



BSI Standards Publication

Information technology — Object oriented BioAPI

Part 4: C++ implementation

National foreword

This British Standard is the UK implementation of [ISO/IEC 30106-4:2019](#).

The UK participation in its preparation was entrusted to Technical Committee IST/44/-/2, Biometric technical interfaces.

A list of organizations represented on this committee can be obtained on request to its secretary.

This publication does not purport to include all the necessary provisions of a contract. Users are responsible for its correct application.

© The British Standards Institution 2019
Published by BSI Standards Limited 2019

ISBN 978 0 580 97429 8

ICS 35.240.15

Compliance with a British Standard cannot confer immunity from legal obligations.

This British Standard was published under the authority of the Standards Policy and Strategy Committee on 30 September 2019.

Amendments/corrigenda issued since publication

Date	Text affected
------	---------------

INTERNATIONAL STANDARD

ISO/IEC 30106-4

First edition
2019-09-15

Information technology — Object oriented BioAPI —

Part 4: C++ implementation

*Technologies de l'information — Objet orienté BioAPI —
Partie 4: Implémentation C++*



Reference number
ISO/IEC 30106-4:2019(E)

AVA Documents
Document Sharing Hub
© ISO/IEC 2019



COPYRIGHT PROTECTED DOCUMENT

© ISO/IEC 2019, Published in Switzerland

All rights reserved. Unless otherwise specified, no part of this publication may be reproduced or utilized otherwise in any form or by any means, electronic or mechanical, including photocopying, or posting on the internet or an intranet, without prior written permission. Permission can be requested from either ISO at the address below or ISO's member body in the country of the requester.

ISO copyright office
Ch. de Blandonnet 8 • CP 401
CH-1214 Vernier, Geneva, Switzerland
Tel. +41 22 749 01 11
Fax +41 22 749 09 47
copyright@iso.org
www.iso.org

Contents

	Page
Foreword	v
Introduction	vi
1 Scope	1
2 Normative references	1
3 Terms and definitions	1
4 BioAPI C++ general requirements	1
5 Data types and constants	2
5.1 Basic data types	2
5.1.1 Enumerations	2
5.1.2 BioAPIData	10
5.1.3 RegistryID	10
5.1.4 UUID	10
5.1.5 Date and Time	10
5.2 Class ACBioParameters	11
5.2.1 Description	11
5.2.2 Properties summary	11
5.3 Class BFPListElement	11
5.3.1 Description	11
5.3.2 Properties summary	11
5.4 Class BFPSchema	11
5.4.1 Description	11
5.4.2 Properties summary	11
5.5 Class BIR	12
5.5.1 Description	12
5.5.2 Properties summary	12
5.5.3 Method summary	13
5.6 Class BSPSchema	14
5.6.1 Description	14
5.6.2 Properties summary	14
5.7 Class candidate	15
5.7.1 Description	15
5.7.2 Properties summary	15
5.8 Class FrameworkSchema	15
5.8.1 Description	15
5.8.2 Properties summary	15
5.9 Class GUIBitmap	16
5.9.1 Description	16
5.9.2 Properties	16
5.10 Class Identifypopulation	16
5.10.1 Description	16
5.10.2 Properties summary	16
5.10.3 Method summary	16
5.11 Class PopulationMember	17
5.11.1 Description	17
5.11.2 Properties summary	17
5.12 Class SecurityProfileType	17
5.12.1 Description	17
5.12.2 Properties summary	17
5.13 Class UnitList	18
5.13.1 Description	18
5.13.2 Properties summary	18
5.13.3 Methods summary	18

