

BS ISO/IEC 30106-2:2016



BSI Standards Publication

Information technology — Object oriented BioAPI

Part 2: Java implementation

bsi.

...making excellence a habit.
AVA Documents™
Document Sharing Hub

National foreword

This British Standard is the UK implementation of ISO/IEC 30106-2:2016.

The UK participation in its preparation was entrusted to Technical Committee IST/44, Biometrics.

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 2016.
Published by BSI Standards Limited 2016

ISBN 978 0 580 75178 3

ICS 35.040

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 April 2016.

Amendments/corrigenda issued since publication

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

INTERNATIONAL STANDARD

BS ISO/IEC 30106-2:2016

**ISO/IEC
30106-2**

First edition
2016-03-15

Information technology — Object oriented BioAPI —

Part 2: Java implementation

*Technologies de l'information — Objet orienté BioAPI —
Partie 2: Mise en oeuvre Java*



Reference number
ISO/IEC 30106-2:2016(E)

AVA Documents
Document Sharing Hub
© ISO/IEC 2016



COPYRIGHT PROTECTED DOCUMENT

© ISO/IEC 2016, 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	vi
Introduction	vii
1 Scope	1
2 Normative references	1
3 BioAPI Java package structure	1
3.1 Package org.bioapi	1
3.1.1 Package description	1
3.1.2 Structure	1
3.2 Package org.bioapi.data	2
3.2.1 Package description	2
3.2.2 Structure	2
4 Data types and constants	2
4.1 Class ACBioParameters	2
4.1.1 Description	2
4.1.2 Method summary	2
4.2 Class BFPListElement	2
4.2.1 Description	2
4.2.2 Method summary	3
4.3 Class BFPSchema	3
4.3.1 Description	3
4.3.2 Method summary	3
4.4 Class BIR	4
4.4.1 Description	4
4.4.2 Method summary	4
4.5 Class BSPSchema	9
4.5.1 Description	9
4.5.2 Method summary	9
4.6 Class candidate	12
4.6.1 Description	12
4.6.2 Method summary	12
4.7 Class DataTypes	13
4.7.1 Description	13
4.7.2 Enumerations	14
4.8 Class date	20
4.8.1 Description	20
4.8.2 Method summary	21
4.9 Class FrameworkSchema	22
4.9.1 Description	22
4.9.2 Method summary	23
4.10 Class GUIBitmap	24
4.10.1 Description	24
4.10.2 Method summary	24
4.11 Class IdentifyPopulation	24
4.11.1 Description	24
4.11.2 Method summary	24
4.12 Class PopulationMember	25
4.12.1 Description	25
4.12.2 Method summary	25
4.13 Class RegistryID	25
4.13.1 Description	25
4.13.2 Method summary	25
4.14 Class SecurityProfileType	26
4.14.1 Description	26

4.14.2	Method summary	26
4.15	Class UnitList	27
4.15.1	Description	27
4.15.2	Method summary	27
4.16	Class UnitListElement	27
4.16.1	Description	27
4.16.2	Method summary	27
4.17	Class UnitSchema	28
4.17.1	Description	28
4.17.2	Method summary	28
4.18	Class UUID	30
4.18.1	Description	30
5	Object oriented interfaces for supporting BioAPI_Units	30
5.1	General	30
5.2	Interface archive	30
5.2.1	Description	30
5.2.2	Method summary	31
5.3	Interface comparison	34
5.3.1	Description	34
5.3.2	Method summary	34
5.4	Interface processing	36
5.4.1	Description	36
5.4.2	Method summary	36
5.5	Interface sensor	37
5.5.1	Description	37
5.5.2	Method summary	38
6	BFP level	39
6.1	Interface BFP	39
6.1.1	Description	39
6.1.2	Imported interfaces	39
6.1.3	Method summary	40
7	BSP level	42
7.1	Interface BSP	42
7.1.1	Description	42
7.1.2	Imported interfaces	42
7.1.3	Method summary	42
8	Framework level	49
8.1	Interface ComponentRegistry	49
8.1.1	Description	49
8.1.2	Method summary	50
8.2	Interface framework	51
8.2.1	Description	51
8.2.2	Inherited interfaces	51
8.2.3	Method summary	52
9	Application interaction	56
9.1	class BioAPIException extends Exception	56
9.1.1	Description	56
9.1.2	Constructor summary	56
9.1.3	Method summary	57
9.2	GUI callback functions	57
9.2.1	Description	57
9.2.2	Callback interface specification	58
10	BSP Interaction	61
10.1	Interface BSPEventListener	61
10.1.1	Method summary	61

