

DRAFT INTERNATIONAL STANDARD

ISO/IEC DIS 30106-3

ISO/IEC JTC 1/SC 37

Secretariat: ANSI

Voting begins on:
2014-12-04

Voting terminates on:
2015-03-04

Information Technology — Object oriented BioAPI —

Part 3: C# implementation

Titre manque

ICS: 35.040

iTeh STANDARD PREVIEW
(standards.iteh.ai)

Full standard:
<https://standards.iteh.ai/catalog/standards/sist/9d90fa5f-f712-4c80-866b-3884db3fb3ce/iso-iec-30106-3-2016>

THIS DOCUMENT IS A DRAFT CIRCULATED FOR COMMENT AND APPROVAL. IT IS THEREFORE SUBJECT TO CHANGE AND MAY NOT BE REFERRED TO AS AN INTERNATIONAL STANDARD UNTIL PUBLISHED AS SUCH.

IN ADDITION TO THEIR EVALUATION AS BEING ACCEPTABLE FOR INDUSTRIAL, TECHNOLOGICAL, COMMERCIAL AND USER PURPOSES, DRAFT INTERNATIONAL STANDARDS MAY ON OCCASION HAVE TO BE CONSIDERED IN THE LIGHT OF THEIR POTENTIAL TO BECOME STANDARDS TO WHICH REFERENCE MAY BE MADE IN NATIONAL REGULATIONS.

RECIPIENTS OF THIS DRAFT ARE INVITED TO SUBMIT, WITH THEIR COMMENTS, NOTIFICATION OF ANY RELEVANT PATENT RIGHTS OF WHICH THEY ARE AWARE AND TO PROVIDE SUPPORTING DOCUMENTATION.



Reference number
ISO/IEC DIS 30106-3:2014(E)

© ISO/IEC 2014

iTeh STANDARD PREVIEW
(standards.iteh.ai)

Full standard:
<https://standards.iteh.ai/catalog/standards/sist/9d90fa5f-f712-4c80-866b-3884db3fb3ce/iso-iec-30106-3-2016>

Copyright notice

This ISO document is a Draft International Standard and is copyright-protected by ISO. Except as permitted under the applicable laws of the user's country, neither this ISO draft nor any extract from it may be reproduced, stored in a retrieval system or transmitted in any form or by any means, electronic, photocopying, recording or otherwise, without prior written permission being secured.

Requests for permission to reproduce should be addressed to either ISO at the address below or ISO's member body in the country of the requester.

ISO copyright office
Case postale 56 • CH-1211 Geneva 20
Tel. + 41 22 749 01 11
Fax + 41 22 749 09 47
E-mail copyright@iso.org
Web www.iso.org

Reproduction may be subject to royalty payments or a licensing agreement.

Violators may be prosecuted.

Contents

Page

Foreword	ix
Introduction.....	x
1 Scope	1
2 BioAPI C# Namespace Structure	1
2.1 Namespace BioAPI	1
2.1.1 Namespace description	1
2.1.2 Structure	1
2.2 Namespace BioAPI.Data	2
2.2.1 Namespace description	2
2.2.2 Structure	2
3 Data types and constants	3
3.1 Class ACBioParameters	3
3.1.1 Description	3
3.1.2 Properties Summary	3
3.2 Class BFPListElement	3
3.2.1 Description	3
3.2.2 Properties Summary	3
3.3 Class BFPSchema [Serializable()]	3
3.3.1 Description	3
3.3.2 Properties Summary	3
3.4 Class BIR	4
3.4.1 Description	4
3.4.2 Properties Summary	4
3.4.3 Method Summary	5
3.4.3.1 virtual void BIRFromByteArray (byte[] record)	5
3.4.3.2 virtual byte[] BIRToByteArray ()	5
3.4.3.3 virtual void Destroy ()	5
3.5 Class BSPSchema [Serializable()]	6
3.5.1 Description	6
3.5.2 Properties Summary	6
3.6 Class Candidate	7
3.6.1 Description	7
3.6.2 Properties Summary	7
3.7 Class DataTypes	8
3.7.1 Description	8
3.7.2 Enumerations	8
3.7.2.1 BiometricSubtype	8
3.7.2.2 BiometricType	8
3.7.2.3 BIRDatabaseAccess	9
3.7.2.4 BSPSchemaOperations	9
3.7.2.5 BSPSchemaOptions	9
3.7.2.6 EventKind	10
3.7.2.7 Facility	10
3.7.2.8 GUIEnrolType	10
3.7.2.9 GUIMoment	10
3.7.2.10 GUIOperation	11
3.7.2.11 GUIResponse	11
3.7.2.12 GUISuboperation	11
3.7.2.13 ProcessedLevel	11
3.7.2.14 Purpose	11