5.14	Class UnitListElement.....	18
5.14.1	Description.....	18
5.14.2	Properties summary.....	18
5.15	Class UnitSchema.....	19
5.15.1	Description.....	19
5.15.2	Properties summary.....	19
6	Object oriented interfaces for supporting BioAPI_Units	20
6.1	General.....	20
6.2	Interface IArchive	20
6.2.1	Description.....	20
6.2.2	Method summary	20
6.3	Interface IComparison.....	25
6.3.1	Description.....	25
6.3.2	Method summary	26
6.4	Interface IProcessing.....	29
6.4.1	Description.....	29
6.4.2	Method summary	29
6.5	Interface ISensor	30
6.5.1	Description.....	30
6.5.2	Method summary	30
7	BFP level.....	32
7.1	Interface IBFP	32
7.1.1	Description.....	32
7.1.2	Imported interfaces.....	32
7.1.3	Properties summary.....	32
7.1.4	Events summary.....	32
7.1.5	Method summary	32
8	BSP level.....	35
8.1	Interface IBSP.....	35
8.1.1	Description.....	35
8.1.2	Imported interfaces.....	36
8.1.3	Properties summary.....	36
8.1.4	Events summary.....	36
8.1.5	Method summary	36
9	Framework level.....	44
9.1	Interface IComponentRegistry.....	44
9.1.1	Description.....	44
9.1.2	Method summary	44
9.2	Interface IFramework.....	46
9.2.1	Description.....	46
9.2.2	Inherited interfaces.....	46
9.2.3	Properties summary.....	46
9.2.4	Method summary	46
10	Application interaction.....	52
10.1	class BioAPIException : Exception.....	52
10.1.1	Description.....	52
10.1.2	Constructor summary	52
10.1.3	Properties summary.....	53
10.1.4	Method Summary.....	53
10.2	Callback functions.....	53
10.2.1	Description.....	53
10.2.2	Callback functions specification	54
	Annex A (informative) Calling sequence examples and sample code	61
	Bibliography	62

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.

The procedures used to develop this document and those intended for its further maintenance are described in the ISO/IEC Directives, Part 1. In particular, the different approval criteria needed for the different types of document should be noted. This document was drafted in accordance with the editorial rules of the ISO/IEC Directives, Part 2 (see www.iso.org/directives).

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. Details of any patent rights identified during the development of the document will be in the Introduction and/or on the ISO list of patent declarations received (see www.iso.org/patents) or the IEC list of patent declarations received (see <http://patents.iec.ch>).

Any trade name used in this document is information given for the convenience of users and does not constitute an endorsement.

For an explanation of the voluntary nature of standards, the meaning of ISO specific terms and expressions related to conformity assessment, as well as information about ISO's adherence to the World Trade Organization (WTO) principles in the Technical Barriers to Trade (TBT) see www.iso.org/iso/foreword.html.

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

A list of all parts in the ISO/IEC 30106 series can be found on the ISO website.

Any feedback or questions on this document should be directed to the user's national standards body. A complete listing of these bodies can be found at www.iso.org/members.html.

Introduction

In this document an application programming interface expressed in C++ language is specified. C++ is a basic, general-purpose, object-oriented programming language that is used in most computer-based platforms and operating systems.

One of the advantages of using C++ is that, as it is supported by most computer platforms, the development, in source code, may be fully (or at least to a great part) supported from one platform to another.

C++ is a programming language standardized by [ISO/IEC 14882](#), and most development platforms allow its use in conjunction to other programming languages and frameworks.

This document is drafted to make a clear and usable interpretation of [ISO/IEC 30106-1](#), when using C++ programming language.

Information technology — Object oriented BioAPI —

Part 4: C++ implementation

1 Scope

This document specifies an interface of a BioAPI C++ framework and BioAPI C++ BSP which will mirror the corresponding components specified in [ISO/IEC 30106-1](#). The semantic equivalence of this document will be maintained with [ISO/IEC 30106-2](#) (Java implementation) and [ISO/IEC 30106-3](#) (C# implementation). In spite of the differences in actual parameters passed between functions, the names and interface structure are the same.

2 Normative references

The following documents are referred to in the text in such a way that some or all of their content constitutes requirements 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 30106-1](#), *Information technology — Object oriented BioAPI — Architecture*

[ISO/IEC 30106-2](#), *Information technology — Object oriented BioAPI — Part 2: Java implementation*

[ISO/IEC 30106-3](#), *Information technology — Object oriented BioAPI — Part 3: C# implementation*

[ISO/IEC 14882](#), *Programming languages — C++*

3 Terms and definitions

No terms and definitions are listed in this document.

