

The parameter `Challenge` is NULL or * `Challenge` is not set though ACBio instance generation is requested when ***BioAPI_BSPAttachSecure*** (BioAPI 2.2 or greater) is called.

```
#define    BioAPIERR_SECURITY_BPUIOINDEX_NOT_SET    (0x000702)
```

The parameter `InitialBPUIOIndexOutput` is NULL though ACBio instance generation is requested when ***BioAPI_BSPAttachSecure*** (BioAPI 2.2 or greater) is called.

```
#define    BioAPIERR_SECURITY_SUPREMUM_BPUIOINDEX_NOT_SET (0x000704)
```

The parameter `SupremumBPUIOIndexOutput` is NULL though ACBio instance generation is requested when ***BioAPI_BSPAttachSecure*** (BioAPI 2.2 or greater) is called.