



BSI Standards Publication

Information technology — Object oriented BioAPI

Part 2: Java implementation

National foreword

This British Standard is the UK implementation of [ISO/IEC 30106-2:2020](#). It supersedes [BS ISO/IEC 30106-2:2016](#), which is withdrawn.

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 committee manager.

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 2020
Published by BSI Standards Limited 2020

ISBN 978 0 539 04204 7

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 November 2020.

Amendments/corrigenda issued since publication

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

Information technology — Object oriented BioAPI —

Part 2: Java implementation

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





COPYRIGHT PROTECTED DOCUMENT

© ISO/IEC 2020, 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 Terms and definitions	1
4 BioAPI Java package structure	1
4.1 Overall structure	1
4.2 Package org.bioapi	1
4.2.1 Package description	1
4.2.2 Structure	2
4.3 Package org.bioapi.data	2
4.3.1 Package description	2
4.3.2 Structure	2
5 Data types and constants	2
5.1 Class ACBioParameters	2
5.1.1 Description	2
5.1.2 Method summary	2
5.2 Class BFPListElement	3
5.2.1 Description	3
5.2.2 Method summary	3
5.3 Class BFPSchema	3
5.3.1 Description	3
5.3.2 Method summary	3
5.4 Class BIR	5
5.4.1 Description	5
5.4.2 Method summary	5
5.5 Class BSPSchema	13
5.5.1 Description	13
5.5.2 Method Summary	13
5.6 Class Candidate	16
5.6.1 Description	16
5.6.2 Method summary	16
5.7 Class DataTypes	17
5.7.1 Description	17
5.7.2 Enumerations	17
5.8 Class Date	26
5.8.1 Description	26
5.8.2 Method summary	26
5.9 Class FrameworkSchema	30
5.9.1 Description	30
5.9.2 Method summary	30
5.10 Class GUIBitmap	32
5.10.1 Description	32
5.10.2 Method summary	32
5.11 Class IdentifyPopulation	33
5.11.1 Description	33
5.11.2 Method summary	33
5.12 Class PopulationMember	34
5.12.1 Description	34
5.12.2 Method summary	34
5.13 Class RegistryID	34
5.13.1 Description	34

5.13.2	Method summary	34
5.14	Class SecurityProfileType	35
5.14.1	Description	35
5.14.2	Method summary	35
5.15	Class UnitList	36
5.15.1	Description	36
5.15.2	Method summary	36
5.16	Class UnitListElement	37
5.16.1	Description	37
5.16.2	Method summary	37
5.17	Class UnitSchema	37
5.17.1	Description	37
5.17.2	Method summary	37
5.18	Class UUID	41
5.18.1	Description	41
6	Object oriented interfaces for supporting BioAPI_Units	41
6.1	General	41
6.2	Interface archive	41
6.2.1	Description	41
6.2.2	Method summary	41
6.3	Interface comparison	46
6.3.1	Description	46
6.3.2	Method summary	46
6.4	Interface processing	49
6.4.1	Description	49
6.4.2	Method summary	49
6.5	Interface sensor	51
6.5.1	Description	51
6.5.2	Method summary	51
7	BFP level	52
7.1	Interface BFP	52
7.1.1	Description	52
7.1.2	Imported interfaces	53
7.1.3	Method summary	53
8	BSP level	55
8.1	Interface BSP	55
8.1.1	Description	55
8.1.2	Imported interfaces	56
8.1.3	Method summary	56
9	Framework level	65
9.1	Interface ComponentRegistry	65
9.1.1	Description	65
9.1.2	Method summary	65
9.2	Interface framework	66
9.2.1	Description	66
9.2.2	Inherited interfaces	67
9.2.3	Method summary	68
10	Application interaction	72
10.1	class BioAPIException extends Exception	72
10.1.1	Description	72
10.1.2	Constructor summary	73
10.1.3	Method Summary	74
10.2	GUI callback functions	74
10.2.1	Description	74
10.2.2	Callback interface specification	75

11	BSP Interaction	80
11.1	Interface BSPEventListener	80
11.1.1	Method summary	80
12	BFP Interaction	80
12.1	Interface BFPEnumerationListener	80
12.1.1	Method summary	80
12.2	Interface BFPEventListener	80
12.2.1	Method summary	80
12.3	Interface BFPGUIProgressEventListener	81
12.3.1	Method summary	81
Annex A	(informative) Java requirements	82
Annex B	(informative) Calling sequence examples and sample code	83
Bibliography		84

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*.

