

DRAFT INTERNATIONAL STANDARD

ISO/IEC DIS 30106-2

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 2: Java implementation

Titre manque

ICS: 35.040

iTeh STANDARD PREVIEW
(standards.iteh.ai)
Full standard:
<https://standards.iteh.ai/catalog/standards/sist/a8b47ef-8b70-4d46-831b-b943383b28b1/iso-iec-30106-2-2016>

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 30106-2:2014(E)

© ISO/IEC 2014

iTeh STANDARD PREVIEW
(standards.iteh.ai)
Full standard:
<https://standards.iteh.ai/catalog/standards/sist/a8b47ef-8b70-4d46-831b-b943383b28b1/iso-iec-30106-2-2016>

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	xi
Introduction.....	xii
1 Scope	1
2 BioAPI Java Package Structure	1
2.1 Package org.bioapi	1
2.1.1 Package description	1
2.1.2 Structure	1
2.2 Package org.bioapi.data	2
2.2.1 Package 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 Method Summary	3
3.1.2.1 int[] getChallenge()	3
3.1.2.2 int[] getInitialBPUIIndexOutput()	3
3.1.2.3 int[] getSupremumBPUIIndexOutput()	3
3.2 Class BFPListElement	3
3.2.1 Description	3
3.2.2 Method Summary	3
3.2.2.1 UUID getBFPID()	3
3.2.2.2 UnitCategoryType getUnitCategory()	4
3.2.2.3 void setBFPID(UUID bfpID)	4
3.2.2.4 void setUnitCategory(UnitCategoryType unitCategory)	4
3.3 Class BFPSchema	4
3.3.1 Description	4
3.3.2 Method Summary	4
3.3.2.1 String getBFPDescription()	4
3.3.2.2 Vector<RegistryID> getBFPSupportedFormats()	4
3.3.2.3 UUID getBFPUUID()	4
3.3.2.4 Vector<BiometricType> getFactorsMask()	4
3.3.2.5 byte[] getFWProperty()	5
3.3.2.6 UUID getFWPropertyID()	5
3.3.2.7 String getPath()	5
3.3.2.8 String getProductVersion()	5
3.3.2.9 String getSpecVersion()	5
3.3.2.10 UnitCategoryType getUnitCategory()	5
3.3.2.11 String getVendor()	6
3.4 Class BIR	6
3.4.1 Description	6
3.4.2 Method Summary	6
3.4.2.1 void birFromArray(byte[] record)	6
3.4.2.2 byte[] birToArray()	6
3.4.2.3 void destroy()	6
3.4.2.4 BiometricSubtype getBDBBiometricSubtype()	6
3.4.2.5 BiometricType getBDBBiometricType()	7
3.4.2.6 byte[] getBDBChallengeResponse()	7
3.4.2.7 Date getBDBCreationDate()	7
3.4.2.8 byte[] getBDBData()	7