ISO and IEC maintain terminological databases for use in standardization at the following addresses:

- ISO Online browsing platform: available at <https://www.iso.org/obp>
- IEC Electropedia: available at <http://www.electropedia.org/>

4 BioAPI C++ general requirements

All classes defined in the scope of this document shall have a constructor with all its properties as parameters, as well as a destructor that will free all allocated memory. In addition, the definition of each of the classes may contain further requirements for the constructor.

The destructor shall fulfill at least, the same requirements that the corresponding `Dispose()` method defined in [ISO/IEC 30106-2](#) and [ISO/IEC 30106-3](#).

In addition, most implementations shall declare the classes as exportable. Therefore, in this document a constant called `BIOAPI` is defined, allowing exportation of the classes.

EXAMPLE In Windows the definition of `BIOAPI` is: `#define BIOAPI __declspec(dllexport)`

To illustrate the use of the specification given in this document, refer to [Annex A](#).

5 Data types and constants

5.1 Basic data types

5.1.1 Enumerations

5.1.1.1 BiometricSubtype	
<u>Description:</u>	Subtype of the biometric data used (e.g. which finger used in finger modalities). When transferring this information into/from a binary format, the Biometric Subtype constants defined in ISO/IEC 30106-1 shall be used.
<u>Enum Constant Summary:</u>	— <i>NoValueAvailable</i> — <i>Left</i> — <i>Right</i> — <i>LeftThumb</i> — <i>LeftIndexFinger</i> — <i>LeftMiddleFinger</i> — <i>LeftRingFinger</i> — <i>LeftLittleFinger</i> — <i>RightThumb</i> — <i>RightIndexFinger</i> — <i>RightMiddleFinger</i> — <i>RightRingFinger</i> — <i>RightLittleFinger</i> — <i>LeftPalm</i> — <i>LeftBackOfHand</i> — <i>LeftWrist</i> — <i>RightPalm</i> — <i>RightBackOfHand</i> — <i>RightWrist</i>

5.1.1.2 BiometricType	
<u>Description:</u>	Type of the biometric data used (e.g. modality). When transferring this information into/from a binary format, the Biometric Type constants defined in ISO/IEC 30106-1 shall be used.
<u>Enum Constant Summary:</u>	— <i>NoValueAvailable</i> — <i>MultipleBiometricTypes</i> — <i>Face</i> — <i>Voice</i> — <i>Finger</i> — <i>Iris</i> — <i>Retina</i> — <i>HandGeometry</i> — <i>SignatureOrSign</i> — <i>Keystroke</i> — <i>LipMovement</i> — <i>Gait</i> — <i>Vein</i> — <i>DNA</i> — <i>Ear</i> — <i>Foot</i> — <i>Scent</i>

5.1.1.3 BIRDatabaseAccess	
<u>Description:</u>	Defines the access mode to the database
<u>Enum Constant Summary:</u>	— <i>Read</i> – access mode which allows only retrieval of records. — <i>ReadWrite</i> – access mode which allows addition, deletion and retrieval of records. — <i>Write</i> – access mode which allows addition and deletion of records, but retrieval is not allowed

5.1.1.4 BSPSchemaOperations

<u>Description:</u>	Enumerates the different operations that a BSP can offer to the biometric application (see 5.6)
<u>Enum Constant Summary:</u>	— <i>CalibrateSensor (0x00020000)</i> — <i>Capture (0x00000004)</i> — <i>CheckQuality (0x00080000)</i> — <i>ControlUnit (0x00400000)</i> — <i>CreateTemplate (0x00000008)</i> — <i>CreateTemplateWithAuxBIR (0x00000020)</i> — <i>EnableEvents (0x00000001)</i> — <i>Enrol (0x00000100)</i> — <i>GetIndicatorStatus (0x00010000)</i> — <i>Identify (0x00000080)</i> — <i>IdentifyAggregate (0x00000400)</i> — <i>PresetIdentifyPopulation (0x00001000)</i> — <i>Process (0x00000010)</i> — <i>ProcessWithAuxBIR (0x01000000)</i> — <i>QueryBFPs (0x00200000)</i> — <i>QueryUnits (0x00100000)</i> — <i>Security (0x10000000)</i> — <i>SetIndicatorStatus (0x00008000)</i> — <i>SetPowerMode (0x00004000)</i> — <i>Verify (0x00000040)</i> — <i>VerifyAggregated (0x00000200)</i> — <i>VerifyWithAuxBIR (0x02000000)</i>