This second edition cancels and replaces the first edition ([ISO/IEC 30106-2:2016](http://www.iso.org/iso/30106-2:2016)), which has been technically revised.

The main changes compared to the previous edition are as follows:

- correction of typing errors;
- addition of AnalyseQuality method.

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

This document specifies an application programming interface expressed in Java language. 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. It is therefore the best way to write desktop and web applications/services. This document provides an advanced and well-designed remote framework.

Although the best practices of Java programming state that variables should be written in lowercase letters, in the case of symbols, such as BSP or BFPs, they have been retained in uppercase letters.

This page deliberately left blank

Information technology — Object oriented BioAPI —

Part 2: Java implementation

1 Scope

This document 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 [ISO/IEC 30106-1](#) is maintained in this document.

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 10646](#), *Information technology — Universal Coded Character Set (UCS)*

[ISO/IEC 30106-1](#), *Information technology — Object oriented BioAPI — Part 1: Architecture*

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 Java package structure

4.1 Overall structure

The BioAPI Java interface is 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.

4.2 Package org.bioapi

4.2.1 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.

4.2.2 Structure

The description of this package is given explaining a bottom-up structure. In [Clause 6](#), the interfaces that need 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 be either the Biometric Service Provider (BSP) or the Biometric Function Provider (BFP), but the specifications in this 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 7](#)) and BSP ([Clause 8](#)) 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 9](#), with the Framework Interface and the Component Registry) and the Application interaction ([Clause 10](#), with the specification of the Exceptions and Callback functions). This provides the equivalence to the API interface in [ISO/IEC 19784-1](#).

4.3 Package org.bioapi.data

4.3.1 Package description

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

4.3.2 Structure

Several data structures are provided to comply with the requirements specified by this document. The whole org.bioapi.data package is specified in [Clause 5](#), where all required classes and enumerations are defined. This has to be complemented to the constants defined in [ISO/IEC 30106-1](#).

5 Data types and constants

5.1 Class ACBioParameters

5.1.1 Description

Provides the information which is used to generate ACBio instances.

5.1.2 Method summary

5.1.2.1 `int[] getChallenge()`

<u>Description:</u>	Return the challenge from the validator of a biometric verification when ACBio is used. This value shall be sent to the field <code>controlValue</code> of type <code>ACBioContentInformation</code> in ACBio instances.
<u>Return Value:</u>	The challenge from the validator of a biometric verification when ACBio is used.

5.1.2.2 `int[] getInitialBPUIOIndexOutput()`

<u>Description:</u>	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 <code>InitialBPUIOIndexOutput</code> and <code>SupremumBPUIOIndexOutput</code> shall be divided into the number of BSP Units and BFPs which are inside the BSP, and assigned to the BSP Units and BSPs.
<u>Return Value:</u>	The initial value of BPU IO index.

5.1.2.3 int[] getSupremumBPUIOIndexOutput()

<u>Description:</u>	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.
<u>Return Value:</u>	The supremum of BPU IO index.

5.2 Class BFPListElement**5.2.1 Description**

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

5.2.2 Method summary**5.2.2.1 UUID getBFPID()**

<u>Description:</u>	Return the UUID assigned to the BFP.
<u>Return Value:</u>	The UUID assigned to the BFP.

5.2.2.2 UnitCategoryType getUnitCategory()

<u>Description:</u>	Return the category of the units.
<u>Return Value:</u>	The category of the units.

5.2.2.3 void setBFPID(UUID bfpID)

<u>Description:</u>	Set the UUID assigned to the BFP.
<u>Parameters:</u>	— <i>bfpID</i> : The UUID assigned to the BFP.