11	BFP interaction	62
11.1	Interface BFPEnumerationListener	62
11.1.1	Method summary	62
11.2	Interface BFPEventListener	62
11.2.1	Method summary	62
11.3	Interface BFPGUIProgressEventListener	63
11.3.1	Method summary	63
Annex A	(informative) Java requirements	64
Annex B	(informative) Calling sequence examples and sample code	65

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.

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).

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

For an explanation on the meaning of ISO specific terms and expressions related to conformity assessment, as well as information about ISO's adherence to the WTO principles in the Technical Barriers to Trade (TBT) see the following URL: [Foreword - Supplementary information](#)

The committee responsible for this document is ISO/TC JTC1, *Information technology*, Subcommittee SC 37, *Biometrics*.

ISO/IEC 30106 consists of the following parts, under the general title *Information technology — Object oriented BioAPI*:

- *Part 1: Architecture*
- *Part 2: Java implementation*
- *Part 3: C# implementation*

Introduction

In this part of ISO/IEC 30106, an application programming interface expressed in Java language is specified. Java is intended to be a simple, general-purpose, object oriented programming language that is aimed at enabling programmers to quickly build a wide range of applications for multiple platforms.

This Java implementation allows an easy use of Java BSPs, Java-based application servers or Java applets. Therefore, it is the best way to write desktop and web applications/services and this specification provides an advanced and well-designed remote framework.

Although the best practices of Java programming states that variables should be written in smallcase letters, in the case of symbols, such as BSP or BFPs, it has been kept as uppercase letters.

Information technology — Object oriented BioAPI —

Part 2: Java implementation

1 Scope

This part of ISO/IEC 30106 specifies an interface of a BioAPI Java framework and BioAPI Java BSP, which will mirror the corresponding components, specified in ISO/IEC 30106-1. The semantic equivalent of this standard is maintained in this part of ISO/IEC 30106.

2 Normative references

The following documents, in whole or in part, are normatively referenced in this document and are indispensable for its application. 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 — BioAPI for object oriented programming languages — Part 1: Architecture*

3 BioAPI Java package structure

The BioAPI Java interface will be divided into several packages. The following is the package structure:

- package org.bioapi: contains functionality to manage units, BSPs, BFPs, the Framework and Applications;
- package org.bioapi.data: contains all the data structures.

3.1 Package org.bioapi

3.1.1 Package description

This package contains all the components responsible for managing and executing the functionality of BioAPI. Component Registry interface is also defined in this package.

3.1.2 Structure

The description of this namespace is given explaining a bottom-up structure. In [Clause 4](#), the interfaces needed to be implemented for each of the Unit types are explained. It is important to note that such interfaces do not refer to an implemented class by itself, as the accessible class will either be the Biometric Service Provider (BSP) or the Biometric Function Provider (BFP), but the specifications in such clause are common to the methods and properties to be added to the implemented BSP and/or BFP classes.

This will be followed by the specification of the implementation of the BFP ([Clause 5](#)) and BSP ([Clause 6](#)) interfaces. These two interfaces provide the lower layer interoperability level, equivalent to the SPI and BFPI interfaces in ISO/IEC 19784-1.

The higher layer of interoperability level is provided by the specification of the Framework ([Clause 7](#), with the Framework Interface and the Component Registry) and the Application interaction ([Clause 8](#),

with the specification of the Exceptions and Callback functions). This provides the equivalence to the API interface in ISO/IEC 19784-1.

3.2 Package org.bioapi.data

3.2.1 Package description

This package contains all data structures needed for the implementation of OO BioAPI.

