DRAFT INTERNATIONAL STANDARD

ISO/IEC DIS 30106-3

ISO/IEC JTC **1**/SC **37** Secretariat: **ANSI**

Voting begins on:
2019-09-23

Voting terminates on:
2019-12-16

Information technology — Object oriented BioAPI —

Part 3:
C# implementation

Technlogies de l'information — Objet orienté BioAPI —

Partie 3: Mise en oeuvre de C#

ICS: 35.240.15

This document is circulated as received from the committee secretariat.

Reference number
ISO/IEC DIS 30106-3:2019(E)

COPYRIGHT PROTECTED DOCUMENT

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

Published in Switzerland

Contents

Page

Foreword ix

Introduction x

1 Scope 1

2 Normative references 1

3 BioAPI C# Namespace Structure 1
3.1 Overall structure 1
3.2 Namespace BioAPI 1
3.2.1 Namespace description 1
3.2.2 Structure 1
3.3 Namespace BioAPI.Data 2
3.3.1 Namespace description 2
3.3.2 Structure 2

4 Data types and constants 3
4.1 Class ACBioParameters 3
4.1.1 Description 3
4.1.2 Properties Summary 3
4.2 Class BFPListElement 3
4.2.1 Description 3
4.2.2 Properties Summary 3
4.3 Class BFPSchema [Serializable()] 3
4.3.1 Description 3
4.3.2 Properties Summary 3
4.3.3 Method Summary 4
4.3.3.1 virtual void Dispose () 4
4.4 Class BIR 4
4.4.1 Description 4
4.4.2 Properties Summary 4
4.4.3 Method Summary 6
4.4.3.1 virtual BIR (byte[] record) 6
4.4.3.2 virtual BIR (RegistryID bDBFormat, bool bDBEncription, bool bIRIntegrity, BiometricType bDBBiometricType, BiometricSubtype bDBBiometricSubtype, RegistryID bDBCaptureDevice, RegistryID bDBFeatureExtractionAlg, RegistryID bDBComparisonAlg, RegistryID bDBCompresionAlg, RegistryID bDBPADTechnique, byte[] bDBChallengeResponse, Date bDBCreationDate, byte[] bDBIndex, ProcessedLevel bDBProcessedLevel, RegistryID bDBProduct, Purpose bDBPurpose, byte bDBQuality, RegistryID bDBQualityAlg, List<Date> bDBValidityPeriod, Date bIRCreationDate, byte[] bIRCreator, byte[] bIRIndex, byte[] bIRPayload, byte[] bIRPointer, List<Date> bIRValidityPeriod, RegistryID sBFormat, byte[] bDBData, byte[] sBData) 6
4.4.3.3 virtual public byte[] ToArray() 6
4.4.3.4 virtual void Dispose () 6
4.5 Class BSPSchema [Serializable()] 7
4.5.1 Description 7
4.5.2 Properties Summary 7
4.5.3 Method Summary 8
4.5.3.1 virtual void Dispose () 8
4.6 Class Candidate 8
4.6.1 Description 8
4.6.2 Properties Summary 8