5.2.2.4 void setUnitCategory(UnitCategoryType unitCategory)

<u>Description:</u>	Set the category of the units.
<u>Parameters:</u>	— <i>unitCategory</i> : The category of the units.

5.3 Class BFPSchema**5.3.1 Description**

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

5.3.2 Method summary**5.3.2.1 String getBFPDescription()**

<u>Description:</u>	Return a NULL-terminated string containing a text description of the BFP.
<u>Return Value:</u>	A NULL-terminated string containing a text description of the BFP.

5.3.2.2 Vector<RegistryID> getBFPSupportedFormats()

<u>Description:</u>	Return a list the data formats that are supported by the BFP.
<u>Return Value:</u>	A list the data formats that are supported by the BFP.

5.3.2.3 UUID getBFPUUID()

<u>Description:</u>	Return the BFP UUID.
<u>Return Value:</u>	The BFP UUID.

5.3.2.4 Vector<BiometricType> getFactorsMask()

<u>Description:</u>	Return a list of the biometric types supported by the BFP.
<u>Return Value:</u>	A list of the biometric types supported by the BFP.

5.3.2.5 byte[] getFWProperty()

<u>Description:</u>	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.
<u>Return Value:</u>	The address and length of a memory buffer containing the BFP property.

5.3.2.6 UUID getFWPropertyID()

<u>Description:</u>	Return the UUID of the format of the following BFP property.
<u>Return Value:</u>	UUID of the format of the following BFP property.

5.3.2.7 String getPath()

<u>Description:</u>	Return 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.
<u>Return Value:</u>	A pointer to a NULL-terminated string containing the path of the file containing the BFP executable code, including the filename.

5.3.2.8 String getProductVersion()

<u>Description:</u>	Return the version string of the BFP software.
<u>Return Value:</u>	The version string of the BFP software.

5.3.2.9 String getSpecVersion()

<u>Description:</u>	Return the major/minor version number of the BioAPI specification to which the BFP was implemented.
<u>Return Value:</u>	The major/minor version number of the BioAPI specification to which the BFP was implemented.

5.3.2.10 UnitCategoryType getUnitCategory()

<u>Description:</u>	Return the category of the BFP identified by the BFP UUID.
<u>Return Value:</u>	The category of the BFP identified by the BFP UUID.

5.3.2.11 String getVendor()

<u>Description:</u>	Return a NULL-terminated string containing the name of the BFP vendor.
<u>Return Value:</u>	A NULL-terminated string containing the name of the BFP vendor.

5.4 Class BIR

5.4.1 Description

Represents BIRs (Biometric Information Records). It supports [ISO/IEC 19785-1](#) 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](#).

5.4.2 Method summary

5.4.2.1 void birFromArray(byte[] record)

<u>Description:</u>	Fills in the BIR data from a byte array coded as CBEFF record, as indicated in the relevant clauses of ISO/IEC 19785-3 and ISO/IEC 19785-4 .
<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).

5.4.2.2 byte[] birToArray()

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

5.4.2.3 void destroy()

<u>Description:</u>	Removes all the information in the current BIR, leaving it empty for a next use.
<u>Exception:</u>	None.

5.4.2.4 BiometricSubtype getBDBBiometricSubtype()

<u>Description:</u>	Return the BDB biometric subtype.
<u>Return Value:</u>	The BDB biometric subtype.

5.4.2.5 BiometricType getBDBBiometricType()

<u>Description:</u>	Return the BDB biometric type.
<u>Return Value:</u>	The BDB biometric type.

5.4.2.6 byte[] getBDBChallengeResponse()

<u>Description:</u>	Return the BDB challenge response.
<u>Return Value:</u>	The BDB challenge response.

5.4.2.7 Date getBDBCreationDate()

<u>Description:</u>	Return the BDB creation date.
<u>Return Value:</u>	The BDB creation date.

5.4.2.8 byte[] getBDBData()

<u>Description:</u>	Return the BDB data array.
<u>Return Value:</u>	The BDB data array.

5.4.2.9 RegistryID getBDBFormat()

<u>Description:</u>	Return the format of the BDB data.
<u>Return Value:</u>	The format of the BDB data.