5.1.1.5 BSPSchemaOptions

<u>Description:</u>	Enumerates the different options that can handle a BSP (see 5.6)
<u>Enum Constant Summary:</u>	— <i>Adaptation (0x00000800)</i> — <i>AppGUI (0x00000010)</i> — <i>ArchiveBFP (0x00020000)</i> — <i>Binning (0x00001000)</i> — <i>BirEncrypt (0x00000200)</i> — <i>BirSign (0x00000100)</i> — <i>CaptureMultiple (0x00400000)</i> — <i>CoarseScores (0x00100000)</i> — <i>ComparisonBFP (0x00040000)</i> — <i>GUIProgressEvents (0x00000020)</i> — <i>IdentifyIndicator (0x00200000)</i> — <i>OCC (0x00004000) (on-card comparison, formerly known as MOC)</i> — <i>AdditionalData (0x00000080)</i> — <i>ProcessingBFP (0x00080000)</i> — <i>ProcessMultiple (0x00800000)</i> — <i>QualityIntermediate (0x00000004)</i> — <i>QualityProcessed (0x00000008)</i> — <i>QualityRaw (0x00000002)</i> — <i>Raw (0x00000001)</i> — <i>SelfContainedDevice (0x00002000)</i> — <i>SensorBFP (0x00010000)</i> — <i>SourcePresent (0x00000040)</i> — <i>SubtypeToCapture (0x00008000)</i> — <i>TemplateUpdate (0x00000400)</i> — <i>Disabilities</i> — <i>PADFeature</i>

5.1.1.6 EventKind	
<u>Description:</u>	Defines the kind of sources that can originate an event
<u>Enum Constant Summary:</u>	— <i>Insert (0x00000001)</i> — <i>Remove (0x00000002)</i> — <i>Fault (0x00000004)</i> — <i>SourcePresent (0x00000008)</i> — <i>SourceRemoved (0x00000010)</i>

5.1.1.7 Facility	
<u>Description:</u>	Defines originator of the error in an exception (see 10.1)
<u>Enum Constant Summary:</u>	— <i>Framework – The error was reported by the framework component.</i> — <i>BSP – The error was reported by the biometric service provider.</i> — <i>Unit – The error was reported by the biometric unit</i>

5.1.1.8 GUIEnrolType	
<u>Description:</u>	Indicates the enrol type of a BSP (see 10.2)
<u>Enum Constant Summary:</u>	— <i>TestVerify</i> — <i>MultipleCapture</i>

5.1.1.9 GUIMoment	
<u>Description:</u>	Determines the moment when the processing of an operation is at the time of calling a GUI callback function (see 10.2)
<u>Enum Constant Summary:</u>	— <i>BeforeStart</i> — <i>AfterEnd</i>

5.1.1.10 GUIOperation	
<u>Description:</u>	Determines the operation being performed when using GUI callback functions (see 10.2)
<u>Enum Constant Summary:</u>	— <i>Capture</i> — <i>Enrol</i> — <i>Identify</i> — <i>Verify</i>

5.1.1.11 GUIResponse	
<u>Description:</u>	Enumeration of the possible actions to be performed by a BSP after a GUI event notification callback made by the BSP has returned control to the BSP (see 10.2)
<u>Enum Constant Summary:</u>	— <i>CycleStart</i> — <i>CycleRestart</i> — <i>Default</i> — <i>OpComplete</i> — <i>OpCancel</i> — <i>ProgressContinue</i> — <i>ProgressAbort</i> — <i>Recapture</i> — <i>SubOpStart</i> — <i>SubOpNext</i>

5.1.1.12 GUISuboperation	
<u>Description:</u>	An enumeration of the possible types of suboperations performed by a BSP during an operation, to be reported to the application in GUI event notifications (see 10.2)
<u>Enum Constant Summary:</u>	— <i>Capture</i> — <i>CreateTemplate</i> — <i>Identify</i> — <i>Process</i> — <i>Verify</i>