4.7 Class DataTypes ... 9
4.7.1 Description ... 9
4.7.2 Enumerations ... 9
4.7.2.1 BiometricSubtype ... 9
4.7.2.2 BiometricType ... 9
4.7.2.3 BIRDatabaseAccess ... 10
4.7.2.4 BSPSchemaOperations ... 10
4.7.2.5 BSPSchemaOptions ... 10
4.7.2.6 EventKind ... 11
4.7.2.7 Facility ... 11
4.7.2.8 GUIEnrolType ... 11
4.7.2.9 GUIMoment ... 11
4.7.2.10 GUIOperation ... 12
4.7.2.11 GUIResponse ... 12
4.7.2.12 GUISuboperation ... 12
4.7.2.13 ProcessedLevel ... 12
4.7.2.14 Purpose ... 12
4.7.2.15 ResultOptions ... 13
4.7.2.16 SecurityOptionsType ... 13
4.7.2.17 UnitCategoryType ... 13
4.7.2.18 UnitIndicatorStatus ... 13
4.7.2.19 UnitPowerMode ... 13
4.8 Class Date ... 14
4.8.1 Description ... 14
4.8.2 Properties Summary ... 14
4.8.3 Methods Summary ... 14
4.8.3.1 virtual bool IsLowerOrEqual (int day, int month, int year) ... 14
4.8.3.2 virtual bool IsLowerOrEqual (int day, int month, int year, int hour, int minute, int second) ... 14
4.8.3.3 virtual bool IsLowerOrEqual (Date date) ... 14
4.8.3.4 virtual bool IsHigherOrEqual (int day, int month, int year) ... 14
4.8.3.5 virtual bool IsHigherOrEqual (int day, int month, int year, int hour, int minute, int second) ... 14
4.8.3.6 virtual bool IsHigherOrEqual (Date date) ... 14
4.9 Class FrameworkSchema ... 15
4.9.1 Description ... 15
4.9.2 Properties Summary ... 15
4.9.3 Method Summary ... 15
4.9.3.1 virtual void Dispose () ... 15
4.10 Class GUIBitmap ... 15
4.10.1 Description ... 15
4.10.2 Properties ... 15
4.10.3 Method Summary ... 16
4.10.3.1 virtual void Dispose () ... 16
4.11 Class Identifypopulation ... 16
4.11.1 Description ... 16
4.11.2 Properties Summary ... 16
4.11.3 Method Summary ... 16
4.11.3.1 virtual void AddMember (PopulationMember member) ... 16
4.11.3.2 virtual void Dispose () ... 16
4.11.3.3 virtual bool IsBound () ... 16
4.11.3.4 virtual void Unbind () ... 17
4.12 Class PopulationMember ... 17
4.12.1 Description ... 17
4.12.2 Properties Summary ... 17
4.13 Class RegistryID ... 17
4.13.1 Description ... 17
4.13.2 Properties Summary ... 17
4.14 Class SecurityProfileType ... 17
4.14.1 Description ... 17

4.14.2 Properties Summary 18
4.14.3 Method Summary 18
4.14.3.1 virtual void Dispose () 18
4.15 Class UnitList 18
4.15.1 Description 18
4.15.2 Properties Summary 18
4.15.3 Methods Summary 18
4.15.3.1 void Add (UnitListElement unitListElement) 18
4.15.3.2 int GetUnitID (UnitCategoryType unitCategoryType) 19
4.15.3.3 virtual void Dispose () 19
4.16 Class UnitListElement 19
4.16.1 Description 19
4.16.2 Properties Summary 19
4.17 Class UnitSchema 19
4.17.1 Description 19
4.17.2 Properties Summary 19
4.17.3 Method Summary 20
4.17.3.1 virtual void Dispose () 20
4.18 Class UUID [Serializable()] 20
4.18.1 Description 20
4.18.2 Properties 20

5 Object oriented interfaces for supporting BioAPI_Units 21
5.1 Introduction 21
5.2 Interface IArchive 21
5.2.1 Description 21
5.2.2 Method Summary 21
5.2.2.1 void CloseDatabase (int unitID) 21
5.2.2.2 void DeleteBIR (int unitID, UUID key) 21
5.2.2.3 BIR GetSingleBIR (int unitID, UUID key) 21
5.2.2.4 List<UUID> ListUUIDs (int unitID) 22
5.2.2.5 Identifypopulation NewIdentifyPopulation (int unitID) 22
5.2.2.6 IdentifyPopulation NewIdentifyPopulation (int unitID, List<UUID> UUIDList) 22
5.2.2.7 IdentifyPopulation NewIdentifyPopulation (int unitID, byte[] query) 22
5.2.2.8 void OpenDatabase (int unitID, byte[] databaseID, BIRDatabaseAccess access) 23
5.2.2.9 UUID StoreBIR (int unitID, BIR biometricReference) 23
5.2.2.10 void StoreBIR (int unitID, BIR biometricReference, UUID key) 23
5.2.2.11 UUID StoreBIR (int unitID, BIR biometricReference, byte[] auxiliaryData) 23
5.2.2.12 void StoreBIR (int unitID, BIR biometricReference, byte[] auxiliaryData, UUID key) 24
5.3 Interface IComparison 24
5.3.1 Description 24
5.3.2 Method Summary 25
5.3.2.1 BIR GetAdaptedBIR (int unitID) 25
5.3.2.2 int GetFMRAchieved (int unitID) 25
5.3.2.3 List<ICandidate> Identify (int unitID, int maxFMRrequested, BIR processedBIR, bool binning, int maxResults, int timeout) 25
5.3.2.4 List<ICandidate> Identify (int unitID, int maxFMRrequested, BIR processedBIR, List<BIR> auxiliaryBIRs, bool binning, int maxResults, int timeout) 25
5.3.2.5 void PresetIdentifyPopulation (int unitID, Identifypopulation population) 26
5.3.2.6 bool Verify (int unitID, int maxFMRrequested, BIR processedBIR, BIR referenceTemplate, List<ResultOptions> options) 26
5.3.2.7 bool Verify (int unitID , int maxFMRrequested, BIR processedBIR, BIR referenceTemplate, List<BIR> auxiliaryBIRs, List<ResultOptions> options) 26
5.4 Interface IProcessing 27
5.4.1 Description 27