3.7.2.15	ResultOptions	12
3.7.2.16	SecurityOptionsType	12
3.7.2.17	UnitCategoryType	12
3.7.2.18	UnitIndicatorStatus	12
3.7.2.19	UnitPowerMode	12
3.8	Class Date	13
3.8.1	Description	13
3.8.2	Properties Summary	13
3.8.3	Methods Summary	13
3.8.3.1	virtual bool IsLowerOrEqual (int day, int month, int year)	13
3.8.3.2	virtual bool IsLowerOrEqual (int day, int month, int year, int hour, int minute, int second)	13
3.8.3.3	virtual bool IsLowerOrEqual (Date date)	13
3.8.3.4	virtual bool IsHigherOrEqual (int day, int month, int year)	13
3.8.3.5	virtual bool IsHigherOrEqual (int day, int month, int year, int hour, int minute, int second)	13
3.8.3.6	virtual bool IsHigherOrEqual (Date date)	13
3.9	Class FrameworkSchema	14
3.9.1	Description	14
3.9.2	Properties Summary	14
3.10	Class GUIBitmap	14
3.10.1	Description	14
3.10.2	Properties	14
3.11	Class Identifypopulation	15
3.11.1	Description	15
3.11.2	Properties Summary	15
3.11.3	Method Summary	15
3.11.3.1	virtual void AddMember (PopulationMember member)	15
3.11.3.2	virtual void Destroy ()	15
3.11.3.3	virtual bool IsBound ()	15
3.11.3.4	virtual void Unbind ()	15
3.12	Class PopulationMember	16
3.12.1	Description	16
3.12.2	Properties Summary	16
3.13	Class RegistryID	16
3.13.1	Description	16
3.13.2	Properties Summary	16
3.14	Class SecurityProfileType	16
3.14.1	Description	16
3.14.2	Properties Summary	16
3.15	Class UnitList	17
3.15.1	Description	17
3.15.2	Properties Summary	17
3.15.3	Methods Summary	17
3.15.3.1	void Add (UnitListElement unitListElement)	17
3.15.3.2	int GetUnitID (UnitCategoryType unitCategoryType)	17
3.16	Class UnitListElement	17
3.16.1	Description	17
3.16.2	Properties Summary	17
3.17	Class UnitSchema	18
3.17.1	Description	18
3.17.2	Properties Summary	18
3.18	Class UUID [Serializable()]	19
3.18.1	Description	19
3.18.2	Properties	19
4	Object oriented interfaces for supporting BioAPI_Units	20
4.1	Introduction	20
4.2	Interface IArchive	20
4.2.1	Description	20