5.4.2.10 byte[] getBDBIndex()

<u>Description:</u>	Return the BDB index.
<u>Return Value:</u>	The BDB index.

5.4.2.11 ProcessedLevel getBDBProcessedLevel()

<u>Description:</u>	Return the BDB processed level.
<u>Return Value:</u>	The BDB processed level.

5.4.2.12 Purpose getBDBPurpose()

<u>Description:</u>	Return the BDB purpose.
<u>Return Value:</u>	The BDB purpose.

5.4.2.13 byte getBDBQuality()

<u>Description:</u>	Return the BDB quality.
<u>Return Value:</u>	The BDB quality.

5.4.2.14 Vector<Date> getBDBValidityPeriod()

<u>Description:</u>	Return the BDB validity period.
<u>Return Value:</u>	The BDB validity period.

5.4.2.15 Date getBIRCreationDate()

<u>Description:</u>	Return the BIR creation date.
<u>Return Value:</u>	The BIR creation date.

5.4.2.16 byte[] getBIRCreator()

<u>Description:</u>	Return the BIR creator array.
<u>Return Value:</u>	The BIR creator array.

5.4.2.17 byte[] getBIRIndex()

<u>Description:</u>	Return the BIR index.
<u>Return Value:</u>	The BIR index.

5.4.2.18 byte[] getBIRAdditionalData()

<u>Description:</u>	Return the BIR additionalData.
<u>Return Value:</u>	The BIR additionalData.

5.4.2.19 Vector<Date> getBIRValidityPeriod()

<u>Description:</u>	Return the BIR validity period.
<u>Return Value:</u>	The BIR validity period.

5.4.2.20 byte getCBEFFVersion()

<u>Description:</u>	Return the version of the CBEFF component.
<u>Return Value:</u>	The version of the CBEFF component.

5.4.2.21 RegistryID getPatronFormat()

<u>Description:</u>	Return the patron format.
<u>Return Value:</u>	The patron format.

5.4.2.22 byte getPatronHeaderVersion()

<u>Description:</u>	Return the header version.
<u>Return Value:</u>	The header version.

5.4.2.23 byte[] getSBData()

<u>Description:</u>	Return the Security Block data array.
<u>Return Value:</u>	The Security Block data array.

5.4.2.24 RegistryID getSBFormat()

<u>Description:</u>	Return the Security Block format.
<u>Return Value:</u>	The Security Block format.

5.4.2.25 boolean hasBDBEncryption()

<u>Description:</u>	Return true if the BDB is encrypted, false otherwise.
<u>Return Value:</u>	True if the BDB is encrypted, false otherwise.

5.4.2.26 boolean hasBDBIntegrity()

<u>Description:</u>	Return true if the BDB has integrity, false otherwise.
<u>Return Value:</u>	True if the BDB has integrity, false otherwise.

5.4.2.27 boolean isBIRSigned()

See [5.4.2.29](#).

5.4.2.28 boolean isQualitySupported()

See [5.4.2.29](#).

5.4.2.29 boolean isQualityKnown()

<u>Description:</u>	Request about each of the abovementioned characteristics of the BIR.
<u>Return Value:</u>	True if the characteristic is available, false otherwise.
<u>Exception:</u>	None.

5.4.2.30 void setBDBBiometricSubtype(BiometricSubtype bdbBiometricSubtype)

<u>Description:</u>	Set the BDB biometric subtype.
<u>Parameters:</u>	— <i>bdbBiometricSubtype</i> : The BDB biometric subtype.

5.4.2.31 void setBDBBiometricType(BiometricType bdbBiometricType)

<u>Description:</u>	Set the BDB biometric type.
<u>Parameters:</u>	— <i>bdbBiometricType</i> : The BDB biometric type.

5.4.2.32 void setBDBChallengeResponse(byte bdbChallengeResponse)

<u>Description:</u>	Set the BDB challenge response.
<u>Parameters:</u>	— <i>bdbChallengeResponse</i> : The BDB challenge response.