5.4.2 Method Summary ... 27
5.4.2.1 BIR CreateTemplate (int unitID, BIR capturedBIR, BIR referenceTemplate, RegistryID outputFormat, byte[] additionalData) ... 27
5.4.2.2 BIR CreateTemplate (int unitID, List<BIR> capturedBIRs, BIR referenceTemplate, RegistryID outputFormat, byte[] additionalData, int unitID) ... 27
5.4.2.3 BIR Process (int unitID , BIR capturedBIR, RegistryID outputFormat) ... 28
5.4.2.4 BIR Process (int unitID, BIR capturedBIR, List<BIR> auxiliaryBIRs, RegistryID outputFormat) ... 28
5.4.2.5 byte AnalyseQuality (int unitID , BIR capturedBIR) ... 28
5.5 Interface ISensor ... 28
5.5.1 Description ... 28
5.5.2 Method Summary ... 29
5.5.2.1 void Calibrate (int unitID , int timeout) ... 29
5.5.2.2 BIR Capture (int unitID , List<Purpose> purpose, BiometricSubtype subtype, RegistryID outputFormat, int timeout, List<ResultOptions> options) ... 29
5.5.2.3 UnitIndicatorStatus GetIndicatorStatus (int unitID) ... 29
5.5.2.4 void SetIndicatorStatus (int unitID , UnitIndicatorStatus indicatorStatus) ... 29

6 BFP level ... 31
6.1 Interface IBFP ... 31
6.1.1 Description ... 31
6.1.2 Imported Interfaces ... 31
6.1.3 Properties Summary ... 31
6.1.4 Events Summary ... 31
6.1.5 Method Summary ... 31
6.1.5.1 void BFPLoad (BFPEventCallback bfpNotifyCallback) ... 31
6.1.5.2 void BFPUnload () ... 32
6.1.5.3 byte[] ControlUnit (int unitID, int controlCode, byte[] inputData) ... 32
6.1.5.4 byte[] GetAuxiliaryData (int unitID) ... 33
6.1.5.5 List<UnitSchema> QueryUnits () ... 33
6.1.5.6 List<UnitSchema> QueryUnits (List<UnitCategoryType> unitCategories) ... 33
6.1.5.7 void SetPowerMode (int unitID, UnitPowerMode powerMode) ... 33