4.2.2	Method Summary	20
4.2.2.1	void CloseDatabase (int unitID)	20
4.2.2.2	void DeleteBIR (int unitID, UUID key)	20
4.2.2.3	BIR GetSingleBIR (int unitID, UUID key)	20
4.2.2.4	List<UUID> ListUUIDs (int unitID)	21
4.2.2.5	Identifypopulation NewIdentifyPopulation (int unitID)	21
4.2.2.6	IdentifyPopulation NewIdentifyPopulation (int unitID, List<UUID> UUIDList)	21
4.2.2.7	IdentifyPopulation NewIdentifyPopulation (int unitID, byte[] query)	21
4.2.2.8	void OpenDatabase (int unitID, byte[] databaseID, BIRDatabaseAccess access)	22
4.2.2.9	UUID StoreBIR (int unitID, BIR biometricReference)	22
4.2.2.10	void StoreBIR (int unitID, BIR biometricReference, UUID key)	22
4.2.2.11	UUID StoreBIR (int unitID, BIR biometricReference, byte[] auxiliaryData)	23
4.2.2.12	void StoreBIR (int unitID, BIR biometricReference, byte[] auxiliaryData, UUID key)	23
4.3	Interface IComparison	23
4.3.1	Description	23
4.3.2	Method Summary	24
4.3.2.1	BIR GetAdaptedBIR (int unitID)	24
4.3.2.2	int GetFMRAchieved (int unitID)	24
4.3.2.3	byte[] GetAdditionalData (int unitID)	24
4.3.2.4	List<ICandidate> Identify (int unitID, int maxFMRrequested, BIR processedBIR, bool binning, int maxResults, int timeout)	24
4.3.2.5	List<ICandidate> Identify (int unitID, int maxFMRrequested, BIR processedBIR, List<BIR> auxiliaryBIRs, bool binning, int maxResults, int timeout)	24
4.3.2.6	void PresetIdentifyPopulation (int unitID, Identifypopulation population)	25
4.3.2.7	bool Verify (int unitID, int maxFMRrequested, BIR processedBIR, BIR referenceTemplate, List<ResultOptions> options)	25
4.3.2.8	bool Verify (int unitID, int maxFMRrequested, BIR processedBIR, BIR referenceTemplate, List<BIR> auxiliaryBIRs, List<ResultOptions> options)	25
4.4	Interface IProcessing	26
4.4.1	Description	26
4.4.2	Method Summary	26
4.4.2.1	BIR CreateTemplate (int unitID, BIR capturedBIR, BIR referenceTemplate, RegistryID outputFormat, byte[] additionalData)	26
4.4.2.2	BIR CreateTemplate (List<BIR> capturedBIRs, BIR referenceTemplate, RegistryID outputFormat, byte[] additionalData, int unitID)	26
4.4.2.3	BIR Process (int unitID, BIR capturedBIR, RegistryID outputFormat)	27
4.4.2.4	BIR Process (int unitID, BIR capturedBIR, List<BIR> auxiliaryBIRs, RegistryID outputFormat)	27
4.5	Interface ISensor	27
4.5.1	Description	27
4.5.2	Method Summary	27
4.5.2.1	void Calibrate (int unitID, int timeout)	27
4.5.2.2	BIR Capture (int unitID, List<Purpose> purpose, BiometricSubtype subtype, RegistryID outputFormat, int timeout, byte[] options)	28
4.5.2.3	UnitIndicatorStatus GetIndicatorStatus (int unitID)	28
4.5.2.4	void SetIndicatorStatus (int unitID, UnitIndicatorStatus indicatorStatus)	28
5	BFP level	29
5.1	Interface IBFP	29
5.1.1	Description	29
5.1.2	Imported Interfaces	29
5.1.3	Properties Summary	29
5.1.4	Events Summary	29
5.1.5	Method Summary	29
5.1.5.1	void BFPLoad (BFPEventCallback bfpNotifyCallback)	29