5.1.1.13 ProcessedLevel

<u>Description:</u>	Determines the level of processing of the BIR
<u>Enum Constant Summary:</u>	— <i>Intermediate</i> — <i>Processed</i> — <i>Raw</i>

5.1.1.14 Purpose

<u>Description:</u>	Defines the purpose for which the BIR or process is intended
<u>Enum Constant Summary:</u>	— <i>Verify</i> — <i>Identify</i> — <i>Enrol</i> — <i>EnrolForVerificationOnly</i> — <i>EnrolForIdentificacionOnly</i> — <i>Audit</i> — <i>Decide</i> — <i>NoPurposeAvailable</i>

5.1.1.15 ResultOptions

<u>Description:</u>	Defines the request to some BioAPI methods, to provide additional results to the originally defined (e.g. see 6.3.2).
<u>Enum Constant Summary:</u>	— <i>RequestAdaptedBIR</i> – Request that a BIR be constructed by adapting the reference template using the processed BIR that is the input to the biometric verification. — <i>RequestAdditionalData</i> – Request that the additionalData should be returned upon successful verification — <i>RequestAdditionalData</i> – Request additional data to be used, for example, in an auditing process.

5.1.1.16 SecurityOptionsType	
<u>Description:</u>	Defines which security options are supported by the BioAPI_Unit
<u>Enum Constant Summary:</u>	— <i>Encryption (0x00000001)</i> – Indicates that the BioAPI Unit supports encryption. — <i>MAC (0x00000002)</i> – Indicates that the BioAPI Unit supports MAC generation. — <i>DigitalSignature (0x00000004)</i> – Indicates that the BioAPI Unit supports digital signature. — <i>ACBioGenerationWithMAC (0x00000010)</i> – Indicates that the BioAPI Unit supports ACBio generation using MAC. — <i>ACBioGenerationWithDigitalSignature (0x00000020)</i> – Indicates that the BioAPI Unit supports ACBio generation using digital signature.

5.1.1.17 UnitCategoryType	
<u>Description:</u>	List the different categories for a BioAPI_Unit
<u>Enum Constant Summary:</u>	— <i>Archive</i> – the unit manages BSP's BIR database. (0x00000001) — <i>Comparison</i> – the unit is the collection of comparison algorithms. (0x00000002) — <i>Processing</i> – the unit is the collection of processing algorithms. (0x00000004) — <i>Sensor</i> – the unit manages hardware sensor (0x00000008) — <i>QualityAssessment</i> – the unit is the collection of the different quality assessment processes (0x00000010)

5.1.1.18 UnitIndicatorStatus	
<u>Description:</u>	Defines possible values for the indicator status
<u>Enum Constant Summary:</u>	— <i>Accept</i> — <i>Busy</i> — <i>Failure</i> — <i>Ready</i> — <i>Reject</i>

5.1.1.19 UnitPowerMode

<u>Description:</u>	Defines possible unit power modes
<u>Enum Constant Summary:</u>	— <i>Detect – mode when unit is able to detect interaction of subject with the sensor.</i> — <i>Normal – mode when all functions are available.</i> — <i>Sleep – minimum mode. All functions off</i>

5.1.2 BioAPIData

5.1.2.1 Description

Defines generic data to be exchanged within Object Oriented BioAPI.

5.1.2.2 C++ definition

```
typedef vector<unsigned char> BIOAPI BioAPIData;
```

5.1.3 RegistryID

5.1.3.1 Description

Defines the identification of the data to be used or the product identification. It is a structure with two components, as defined in the different fields of [ISO/IEC 19785](#) Biometric Information Records (BIRs)

5.1.3.2 Properties summary

- unsigned short Owner
- unsigned short Type

5.1.3.3 C++ definition

```
typedef struct BIOAPI RegistryID {  
    unsigned short owner;  
    unsigned short type;  
} RegistryID;
```

5.1.4 UUID

5.1.4.1 Description

Throughout this document, there is a need to use Unique Identifiers, either for components or for users. This document defines the Universally Unique Identifier (UUID) as an array of 16 bytes.