7 BSP level ... 34
7.1 Interface IBSP ... 34
7.1.1 Description ... 34
7.1.2 Imported Interfaces ... 34
7.1.3 Properties Summary ... 34
7.1.4 Events Summary ... 34
7.1.5 Method Summary ... 34
7.1.5.1 void BSPLoad (BSPEventCallback bspNotifyCallback) ... 35
7.1.5.2 void BFPUnload () ... 35
7.1.5.3 byte CheckQuality (BIR inputBIR, RegistryID qualityAlgorithmID) ... 35
7.1.5.4 byte[] ControlUnit (int unitID, int controlCode, byte[] inputData) ... 35
7.1.5.5 UUID Enrol (UnitList unitList, BIR capturedBIR, BIR referenceTemplate, List<Purpose> purpose, BiometricSubtype subtype, RegistryID outputFormat, byte[] additionalData, List<ResultOptions> options) ... 36
7.1.5.6 UUID Enrol (UnitList unitList, List<BIR> capturedBIRs, BIR referenceTemplate, List<Purpose> purpose, BiometricSubtype subtype, RegistryID outputFormat, byte[] additionalData, List<ResultOptions> options) ... 36
7.1.5.7 UUID Enrol (UnitList unitList, int numberOfPresentations, int numberOfAttempts, BIR referenceTemplate, List<Purpose> purpose, BiometricSubtype subtype, RegistryID outputFormat, byte[] additionalData, int timeout, List<ResultOptions> options) ... 36
7.1.5.8 UUID Enrol (UnitList unitList, BIR capturedBIR, UUID referenceID, List<Purpose> purpose, BiometricSubtype subtype, RegistryID outputFormat, byte[] additionalData, List<ResultOptions> options) ... 36

7.1.5.9 UUID Enrol (UnitList unitList, List<BIR> capturedBIRs, UUID referenceID, List<Purpose> purpose, BiometricSubtype subtype, RegistryID outputFormat, byte[] additionalData, List<ResultOptions> options) 36
7.1.5.10 UUID Enrol (UnitList unitList, int numberOfPresentations, int numberOfAttempts, UUID referenceID, List<Purpose> purpose, BiometricSubtype subtype, RegistryID outputFormat, byte[] additionalData, int timeout, List<ResultOptions> options) 36
7.1.5.11 byte[] GetAuxiliaryData (int unitID) 37
7.1.5.12 List<ICandidate> IdentifyAggregated (UnitList unitList, int maxFMRrequested, BiometricSubtype subtype, bool binning, int maxResults, int timeout, List<ResultOptions> options) 38
7.1.5.13 List<ICandidate> IdentifyAggregated (UnitList unitList, BIR inputBIR, int maxFMRrequested, BiometricSubtype subtype, bool binning, int maxResults, int timeout, List<ResultOptions> options) 38
7.1.5.14 List<BFPListElement> QueryBFPs () 38
7.1.5.15 List<BFPListElement> QueryBFPs (List<UnitCategoryType> unitCategories) 38
7.1.5.16 List<UnitSchema> QueryUnits () 39
7.1.5.17 List<UnitSchema> QueryUnits (List<UnitCategoryType> unitCategories) 39
7.1.5.18 bool VerifyAggregated (UnitList unitList, int maxFMRrequested, BIR referenceTemplate, BiometricSubtype subtype, int timeout, List<ResultOptions> options) 39
7.1.5.19 bool VerifyAggregated (UnitList unitList, int maxFMRrequested, BIR inputBIR, BIR referenceTemplate, BiometricSubtype subtype, int timeout, List<ResultOptions> options) 39
7.1.5.20 bool VerifyAggregated (UnitList unitList, int maxFMRrequested, UUID referenceKey, BiometricSubtype subtype, int timeout, List<ResultOptions> options) 39
7.1.5.21 bool VerifyAggregated (UnitList unitList, int maxFMRrequested, BIR inputBIR, UUID referenceKey, BiometricSubtype subtype, int timeout, List<ResultOptions> options) 39
7.1.5.22 void SetPowerMode (int unitID, UnitPowerMode powerMode) 40
7.1.5.23 void SubscribeToGUIEvents (GUISelectEventCallback guiSelectEventCallback, GUIStateEventCallback guiStateEventCallback, GUIProgressEventCallback guiProgressEventCallback) 40
7.1.5.24 void UnsubscribeFromGUIEvents () 41