5.1.5.2	byte[] ControlUnit (int unitID, int controlCode, byte[] inputData).....	30
5.1.5.3	List<UnitSchema> QueryUnits ()	30
5.1.5.4	List<UnitSchema> QueryUnits (List<UnitCategoryType> unitCategories)	30
5.1.5.5	void SetPowerMode (int unitID, UnitPowerMode powerMode).....	31
5.1.5.6	void UnitAttach (int unitID).....	31
5.1.5.7	void UnitDetach (int unitID)	31
6	BSP level.....	32
6.1	Interface IBSP.....	32
6.1.1	Description	32
6.1.2	Imported Interfaces.....	32
6.1.3	Properties Summary.....	32
6.1.4	Events Summary.....	32
6.1.5	Method Summary.....	33
6.1.5.1	byte CheckQuality (BIR inputBIR, RegistryID qualityAlgorithmID).....	33
6.1.5.2	byte[] ControlUnit (int unitID, int controlCode, byte[] inputData).....	33
6.1.5.3	UUID Enrol (UnitList unitList, BIR capturedBIR, BIR referenceTemplate, List<Purpose> purpose, BiometricSubtype subtype, RegistryID outputFormat, byte[] additionalData, int timeout, List<ResultOptions> options).....	34
6.1.5.4	UUID Enrol (UnitList unitList, List<BIR> capturedBIRs, BIR referenceTemplate, List<Purpose> purpose, BiometricSubtype subtype, RegistryID outputFormat, byte[] additionalData, int timeout, List<ResultOptions> options)	34
6.1.5.5	UUID Enrol (UnitList unitList, int numberOfSamples, BIR referenceTemplate, List<Purpose> purpose, BiometricSubtype subtype, RegistryID outputFormat, byte[] additionalData, int timeout, List<ResultOptions> options).....	34
6.1.5.6	UUID Enrol (UnitList unitList, BIR capturedBIR, UUID referenceID, List<Purpose> purpose, BiometricSubtype subtype, RegistryID outputFormat, byte[] additionalData, int timeout, List<ResultOptions> options).....	34
6.1.5.7	UUID Enrol (UnitList unitList, List<BIR> capturedBIRs, UUID referenceID, List<Purpose> purpose, BiometricSubtype subtype, RegistryID outputFormat, byte[] additionalData, int timeout, List<ResultOptions> options).....	34
6.1.5.8	UUID Enrol (UnitList unitList, int numberOfSamples, UUID referenceID, List<Purpose> purpose, BiometricSubtype subtype, RegistryID outputFormat, byte[] additionalData, int timeout, List<ResultOptions> options).....	34
6.1.5.9	byte[] GetAuxiliaryData (int unitID).....	35
6.1.5.10	List<ICandidate> IdentifyAggregated (UnitList unitList, int maxFMRrequested, BiometricSubtype subtype, bool binning, int maxResults, int timeout, List<ResultOptions> options)	35
6.1.5.11	List<BFPListElement> QueryBFPs ()	36
6.1.5.12	List<BFPListElement> QueryBFPs (List<UnitCategoryType> unitCategories)	36
6.1.5.13	List<UnitSchema> QueryUnits ()	36
6.1.5.14	List<UnitSchema> QueryUnits (List<UnitCategoryType> unitCategories)	36
6.1.5.15	bool VerifyAggregated (UnitList unitList, int maxFMRrequested, BIR referenceTemplate, BiometricSubtype subtype, int timeout, List<ResultOptions> options)	37
6.1.5.16	bool VerifyAggregated (UnitList unitList, int maxFMRrequested, BIR processedBIR, UUID referenceKey, BiometricSubtype subtype, int timeout, List<ResultOptions> options)	37
6.1.5.17	bool VerifyAggregated (UnitList unitList, int maxFMRrequested, UUID referenceKey, BiometricSubtype subtype, int timeout, List<ResultOptions> options)	37
6.1.5.18	void SetPowerMode (int unitID, UnitPowerMode powerMode).....	37