5.4.2.33 void setBDBCreationDate(Date bdbCreationDate)

<u>Description:</u>	Set the BDB creation date.
<u>Parameters:</u>	— <i>bdbCreationDate</i> : The BDB creation date.

5.4.2.34 void setBDBEncryption(boolean bdbEncryption)

<u>Description:</u>	Determine if BDB data is encrypted or not.
<u>Parameters:</u>	— <i>bdbEncryption</i> : True if the BDB is encrypted, false otherwise.

5.4.2.35 void setBDBFormat(RegistryID bdbFormat)

<u>Description:</u>	Set the format of the BDB data.
<u>Parameters:</u>	— <i>bdbFormat</i> : The format of the BDB data.

5.4.2.36 void setBDBData(byte[] bdbData)

<u>Description:</u>	Set the BDB data.
<u>Parameters:</u>	— <i>bdbData</i> : The BDB data.

5.4.2.37 void setBDBIndex(byte[] bdbIndex)

<u>Description:</u>	Set the BDB index array.
<u>Parameters:</u>	— <i>bdbIndex</i> : The BDB index array.

5.4.2.38 void setBDBIntegrity(boolean bdbIntegrity)

<u>Description:</u>	Determine if BDB data has integrity or not.
<u>Parameters:</u>	— <i>bdbIntegrity</i> : True if the BDB has integrity, false otherwise.

5.4.2.39 void setBDBQuality(byte bdbQuality)

<u>Description:</u>	Set the BDB quality.
<u>Parameters:</u>	— <i>bdbQuality</i> : The BDB quality.

5.4.2.40 void setBDBProcessedLevel(ProcessedLevel bdbProcessedLevel)

<u>Description:</u>	Set the BDB processed level.
---------------------	------------------------------

<u>Parameters:</u>	— <i>bdbProcessedLevel</i> : The BDB processed level.
--------------------	---

5.4.2.41 void setBDBPurpose(Purpose bdbPurpose)

<u>Description:</u>	Set the BDB purpose.
<u>Parameters:</u>	— <i>bdbPurpose</i> : The BDB purpose.

5.4.2.42 void setBDBValidityPeriod(Vector<Date> bdbValidityPeriod)

<u>Description:</u>	Set the BDB validity period.
<u>Parameters:</u>	— <i>bdbValidityPeriod</i> : The BDB validity period.

5.4.2.43 void setBIRCreationDate(Date birCreationDate)

<u>Description:</u>	Set the BIR creation date.
<u>Parameters:</u>	— <i>birCreationDate</i> : The BIR creation date.

5.4.2.44 void setBIRCreator(byte[] birCreator)

<u>Description:</u>	Set the BIR creator array.
<u>Parameters:</u>	— <i>birCreator</i> : The BIR creator array.

5.4.2.45 void setBIRIndex(byte[] birIndex)

<u>Description:</u>	Set the BIR index.
<u>Parameters:</u>	— <i>birIndex</i> : The BIR index.

5.4.2.46 void setBIRAdditionalData(byte[] birAdditionalData)

<u>Description:</u>	Set the BIR additionalData.
<u>Parameters:</u>	— <i>birAdditionalData</i> : The BIR additionalData.

5.4.2.47 void setBIRValidityPeriod(Vector<Date> birValidityPeriod)

<u>Description:</u>	Set the BIR validity period.
<u>Parameters:</u>	— <i>birValidityPeriod</i> : The BIR validity period.

5.4.2.48 void setCBEFFVersion(byte cbeffVersion)

<u>Description:</u>	Set the version of the CBEFF component.
<u>Parameters:</u>	— <i>cbeffVersion</i> : The version of the CBEFF component.

5.4.2.49 void setPatronFormat(RegistryID patronFormat)

<u>Description:</u>	Set the patron format.
<u>Parameters:</u>	— <i>patronFormat</i> : The patron format.

5.4.2.50 void setPatronHeaderVersion(byte patronHeaderVersion)

<u>Description:</u>	Set the header version.
<u>Parameters:</u>	— <i>patronHeaderVersion</i> : The header version.