3.2.2 Structure

Several data structures are provided to comply with the requirements specified in this part of ISO/IEC 30106. All the BioAPI.Data namespace is specified in [Clause 3](#), where all needed classes and enumerations are defined. This has to be complemented to the constants defined in ISO/IEC 30106-1.

4 Data types and constants

4.1 Class ACBioParameters

4.1.1 Description

Structure that provides the information that is used to generate ACBio instances.

4.1.2 Method summary

4.1.2.1	int[] getChallenge()
Description:	Return the 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.
Return value:	The challenge from the validator of a biometric verification when ACBio is used.

4.1.2.2	int[] getInitialBPUIOIndexOutput()
Description:	Return 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 inside the BSP and assigned to the BSP Units and BSPs.
Return value:	The initial value of BPU IO index.

4.1.2.3	int[] getSupremumBPUIOIndexOutput()
Description:	Return 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.
Return value:	The supremum of BPU IO index.

4.2 Class BFPListElement

4.2.1 Description

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

4.2.2 Method summary

4.2.2.1	UUID getBFPID()
Description:	Return the UUID assigned to the BFP.
Return value:	UUID assigned to the BFP.

4.2.2.2	UnitCategoryType getUnitCategory()
Description:	Return the category of the units.
Return value:	Category of the units.

4.2.2.3	void setBFPID(UUID bfpID)
Description:	Set the UUID assigned to the BFP.
Parameters:	<i>bfpID</i> : UUID assigned to the BFP.

4.2.2.4	void setUnitCategory(UnitCategoryType unitCategory)
Description:	Set the category of the units.
Parameters:	<i>unitCategory</i> : category of the units.

4.3 Class BFPSchema

4.3.1 Description

Represents the record in the component registry that defines the properties of the BFP installed in the system.

4.3.2 Method summary

4.3.2.1	String getBFPDescription()
Description:	Return a string containing a text description of the BFP.
Return value:	A string containing a text description of the BFP.

4.3.2.2	Vector<RegistryID> getBFPSupportedFormats()
Description:	Return a list the data formats that are supported by the BFP.
Return value:	A list the data formats that are supported by the BFP.

4.3.2.3	UUID getBFPUUID()
Description:	Return the BFP UUID
Return value:	BFP UUID

4.3.2.4	Vector<BiometricType> getFactorsMask()
Description:	Return a list of the biometric types supported by the BFP.
Return value:	A list of the biometric types supported by the BFP.

4.3.2.5	byte[] getFWProperty()
Description:	Return the 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.
Return value:	The address and length of a memory buffer containing the BFP property.

4.3.2.6	UUID getFWPropertyID()
Description:	Return the UUID of the format of the following BFP property.
Return value:	UUID of the format of the following BFP property.

4.3.2.7	String getPath()
Description:	Return a pointer to a 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 10646:2014, 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.
Return value:	A pointer to a string containing the path of the file containing the BFP executable code, including the filename.

4.3.2.8	String getProductVersion()
Description:	Return the version string of the BFP software.
Return value:	The version string of the BFP software.

4.3.2.9	String getSpecVersion()
Description:	Return the major/minor version number of the BioAPI specification to which the BFP was implemented.
Return value:	The major/minor version number of the BioAPI specification to which the BFP was implemented.

4.3.2.10	UnitCategoryType getUnitCategory()
Description:	Return the category of the BFP identified by the BFP UUID.
Return value:	The category of the BFP identified by the BFP UUID.

4.3.2.11	String getVendor()
Description:	Return a string containing the name of the BFP vendor.
Return value:	A string containing the name of the BFP vendor.

4.4 Class BIR

4.4.1 Description

This interface represents Biometric Information Records (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.