6.1.5.19	void SubscribeToGUIEvents (GUISelectEventCallback guiSelectEventCallback, GUIStateEventCallback guiStateEventCallback, GUIProgressEventCallback guiProgressEventCallback).....	38
6.1.5.20	void UnsubscribeFromGUIEvents ()	38
7	Framework level	39
7.1	Interface IComponentRegistry	39
7.1.1	Description	39
7.1.2	Method Summary	39
7.1.2.1	void InstallBFP (BFPSchema bfpSchema, bool update).....	39
7.1.2.2	void InstallBSP (BSPSchema bspSchema, bool update).....	39
7.1.2.3	void UninstallBFP (UUID bfpUUID).....	39
7.1.2.4	void UninstallBSP (UUID bspUUID).....	40
7.2	Interface IFramework	40
7.2.1	Description	40
7.2.2	Inherited interfaces	40
7.2.3	Properties Summary	40
7.2.4	Method Summary	40
7.2.4.1	void BSPLoad (UUID bspID, BFPEventCallback notifyCallback, BFPEnumerationCallback bfpEnumeration, String context).....	40
7.2.4.2	void BSPUnload (UUID bspID, String context).....	41
7.2.4.3	void EnableEventNotifications (UUID bspID, List<EventKind> events)	42
7.2.4.4	List<BFPSchema> EnumBFPs ()	42
7.2.4.5	List<BSPSchema> EnumBSPs ().....	43
7.2.4.6	void Init (String version)	43
7.2.4.7	List<BFPListElement> QueryBFPs (UUID bspUUID).....	43
7.2.4.8	List<UnitSchema> QueryUnits (UUID bspUUID).....	44
7.2.4.9	void Terminate ()	44
8	Application interaction.....	45
8.1	class BioAPIException : Exception	45
8.1.1	Description	45
8.1.2	Constructor Summary.....	45
8.1.2.1	public BioAPIException (Facility source, int code).....	45
8.1.2.2	public BioAPIException (Facility source, int code, string message)	45
8.1.2.3	public BioAPIException (Facility source, int code, string message, Exception cause).....	45
8.1.2.4	public BioAPIException (Facility source, int code, Exception cause)	45
8.1.3	Properties Summary	46
8.1.3.1	public int ErrorCode.....	46
8.1.3.2	public Facility Source	46
8.1.4	Method Summary	46
8.1.4.1	public int GetErrorCode()	46
8.1.4.2	public Facility GetSource().....	46
8.2	Callback functions.....	46
8.2.1	Description	46
8.2.2	Callback functions specification	47
8.2.2.1	delegate void BSPEventCallback (object sender, UUID bspUUID, int unitID, UnitSchema unitSchema, EventKind eventKind)	47
8.2.2.2	delegate bool GUISelectEventCallback (object sender, UUID bspUUID, int unitID, GUIEnrolType enrolType, GUIOperation operation, GUIMoment moment, int resultCode, int maxNumEnrollSamples, List<BiometricSubtype> selectableInstances, List<BiometricSubtype> selectedInstances, List<BiometricSubtype> capturedInstances, string text, GUIResponse response).....	47
8.2.2.3	delegate bool GUIStateEventCallback(object sender, UUID bspUUID, int unitID, GUIOperation operation, GUISuboperation suboperation, Purpose purpose, GUIMoment moment, int resultCode, int EnrolSampleIndex, List<GUIBitmap> bitmaps, string text, GUIResponse response, int enrolSampleIndexToRecapture)	48

8.2.2.4	delegate bool GUIProgressEventCallback(object sender, UUID bspUUID, int unitID, GUIOperation operation, GUISuboperation suboperation, Purpose purpose, GUIMoment moment, byte suboperationProgress, List<GUIBitmap> bitmaps, string text, GUIResponse response).....	49
8.2.2.5	delegate void BFPEventCallback(object sender, UUID bfpUUID, int unitID, UnitSchema unitSchema, EventKind eventKind).....	50
8.2.2.6	delegate void BFPGUIProgressEventCallback(object sender, int unitID, String context, List<GUIBitmap> bitmaps, byte response)	50
8.2.2.7	delegate List<BFPSchema> BFPEnumerationCallback().....	51
Annex A (informative)	Namespace BioAPI.Template.....	52
A.1	Introduction	52
A.2	Namespace for BioAPI.Template	52
Annex B (informative)	Calling sequence examples and sample code	54
B.1	Reference Implementation	54
B.2	API Architecture	54

iTeh STANDARD PREVIEW
 (standards.iteh.ai)
 Full standard:
<https://standards.iteh.ai/catalog/standards/sist/9d90fa5f-f712-4c80-866b-3884db3fb3ce/iso-iec-30106-3-2016>

Foreword

ISO (the International Organization for Standardization) is a worldwide federation of national standards bodies (ISO member bodies). The work of preparing International Standards is normally carried out through ISO technical committees. Each member body interested in a subject for which a technical committee has been established has the right to be represented on that committee. International organizations, governmental and non-governmental, in liaison with ISO, also take part in the work. ISO collaborates closely with the International Electrotechnical Commission (IEC) on all matters of electrotechnical standardization.