5.4.2.51 void setSBData(byte[] sbData)

<u>Description:</u>	Set the SB data.
---------------------	------------------

<u>Parameters:</u>	— <i>sbData</i> : The SB data.
--------------------	--------------------------------

5.4.2.52 void setSBFormat(RegistryID sbFormat)

<u>Description:</u>	Set the format of the SB data.
<u>Parameters:</u>	— <i>sbFormat</i> : The format of the SB data.

5.5 Class BSPSchema

5.5.1 Description

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

5.5.2 Method Summary

5.5.2.1 UUID getBSPAccessUUID()

<u>Description:</u>	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 document 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.
<u>Return Value:</u>	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.

5.5.2.2 String getBSPDescription()

<u>Description:</u>	Return a NULL-terminated string containing a text description of the BSP.
<u>Return Value:</u>	A NULL-terminated string containing a text description of the BSP.

5.5.2.3 Vector<RegistryID> getBSPSupportedAlgorithms()

<u>Description:</u>	Return an array of BioAPI_ALGORITHM_ID structures specifying the supported algorithms.
<u>Return Value:</u>	Array of BioAPI_ALGORITHM_ID structures specifying the supported algorithms.

5.5.2.4 Vector<RegistryID> getBSPSupportedFormats()

<u>Description:</u>	Return a list of the data formats that are supported by the BSP.
<u>Return Value:</u>	A list of the data formats that are supported by the BSP.

5.5.2.5 Vector<UUID> getBSPSupportedTransformOperation()

<u>Description:</u>	Return an array of BioAPI_UUID structures specifying the transform operations supported within the BioAPI_Transform operation.
<u>Return Value:</u>	Array of BioAPI_UUID structures specifying the transform operations supported within the BioAPI_Transform operation.

5.5.2.6 UUID getBSPUUID()

<u>Description:</u>	Return the BSP UUID.
<u>Return Value:</u>	The BSP UUID.

5.5.2.7 int getDefaultCalibrateTimeout()

<u>Description:</u>	Return the default timeout value in milliseconds used by the BSP for Calibrate operations when no timeout is specified by the application.
<u>Return Value:</u>	The default timeout value for Calibrate operations.

5.5.2.8 int getDefaultCaptureTimeout()

<u>Description:</u>	Return the default timeout value in milliseconds used by the BSP for Capture operations when no timeout is specified by the application.
<u>Return Value:</u>	The default timeout value for Capture operations.

5.5.2.9 int getDefaultEnrolTimeout()

<u>Description:</u>	Return the default timeout value in milliseconds used by the BSP for Enrol operations when no timeout is specified by the application.
<u>Return Value:</u>	The default timeout value for Enrol operations.

5.5.2.10 int getDefaultIdentifyTimeout()

<u>Description:</u>	Return the default timeout value in milliseconds used by the BSP for Identify operations when no timeout is specified by the application.
<u>Return Value:</u>	The default timeout value for Identify operations.

5.5.2.11 int getDefaultVerifyTimeout()

<u>Description:</u>	Return the default timeout value in milliseconds used by the BSP for Verify operations when no timeout is specified by the application.
<u>Return Value:</u>	The default timeout value for Verify operations.

5.5.2.12 Vector<BiometricType> getFactorsMask()

<u>Description:</u>	Return a list of the biometric types supported by the BSP.
<u>Return Value:</u>	A list of the biometric types supported by the BSP.

5.5.2.13 byte[] getHostingEndpointIRI()

<u>Description:</u>	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 document 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.
<u>Return Value:</u>	An IRI identifying the framework whose component registry contains a registration of the BSP.

5.5.2.14 int getMaxBSPDbSize()

<u>Description:</u>	Return the maximum size of a BSP-controlled BIR database. This 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).
<u>Return Value:</u>	The maximum size of a BSP-controlled BIR database.

5.5.2.15 int getMaxIdentify()

<u>Description:</u>	Return the largest population supported by the identify function. Unlimited = 0xFFFFFFFF.
<u>Return Value:</u>	The largest population supported by the identify function.