4.4.2 Method summary

4.4.2.1	void birFromArray(byte[] record)
Description:	Fills in the BIR data from a byte array coded as ISO/IEC 19785 record.
Parameters:	<i>record</i> : the byte array containing the CBEFF record.
Exception:	If the input parameters are invalid, the format is not supported or operation fails due to error. BioAPIException (see 9.1)

4.4.2.2	byte[] birToByteArray()
Description:	Serializes a BIR record so as 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 9.1)

4.4.2.3	void destroy()
Description:	Removes all the information in the current BIR, leaving it empty for a next use.
Exception:	None

4.4.2.4	BiometricSubtype getBDBBiometricSubtype()
Description:	Return the BDB biometric subtype
Return value:	The BDB biometric subtype

4.4.2.5	BiometricType getBDBBiometricType()
Description:	Return the BDB biometric type
Return value:	The BDB biometric type

4.4.2.6	byte[] getBDBChallengeResponse()
Description:	Return the BDB challenge response
Return value:	The BDB challenge response

4.4.2.7	Date getBDBCreationDate()
Description:	Return the BDB creation date
Return value:	The BDB creation date

4.4.2.8	byte[] getBDBData()
Description:	Return the BDB data array
Return value:	The BDB data array

4.4.2.9	RegistryID getBDBFormat()
Description:	Return the format of the BDB data
Return value:	The format of the BDB data

4.4.2.10	byte[] getBDBIndex()
Description:	Return the BDB index
Return value:	The BDB index

4.4.2.11	ProcessedLevel getBDBProcessedLevel()
Description:	Return the BDB processed level
Return value:	The BDB processed level

4.4.2.12	Purpose getBDBPurpose()
Description:	Return the BDB purpose
Return value:	The BDB purpose

4.4.2.13	byte getBDBQuality()
Description:	Return the BDB quality
Return value:	The BDB quality

4.4.2.14	Vector<Date> getBDBValidityPeriod()
Description:	Return the BDB validity period
Return value:	The BDB validity period

4.4.2.15	Date getBIRCreationDate()
Description:	Return the BIR creation date
Return value:	The BIR creation date

4.4.2.16	byte[] getBIRCreator()
Description:	Return the BIR creator array
Return value:	The BIR creator array

4.4.2.17	byte[] getBIRIndex()
Description:	Return the BIR index
Return value:	The BIR index

4.4.2.18	byte[] getBIRAdditionalData()
Description:	Return the BIR additionalData
Return value:	The BIR additionalData

4.4.2.19	Vector<Date> getBIRValidityPeriod()
Description:	Return the BIR validity period
Return value:	The BIR validity period

4.4.2.20	byte getCBEFFVersion()
Description:	Return the version of the CBEFF component
Return value:	The version of the CBEFF component

4.4.2.21	RegistryID getPatronFormat()
Description:	Return the patron format
Return value:	The patron format

4.4.2.22	byte getPatronHeaderVersion()
Description:	Return the header version
Return value:	The header version

4.4.2.23	byte[] getSBData()
Description:	Return the Security Block data array
Return value:	The Security Block data array

4.4.2.24	RegistryID getSBFormat()
Description:	Return the Security Block format
Return value:	The Security Block format

4.4.2.25	boolean hasBDBEncryption()
Description:	Return true if the BDB is encrypted, false otherwise.
Return value:	True if the BDB is encrypted, false otherwise.

4.4.2.26	boolean hasBDBIntegrity()
Description:	Return true if the BDB has integrity, false otherwise.
Return value:	True if the BDB has integrity, false otherwise.

4.4.2.27	boolean isBIRSigned()
4.4.2.28	boolean isQualitySupported()
4.4.2.29	boolean isQualityKnown()
Description:	Request about each of the above mentioned characteristics of the BIR.
Return value:	True if the characteristic is available, false in other case.
Exception:	None

4.4.2.30	void setBDBBiometricSubtype(BiometricSubtype bdbBiometricSubtype)
Description:	Set the BDB biometric subtype
Parameters:	<i>bdbBiometricSubtype</i> : BDB biometric subtype

4.4.2.31	void setBDBBiometricType(BiometricType bdbBiometricType)
Description:	Set the BDB biometric type
Parameters:	<i>bdbBiometricType</i> : BDB biometric type

4.4.2.32	void setBDBChallengeResponse(byte bdbChallengeResponse)
Description:	Set the BDB challenge response
Parameters:	<i>bdbChallengeResponse</i> : BDB challenge response

4.4.2.33	void setBDBCreationDate(Date bdbCreationDate)
Description:	Set the BDB creation date
Parameters:	<i>bdbCreationDate</i> : BDB creation date

4.4.2.34	void setBDBEncryption(boolean bdbEncryption)
Description:	Determine if BDB data is encrypted or not
Parameters:	<i>bdbEncryption</i> : true if the BDB is encrypted, false otherwise.