International Standards are drafted in accordance with the rules given in the ISO/IEC Directives, Part 2.

The main task of technical committees is to prepare International Standards. Draft International Standards adopted by the technical committees are circulated to the member bodies for voting. Publication as an International Standard requires approval by at least 75 % of the member bodies casting a vote.

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. ISO shall not be held responsible for identifying any or all such patent rights.

ISO/IEC 30106-3 was prepared by Technical Committee ISO/TC JTC1, *Information Technology*, Subcommittee SC 37, *Biometrics*.

This second/third/... edition cancels and replaces the first/second/... edition (), [clause(s) / subclause(s) / table(s) / figure(s) / annex(es)] of which [has / have] been technically revised.

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 the standard an application programming interface expressed in C# language is specified. C# 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 the Microsoft .NET platform.

One of the advantages of using C# is that, as it is designed for the CLI (Common Language Infrastructure), allows multiple high-level languages to be used on different computer platforms without being rewritten for specific architectures.

C# shares some features (overloading, some syntactic details, etc.) with C++ but includes new characteristics (reference and output parameters, enumerations, unified type system, etc). Besides, C# is very similar to Java (Interfaces, Exceptions, object-orientation, etc.), which implies that the structure of interfaces and namespaces ((which is the equivalent to packages in Java language)) is mostly the same as Java but, as expected, code implementation and compilation are different.

As Java implementation allows an easy use of Java BSPs, Java-based application servers or Java applets, C# is the best way to write windows desktop and web applications/services and provides an advanced and well designed remote framework.

ITeH STANDARD PREVIEW
(standards.iteh.ai)
Full standard:
<https://standards.iteh.ai/catalog/standards/sist/9d33388f-f712-4c80-866b-3884db3fb3ce/iso-iec-30106-3-2014>

Information Technology — Object Oriented BioAPI — Part 3: C# Implementation

1 Scope

The proposed standard will specify 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 standard will be maintained with ISO/IEC 30106-2 (Java implementation). In spite of the differences in actual parameters passed between functions, the names and interface structure are the same.

2 BioAPI C# Namespace Structure

The BioAPI C# interface will be divided into several namespaces. The following is the namespace structure:

- Namespace BioAPI: Contains functions for the attach sessions and managing units.
- Namespace BioAPI.Data: Contains all the data structures.

2.1 Namespace BioAPI

2.1.1 Namespace description

This namespace contains all the components responsible for managing and executing the functionality of BioAPI, either in a Framework-free deployment or when an OO BioAPI Framework is used. Component Registry interface is also defined in this namespace.

2.1.2 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 they accessible class will be either 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 provide the equivalence to the API interface in ISO/IEC 19784-1.

2.2 Namespace BioAPI.Data

2.2.1 Namespace description

This namespace contains all data structures needed for the implementation of either a Framework-based version of OO BioAPI, or a Framework-free one.

2.2.2 Structure

Several data structures are provided to comply with the requirements of specifying this International Standard. 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 Part 1 of this International Standard.

iTeh STANDARD PREVIEW
(standards.iteh.ai)
Full standard:
<https://standards.iteh.ai/catalog/standards/sist/9d90fa5f-f712-4c80-866b-3884db3fb3ce/iso-iec-30106-3-2016>

3 Data types and constants

3.1 Class ACBioParameters

3.1.1 Description

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

3.1.2 Properties Summary

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

3.2 Class BFPListElement

3.2.1 Description

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

3.2.2 Properties Summary

- UnitCategoryType UnitCategory { get; set; }: The category of the unit
- UUID BFPID { get; set; }: The UUID assigned to the BFP

3.3 Class BFPSchema [Serializable()]

3.3.1 Description

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

3.3.2 Properties Summary

- UUID BFPUUID {get;}: UUID of the BFP
- UnitCategoryType BFPCategory {get;}: Category of the BFP identified by the BFPUUID
- String BFPDescription {get;}: A NULL-terminated string containing a text description of the BFP