5.5.2.16 int getMaxNumEnrollInstances()

<u>Description:</u>	Return the maximum number of distinct instances for which a BSP can create reference templates in one enrol operation. This information can be useful to an application that uses the application-controlled GUI feature.
<u>Return Value:</u>	The maximum number of distinct instances for which a BSP can create reference templates in one enrol operation.

5.5.2.17 int getMaxAdditionalDataSize()

<u>Description:</u>	Return the maximum additionalData size (in bytes) that the BSP can accept.
<u>Return Value:</u>	The maximum additionalData size (in bytes) that the BSP can accept.

5.5.2.18 Vector<BSPSchemaOperations> getOperations()

<u>Description:</u>	Return a list of the biometric operations supported by the BSP.
<u>Return Value:</u>	A list of the biometric operations supported by the BSP.

5.5.2.19 Vector<BSPSchemaOptions> getOptions()

<u>Description:</u>	Return a list of the biometric options supported by the BSP.
<u>Return Value:</u>	A list of the biometric options supported by the BSP.

5.5.2.20 String getPath()

<u>Description:</u>	Return 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.
<u>Return Value:</u>	A pointer to a NULL-terminated string containing the path of the file containing the BSP executable code, including the filename.

5.5.2.21 int getAdditionalDataPolicy()

<u>Description:</u>	Return the threshold setting (maximum FMR value) used to determine when to release additionalData after successful verification.
<u>Return Value:</u>	The threshold setting (maximum FMR value) used to determine when to release additionalData after successful verification.

5.5.2.22 String getProductVersion()

<u>Description:</u>	Return the version string of the BSP software.
<u>Return Value:</u>	The version string of the BSP software.

5.5.2.23 String getSpecVersion()

<u>Description:</u>	Return the major/minor version number of the BioAPI specification to which the BSP was implemented.
<u>Return Value:</u>	Major/minor version number of the BioAPI specification to which the BSP was implemented.

5.5.2.24 String getVendor()

<u>Description:</u>	Return a NULL-terminated string containing the name of the BSP vendor.
<u>Return Value:</u>	A NULL-terminated 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](#)).

5.6 Class Candidate

5.6.1 Description

Defines each of the resulting candidates from the Identify functionality.

5.6.2 Method summary

5.6.2.1 int getFMRAchieved()

<u>Description:</u>	Return an integer value which states the FMR achieved by the candidate during the Identify process.
<u>Return Value:</u>	FMR value.

5.6.2.2 UUID getKey()

<u>Description:</u>	Return a key value which defines the UUID of the candidate in the system (e.g. in the database).
<u>Return Value:</u>	Key value.

5.6.2.3 void setFMRAchieved(int fmrAchieved)

<u>Description:</u>	Set the FMR achieved by the candidate during the Identify process.
---------------------	--

<u>Parameters:</u>	— <i>fmrAchieved</i> : FMR value.
--------------------	-----------------------------------

5.6.2.4 void setKey(UUID key)

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

5.7 Class DataTypes

5.7.1 Description

Defines the different data types that can be used throughout this document. It embraces enumerations and constants. Values for the constants not defined in this document are defined in [ISO/IEC 30106-1](#).

5.7.2 Enumerations

5.7.2.1 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>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>
--------------------------------------	--

5.7.2.2 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>NO_VALUE_AVAILABLE</i> — <i>MULTIPLE_BIOMETRIC_TYPES</i> — <i>FACE</i> — <i>VOICE</i> — <i>FINGER</i> — <i>IRIS</i> — <i>RETINA</i> — <i>HAND_GEOMETRY</i> — <i>SIGNATURE_OR_SIGN</i> — <i>KEYSTROKE</i> — <i>LIP_MOVEMENT</i> — <i>GAIT</i> — <i>VEIN</i> — <i>DNA</i> — <i>EAR</i> — <i>FOOT</i> — <i>SCENT</i>
-------------------------------	--

5.7.2.3 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>READ_WRITE</i>: Access mode which allows addition, deletion and retrieval of records. — <i>WRITE</i>: Access mode which allows addition and deletion of records, but for which retrieval is not allowed.