4.4.2.35	void setBDBFormat(RegistryID bdbFormat)
Description:	Set the format of the BDB data
Parameters:	<i>bdbFormat</i> : format of the BDB data

4.4.2.36	void setBDBData(byte[] bdbData)
Description:	Set the BDB data
Parameters:	<i>bdbData</i> : BDB data

4.4.2.37	void setBDBIndex(byte[] bdbIndex)
Description:	Set the BDB index array
Parameters:	<i>bdbIndex</i> : BDB index array

4.4.2.38	void setBDBIntegrity(boolean bdbIntegrity)
Description:	Determine if BDB data has integrity or not
Parameters:	<i>bdbIntegrity</i> : true if the BDB has integrity, false otherwise.

4.4.2.39	void setBDBQuality(byte bdbQuality)
Description:	Set the BDB quality
Parameters:	<i>bdbQuality</i> : BDB quality

4.4.2.40	void setBDBProcessedLevel(ProcessedLevel bdbProcessedLevel)
Description:	Set the BDB processed level
Parameters:	<i>bdbProcessedLevel</i> : BDB processed level

4.4.2.41	void setBDBPurpose(Purpose bdbPurpose)
Description:	Set the BDB purpose
Parameters:	<i>bdbPurpose</i> : BDB purpose

4.4.2.42	void setBDBValidityPeriod(Vector<Date> bdbValidityPeriod)
Description:	Set the BDB validity period
Parameters:	<i>bdbValidityPeriod</i> : BDB validity period

4.4.2.43	void setBIRCreationDate(Date birCreationDate)
Description:	Set the BIR creation date
Parameters:	<i>birCreationDate</i> : BIR creation date

4.4.2.44	void setBIRCreator(byte[] birCreator)
Description:	Set the BIR creator array
Parameters:	<i>birCreator</i> : BIR creator array

4.4.2.45	void setBIRIndex(byte[] birIndex)
Description:	Set the BIR index
Parameters:	<i>birIndex</i> : BIR index

4.4.2.46	void setBIRAdditionalData(byte[] birAdditionalData)
Description:	Set the BIR additionalData
Parameters:	<i>birAdditionalData</i> : BIR additionalData

4.4.2.47	void setBIRValidityPeriod(Vector<Date> birValidityPeriod)
Description:	Set the BIR validity period
Parameters:	<i>birValidityPeriod</i> : BIR validity period

4.4.2.48	void setCBEFFVersion(byte cbeffVersion)
Description:	Set the version of the CBEFF component
Parameters:	<i>cbeffVersion</i> : version of the CBEFF component header version

4.4.2.49	void setPatronFormat(RegistryID patronFormat)
Description:	Set the patron format
Parameters:	<i>patronFormat</i> : patron format

4.4.2.50	void setPatronHeaderVersion(byte patronHeaderVersion)
Description:	Set the header version
Parameters:	<i>patronHeaderVersion</i> : header version

4.4.2.51	void setSBData(byte[] sbData)
Description:	Set the SB data
Parameters:	<i>sbData</i> : SB data

4.4.2.52	void setSBFormat(RegistryID sbFormat)
Description:	Set the format of the SB data
Parameters:	<i>sbFormat</i> : format of the SB data

4.5 Class BSPSchema

4.5.1 Description

Represents the record in the component registry that defines the properties of the BSP installed in the system.