5.1.4.2 C++ definition

```
typedef array<unsigned char, 16> BIOAPI UUID;
```

5.1.5 Date and Time

In the scope of this document, date and time will be handled by the tm structure contained in the standard ctime library.

EXAMPLE

```
#include <ctime>
tm BDBCreationDate;
```

5.2 Class ACBioParameters

5.2.1 Description

The structure that provides the information which is used to generate ACBio instances.

5.2.2 Properties summary

- vector<int> Challenge – Challenge from the validator of a biometric verification when ACBio is used. This value shall be sent to the field controlValue of type ACBioContentInformation in ACBio instances.
- vector<int> 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 accepted by the BSP, and assigned to the BSP Units and BFPs.
- vector<int> 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.

5.3 Class BFPListElement

5.3.1 Description

Identifies a BFP by category and UUID. A list is returned by a BSP when queried for the installed BFPs that it supports.

5.3.2 Properties summary

- UnitCategoryType UnitCategory: The category of the unit.
- UUID BFPID: The UUID assigned to the BFP.

5.4 Class BFPSchema

5.4.1 Description

Represents the record in the component registry that defines the properties of the BFP installed in the system. Is a serializable class.

5.4.2 Properties summary

- UUID BFPUUID: UUID of the BFP.
- UnitCategoryType BFPCategory: Category of the BFP identified by the BFPUUID.
- string BFPDescription: A NULL-terminated string containing a text description of the BFP.
- string Path: A pointer to a NULL-terminated string containing the path of the file containing the BFP executable code, including the filename. The path may be a URL. This string shall consist of [ISO/IEC 10646](#) characters encoded in UTF-8 (see ISO/IEC 10646:2017, Annex D). When BFPSchema is used within a function call, the component that receives the call allocates the memory for the Path schema element and the calling component frees the memory.
- string SpecVersion: Major/minor version number of the BioAPI specification to which the BFP was implemented.
- string ProductVersion: The version string of the BFP software.

- string Vendor: A NULL-terminated string containing the name of the BFP vendor.
- svector<unsigned char> BFPPProperty.
- vector<RegistryID> BFPSupportedFormats: A list the data formats that are supported by the BFP (see [7.1](#)).
- vector<BiometricType> FactorsMask: A list of the biometric types supported by the BFP (see [7.1](#)).
- UUID FwPropertyID: UUID of the format of the following BFP property.
- vector<unsigned char> FwProperty: Address and length of a memory buffer containing the BFP property. The format and content of the BFP property can either be specified by a vendor or can be specified in a related standard.

5.5 Class BIR

5.5.1 Description

This interface represents BIRs. It supports [ISO/IEC 19785](#) definitions, both for Simple-BIRs or for Complex-BIRs. The specification of the patron format that shall be used is given in [ISO/IEC 30106-1](#).

5.5.2 Properties summary

- RegistryID SelfID;
- unsigned char CBEFFVersion;
- unsigned char PatronHeaderVersion;
- RegistryID BDBFormat;
- bool BDBEncryption;
- bool BIRIntegrity;
- BiometricType BDBBiometricType;
- BiometricSubtype BDBBiometricSubtype;
- RegistryID BDBCaptureDevice;
- RegistryID BDBFeatureExtractionAlg;
- RegistryID BDBComparisonAlg;
- RegistryID BDBCompresionAlg;
- RegistryID BDBPADTechnique;
- vector<unsigned char> BDBChallengeResponse;
- tm BDBCreationDate;
- vector<unsigned char> BDBIndex;
- ProcessedLevel BDBProcessedLevel;
- RegistryID BDBProduct;
- Purpose BDBPurpose;
- unsigned char BDBQuality;

- RegistryID BDBQualityAlg;
- vector<tm> BDBValidityPeriod;
- tm BIRCreationDate;
- vector<unsigned char> BIRCreator;
- vector<unsigned char> BIRIndex;
- vector<unsigned char> BIRPayload;
- vector<unsigned char> BIRPointer;
- vector<tm> BIRValidityPeriod;
- RegistryID SBFormat;
- vector<unsigned char> BDBData;
- vector<unsigned char> SBData.