8 Framework level 42
8.1 Interface IComponentRegistry 42
8.1.1 Description 42
8.1.2 Method Summary 42
8.1.2.1 void InstallBFP (BFPSchema bfpSchema, bool update) 42
8.1.2.2 void InstallBSP (BSPSchema bspSchema, bool update) 42
8.1.2.3 void UninstallBFP (UUID bfpUUID) 42
8.1.2.4 void UninstallBSP (UUID bspUUID) 43
8.2 Interface IFramework 43
8.2.1 Description 43
8.2.2 Inherited interfaces 43
8.2.3 Properties Summary 43
8.2.4 Method Summary 43
8.2.4.1 void BSPLoad (UUID bspID, BFPEventCallback notifyCallback, BFPEnumerationCallback bfpEnumeration, String context) 43
8.2.4.2 void BSPUnload (UUID bspID, String context) 44
8.2.4.3 void EnableEventNotifications (UUID bspID, List<EventKind> events) 45
8.2.4.4 List<BFPSchema> EnumBFPs () 45
8.2.4.5 List<BSPSchema> EnumBSPs () 46
8.2.4.6 void Init (String version) 46
8.2.4.7 List<BFPListElement> QueryBFPs (UUID bspUUID) 46
8.2.4.8 List<UnitSchema> QueryUnits (UUID bspUUID) 46
8.2.4.9 void Terminate () 47

9 Application interaction 48
9.1 class BioAPIException : Exception 48
9.1.1 Description 48
9.1.2 Constructor Summary 48
9.1.2.1 public BioAPIException (Facility source, int code) 48
9.1.2.2 public BioAPIException (Facility source, int code, string message) 48
9.1.2.3 public BioAPIException (Facility source, int code, string message, Exception cause) 48
9.1.2.4 public BioAPIException (Facility source, int code, Exception cause) 48
9.1.3 Properties Summary 49
9.1.3.1 public int ErrorCode 49
9.1.3.2 public Facility Source 49
9.1.4 Method Summary 49
9.1.4.1 public int GetErrorCode() 49
9.1.4.2 public Facility GetSource() 49
9.2 Callback functions 49
9.2.1 Description 49
9.2.2 Callback functions specification 50
9.2.2.1 delegate void BSPEventCallback (object sender, UUID bspUUID, int unitID, UnitSchema unitSchema, EventKind eventKind) 50
9.2.2.2 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) 50
9.2.2.3 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) 51
9.2.2.4 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) 52
9.2.2.5 delegate void BFPEventCallback(object sender, UUID bfpUUID, int unitID, UnitSchema unitSchema, EventKind eventKind) 53
9.2.2.6 delegate void BFPGUIProgressEventCallback(object sender, int unitID, String context, List<GUIBitmap> bitmaps, byte response) 53
9.2.2.7 delegate List<BFPSchema> BFPEnumerationCallback() 54

Annex A (informative) Calling sequence examples and sample code 55
A.1 Reference Implementation 55
A.2 API Architecture 55

Foreword

ISO (the International Organization for Standardization) and IEC (the International Electrotechnical Commission) form the specialized system for worldwide standardization. National bodies that are members of ISO or IEC participate in the development of International Standards through technical committees established by the respective organization to deal with particular fields of technical activity. ISO and IEC technical committees collaborate in fields of mutual interest. Other international organizations, governmental and non-governmental, in liaison with ISO and IEC, also take part in the work. In the field of information technology, ISO and IEC have established a joint technical committee, ISO/IEC JTC 1.

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

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

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

ISO/IEC 30106-3 was prepared by Joint Technical Committee ISO/IEC JTC 1, *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.

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 Normative references

The following referenced documents are indispensable for the application of this document. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

ISO/IEC 30106-1, Information Technology – BioAPI for object oriented programming languages

3 BioAPI C# Namespace Structure

3.1 Overall structure

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

- Namespace BioAPI: Contains functionality to manage units, BSPs, BFPs, the Framework and Applications.
- Namespace BioAPI.Data: Contains all the data structures.

3.2 Namespace BioAPI

3.2.1 Namespace description

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

3.2.2 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.

3.3 Namespace BioAPI.Data

3.3.1 Namespace description

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

3.3.2 Structure

Several data structures are provided to comply with the requirements of specifying this Intenational 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.

4 Data types and constants

4.1 Class ACBioParameters

4.1.1 Description

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

4.1.2 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 accepted by 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.

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 Properties Summary

— UnitCategoryType UnitCategory { get; set; }: The category of the unit

— UUID BFPID { get; set; }: The UUID assigned to the BFP

4.3 Class BFPSchema [Serializable()]

4.3.1 Description

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

4.3.2 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