4.5.2 Method summary

4.5.2.1	UUID getBSPAccessUUID()
Description:	Return a UUID, unique within the scope of an application, which the application may use to refer to the BSP as an alternative to the BSP product UUID. This parameter shall be ignored by frameworks conforming to this part of ISO/IEC 30106 and can be set to any UUID value by an application. It is provided to support interworking standards, which may specify the use of identical BSPs present on multiple computers from within an application running on the same or a different computer.
Return value:	A UUID, unique within the scope of an application, which the application may use to refer to the BSP as an alternative to the BSP product UUID.

4.5.2.2	String getBSPDescription()
Description:	Return a string containing a text description of the BSP.
Return value:	A string containing a text description of the BSP.

4.5.2.3	Vector<RegistryID> getBSPSupportedAlgorithms()
Description:	Return an array of BioAPI_ALGORITHM_ID structures specifying the supported algorithms.
Return value:	Array of BioAPI_ALGORITHM_ID structures specifying the supported algorithms.

4.5.2.4	Vector<RegistryID> getBSPSupportedFormats()
Description:	Return a list the data formats that are supported by the BSP.
Return value:	A list the data formats that are supported by the BSP.

4.5.2.5	Vector<UUID> getBSPSupportedTransformOperation()
Description:	Return an array of BioAPI_UUID structures specifying the transform operations supported within the BioAPI_Transform operation.
Return value:	Array of BioAPI_UUID structures specifying the transform operations supported within the BioAPI_Transform operation.

4.5.2.6	UUID getBSPUUID()
Description:	Return the BSP UUID
Return value:	The BSP UUID

4.5.2.7	int getDefaultCalibrateTimeout()
Description:	Return the default timeout value in milliseconds used by the BSP for Calibrate operations when no timeout is specified by the application.
Return value:	The default timeout value for Calibrate operations.

4.5.2.8	int getDefaultCaptureTimeout()
Description:	Return the default timeout value in milliseconds used by the BSP for Capture operations when no timeout is specified by the application.
Return value:	The default timeout value for Capture operations.

4.5.2.9	int getDefaultEnrolTimeout()
Description:	Return the default timeout value in milliseconds used by the BSP for Enrol operations when no timeout is specified by the application.
Return value:	The default timeout value for Enrol operations.

4.5.2.10	int getDefaultIdentifyTimeout()
Description:	Return the default timeout value in milliseconds used by the BSP for Identify operations when no timeout is specified by the application.
Return value:	The default timeout value for Identify operations.

4.5.2.11	int getDefaultVerifyTimeout()
Description:	Return the default timeout value in milliseconds used by the BSP for Verify operations when no timeout is specified by the application.
Return value:	The default timeout value for Verify operations.

4.5.2.12	Vector<BiometricType> getFactorsMask()
Description:	Return a list of the biometric types supported by the BSP.
Return value:	A list of the biometric types supported by the BSP.

4.5.2.13	byte[] getHostingEndpointIRI()
Description:	Return an IRI identifying the framework whose component registry contains a registration of the BSP. This parameter shall be ignored by frameworks conforming to this part of ISO/IEC 30106 and shall be set to NULL by an application. It is provided to support interworking standards, which may specify the use of identical BSPs present on multiple computers from within an application running on the same or a different computer.
Return value:	An IRI identifying the framework whose component registry contains a registration of the BSP.

4.5.2.14	int getMaxBSPDbSize()
Description:	Return the maximum size of a BSP-controlled BIR database. It applies only when a BSP is only capable of directly managing a single archive unit. A value of zero means that no information about the database size is being provided for one of the following three reasons: a) databases are not supported; b) it is capable of managing multiple units (either directly or through a BFP interface), each of which may have a different maximum size, and information about these units will be provided as part of the insert notification (part of Unit Schema); or c) one archive unit is supported, but the information is not given here ; it will be provided in the insert notification.
Return value:	The maximum size of a BSP-controlled BIR database.

4.5.2.15	int getMaxIdentify()
Description:	Return the largest population supported by the identify function. Unlimited = 0xFFFFFFFF.
Return value:	The largest population supported by the identify function.