5.5.3 Method summary

5.5.3.1 BIR (vector<unsigned char> record)	
<u>Description:</u>	Constructs the BIR data from a byte array coded as an ISO/IEC 19785 self-identifying record
<u>Parameters:</u>	— <i>record</i> : the byte array containing the CBEFF record
<u>Exception:</u>	If the input parameters are invalid, the format is not supported or operation fails due to error. BioAPIException (see 10.1)

5.5.3.2 BIR(RegistryID bDBFormat, bool bDBEncryption, bool bIRIntegrity, BiometricType bDBBiometricType, BiometricSubtype bDBBiometricSubtype, RegistryID bDBCaptureDevice, RegistryID bDBFeatureExtractionAlg, RegistryID bDBComparisonAlg, RegistryID bDBCompressionAlg, RegistryID bDBPADTechnique, vector<unsigned char> bDBChallengeResponse, tm bDBCreationDate, vector<unsigned char> bDBIndex, ProcessedLevel bDBProcessedLevel, RegistryID bDBProduct, Purpose bDBPurpose, unsigned char bDBQuality, RegistryID bDBQualityAlg, vector<tm> bDBValidityPeriod, tm bIRCreationDate, vector<unsigned char> bIRCreator, vector<unsigned char> bIRIndex, vector<unsigned char> bIRPayload, vector<unsigned char> bIRPointer, vector<tm> bIRValidityPeriod, RegistryID SBFormat, vector<unsigned char> bDBData, vector<unsigned char> SBData);	
<u>Description:</u>	Constructs the BIR data from its individual components
<u>Parameters:</u>	— Each of the properties in the BIR class
<u>Exception:</u>	If the input parameters are invalid, the format is not supported or operation fails due to error. BioAPIException (see 10.1)

5.5.3.3 vector<unsigned char> ToArray()

Description:	Serializes a BIR record in order to provide it as a byte array representing the CBEFF information
Return Value:	The byte array containing the CBEFF information
Exception:	If the input parameters are invalid, the format is not supported or operation fails due to error. BioAPIException (see 10.1)

5.6 Class BSPSchema

5.6.1 Description

Represents the record in the component registry that defines the properties of the BSP installed in the system. Is a serializable class.

5.6.2 Properties summary

- UUID BSPUUID.
- String BSPDescription: A NULL-terminated string containing a text description of the BSP.
- string Path: A pointer to a NULL-terminated string containing the path of the file containing the BSP executable code, including the filename. The path may be a URL. This string shall consist of [ISO/IEC 10646](#) characters encoded in UTF-8 (see ISO/IEC 10646:2017, Annex D). When BioAPI_BSP_SCHEMA is used within a function call, the component that receives the call allocates the memory for the Path schema element and the calling component frees the memory.
- string SpecVersion: Major/minor version number of the BioAPI specification to which the BSP was implemented.
- string ProductVersion: The version string of the BSP software.
- string Vendor: A NULL-terminated string containing the name of the BSP vendor.
- vector<RegistryID> BSPSupportedFormats: A list the data formats that are supported by the BSP (see [5.1.3](#)).
- vector<BiometricType> FactorsMask: A list of the biometric types supported by the BSP (see 5.1.1.2).
- vector<BSPSchemaOperations> Operations: A list of the biometric operations supported by the BSP (see 4.1.1.4).
- vector<BSPSchemaOptions> Options: A list of the biometric options supported by the BSP (see 5.1.1.5).
- int AdditionalDataPolicy: Threshold setting (maximum FMR value) used to determine when to release additionalData after successful verification.
- int MaxAdditionalDataSize: Maximum additionalData size (in bytes) that the BSP can accept.
- int DefaultVerifyTimeout: Default timeout value in milliseconds used by the BSP for Verify operations when no timeout is specified by the application.
- int DefaultIdentifyTimeout: Default timeout value in milliseconds used by the BSP for Identify and BioAPI_IdentifyMatch operations when no timeout is specified by the application.
- int DefaultCaptureTimeout: Default timeout value in milliseconds used by the BSP for Capture operations when no timeout is specified by the application.
- int DefaultEnrolTimeout: Default timeout value in milliseconds used by the BSP for Enrol operations when no timeout is specified by the application.