4.5.2.16	int getMaxNumEnrollInstances()
Description:	Return the maximum number of distinct instances that a BSP can create reference templates for, in one enroll operation. This information can be useful to an application that uses the application-controlled GUI feature.
Return value:	The maximum number of distinct instances that a BSP can create reference templates for in one enroll operation.

4.5.2.17	int getMaxAdditionalDataSize()
Description:	Return the maximum additionalData size (in bytes) that the BSP can accept
Return value:	The maximum additionalData size (in bytes) that the BSP can accept

4.5.2.18	Vector<BSPSchemaOperations> getOperations()
Description:	Return a list of the biometric operations supported by the BSP.
Return value:	A list of the biometric operations supported by the BSP.

4.5.2.19	Vector<BSPSchemaOptions> getOptions()
Description:	Return a list of the biometric options supported by the BSP.
Return value:	A list of the biometric options supported by the BSP.

4.5.2.20	String getPath()
Description:	Return a pointer to a 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 10646:2014, 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.
Return value:	A pointer to a string containing the path of the file containing the BSP executable code, including the filename.

4.5.2.21	int getAdditionalDataPolicy()
Description:	Return the threshold setting (maximum FMR value) used to determine when to release additionalData after successful verification.
Return value:	The threshold setting (maximum FMR value) used to determine when to release additionalData after successful verification.

4.5.2.22	String getProductVersion()
Description:	Return the version string of the BSP software.
Return value:	The version string of the BSP software.

4.5.2.23	String getSpecVersion()
Description:	Return the major/minor version number of the BioAPI specification to which the BSP was implemented.
Return value:	Major/minor version number of the BioAPI specification to which the BSP was implemented.

4.5.2.24	String getVendor()
Description:	Return a string containing the name of the BSP vendor.
Return value:	A string containing the name of the BSP vendor.

NOTE The “BSPAccess_UUID” and the “HostingendpointIRI” are part of the definition of the C type BioAPI_BSP_SCHEMA, but are not part of the BSP schema information stored in the component registry (see ISO/IEC 19784-1).

4.6 Class candidate

4.6.1 Description

Defines each of the resulting candidates from the Identify functionality.

4.6.2 Method summary

4.6.2.1	int getFMRAchieved()
Description:	Return a integer value which states the FMR achieved by the candidate during the Identify process.
Return value:	FMR value

4.6.2.2	UUID getKey()
Description:	Return a key value which defines the UUID of the candidate in the system (e.g. in the database).
Return value:	Key value

4.6.2.3	void setFMRAchieved(int fmrAchieved)
Description:	Set the FMR achieved by the candidate during the Identify process.
Parameters:	<i>fmrAchieved</i> : FMR value

4.6.2.4	void setKey(UUID key)
Description:	Set the key value which defines the UUID of the candidate in the system (e.g. in the database).
Parameters:	<i>key</i> : key value

4.7 Class DataTypes

4.7.1 Description

This class defines the different data types that can be used throughout this part of ISO/IEC 30106. It embraces enumerations and constants. Values for the constants not defined in this part of ISO/IEC 30106 are defined in ISO/IEC 30106-1.

4.7.2 Enumerations

4.7.2.1	BiometricSubtype
Description:	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.
Enum constant summary:	— <i>NO_VALUE_AVAILABLE</i> — <i>LEFT</i> — <i>RIGHT</i> — <i>LEFT_THUMB</i> — <i>LEFT_INDEX_FINGER</i> — <i>LEFT_MIDDLE_FINGER</i> — <i>LEFT_RING_FINGER</i> — <i>LEFT_LITTLE_FINGER</i> — <i>RIGHT_THUMB</i> — <i>RIGHT_INDEX_FINGER</i> — <i>RIGHT_MIDDLE_FINGER</i> — <i>RIGHT_RING_FINGER</i> — <i>RIGHT_LITTLE_FINGER</i> — <i>LEFT_PALM</i> — <i>LEFT_BACK_OF_HAND</i> — <i>LEFT_WRIST</i> — <i>RIGHT_PALM</i> — <i>RIGHT_BACK_OF_HAND</i> — <i>RIGHT_WRIST</i>