3.4.2.9	RegistryID getBDBFormat().....	7
3.4.2.10	byte[] getBDBIndex().....	7
3.4.2.11	ProcessedLevel getBDBProcessedLevel().....	7
3.4.2.12	Purpose getBDBPurpose().....	7
3.4.2.13	byte getBDBQuality().....	8
3.4.2.14	Vector<Date> getBDBValidityPeriod().....	8
3.4.2.15	Date getBIRCreationDate().....	8
3.4.2.16	byte[] getBIRCreator().....	8
3.4.2.17	byte[] getBIRIndex().....	8
3.4.2.18	byte[] getBIRAdditionalData().....	8
3.4.2.19	Vector<Date> getBIRValidityPeriod().....	8
3.4.2.20	byte getCBEFFVersion().....	9
3.4.2.21	RegistryID getPatronFormat().....	9
3.4.2.22	byte getPatronHeaderVersion().....	9
3.4.2.23	byte[] getSBData().....	9
3.4.2.24	RegistryID getSBFormat().....	9
3.4.2.25	boolean hasBDBEncryption().....	9
3.4.2.26	boolean hasBDBIntegrity().....	9
3.4.2.27	boolean isBIRSigned().....	10
3.4.2.28	boolean isQualitySupported().....	10
3.4.2.29	boolean isQualityKnown().....	10
3.4.2.30	void setBDBBiometricSubtype(BiometricSubtype bdbBiometricSubtype).....	10
3.4.2.31	void setBDBBiometricType(BiometricType bdbBiometricType).....	10
3.4.2.32	void setBDBChallengeResponse(byte bdbChallengeResponse).....	10
3.4.2.33	void setBDBCreationDate(Date bdbCreationDate).....	10
3.4.2.34	void setBDBEncryption(boolean bdbEncryption).....	10
3.4.2.35	void setBDBFormat(RegistryID bdbFormat).....	10
3.4.2.36	void setBDBData(byte[] bdbData).....	11
3.4.2.37	void setBDBIndex(byte[] bdbIndex).....	11
3.4.2.38	void setBDBIntegrity(boolean bdbIntegrity).....	11
3.4.2.39	void setBDBQuality(byte bdbQuality).....	11
3.4.2.40	void setBDBProcessedLevel(ProcessedLevel bdbProcessedLevel).....	11
3.4.2.41	void setBDBPurpose(Purpose bdbPurpose).....	11
3.4.2.42	void setBDBValidityPeriod(Vector<Date> bdbValidityPeriod).....	11
3.4.2.43	void setBIRCreationDate(Date birCreationDate).....	11
3.4.2.44	void setBIRCreator(byte[] birCreator).....	12
3.4.2.45	void setBIRIndex(byte[] birIndex).....	12
3.4.2.46	void setBIRAdditionalData(byte[] birAdditionalData).....	12
3.4.2.47	void setBIRValidityPeriod(Vector<Date> birValidityPeriod).....	12
3.4.2.48	void setCBEFFVersion(byte cbeffVersion).....	12
3.4.2.49	void setPatronFormat(RegistryID patronFormat).....	12
3.4.2.50	void setPatronHeaderVersion(byte patronHeaderVersion).....	12
3.4.2.51	void setSBData(byte[] sbData).....	13
3.4.2.52	void setSBFormat(RegistryID sbFormat).....	13
3.5	Class BSPSchema.....	13
3.5.1	Description.....	13
3.5.2	Method Summary.....	13
3.5.2.1	UUID getBSPAccessUUID().....	13
3.5.2.2	String getBSPDescription().....	13
3.5.2.3	Vector<RegistryID> getBSPSupportedAlgorithms().....	13
3.5.2.4	Vector<RegistryID> getBSPSupportedFormats().....	13
3.5.2.5	Vector<UUID> getBSPSupportedTransformOperation().....	14
3.5.2.6	UUID getBSPUUID().....	14
3.5.2.7	int getDefaultCalibrateTimeout().....	14
3.5.2.8	int getDefaultCaptureTimeout().....	14
3.5.2.9	int getDefaultEnrolTimeout().....	14
3.5.2.10	int getDefaultIdentifyTimeout().....	14
3.5.2.11	int getDefaultVerifyTimeout().....	14
3.5.2.12	Vector<BiometricType> getFactorsMask().....	15
3.5.2.13	byte[] getHostingEndpointIRI().....	15

3.5.2.14	int getMaxBSPDbSize().....	15
3.5.2.15	int getMaxIdentify()	15
3.5.2.16	int getMaxNumEnrollInstances()	15
3.5.2.17	int getMaxAdditionalDataSize()	15
3.5.2.18	Vector<BSPSchemaOperations> getOperations().....	16
3.5.2.19	Vector<BSPSchemaOptions> getOptions()	16
3.5.2.20	String getPath()	16
3.5.2.21	int getAdditionalDataPolicy()	16
3.5.2.22	String getProductVersion()	16
3.5.2.23	String getSpecVersion()	16
3.5.2.24	String getVendor().....	17
3.6	Class Candidate.....	17
3.6.1	Description	17
3.6.2	Method Summary	17
3.6.2.1	int getFMRAchieved()	17
3.6.2.2	UUID getKey()	17
3.6.2.3	void setFMRAchieved(int fmrAchieved)	17
3.6.2.4	void setKey(UUID key).....	17
3.7	Class DataTypes.....	18
3.7.1	Description	18
3.7.2	Enumerations.....	18
3.7.2.1	BiometricSubtype.....	18
3.7.2.2	BiometricType	18
3.7.2.3	BIRDatabaseAccess.....	19
3.7.2.4	BSPSchemaOperations.....	19
3.7.2.5	BSPSchemaOptions.....	19
3.7.2.6	EventKind.....	20
3.7.2.7	Facility	20
3.7.2.8	GUIEnrolType	20
3.7.2.9	GUIMoment	20
3.7.2.10	GUIOperation	21
3.7.2.11	GUIResponse.....	21
3.7.2.12	GUISuboperation.....	21
3.7.2.13	ProcessedLevel	21
3.7.2.14	Purpose	21
3.7.2.15	ResultOptions.....	22
3.7.2.16	SecurityOptionsType	22
3.7.2.17	UnitCategoryType	22
3.7.2.18	UnitIndicatorStatus	22
3.7.2.19	UnitPowerMode	22
3.8	Class Date	23
3.8.1	Description	23
3.8.2	Method Summary	23
3.8.2.1	int getDayOfMonth().....	23
3.8.2.2	int getHour().....	23
3.8.2.3	int getMinute().....	23
3.8.2.4	int getMonth()	23
3.8.2.5	int getSecond()	23
3.8.2.6	int getYear()	23
3.8.2.7	boolean isLowerOrEqual (int day, int month, int year)	24
3.8.2.8	boolean isLowerOrEqual (int day, int month, int year, int hour, int minute, int second)	24
3.8.2.9	boolean isLowerOrEqual (Date date)	24
3.8.2.10	boolean isHigherOrEqual (int day, int month, int year).....	24
3.8.2.11	boolean isHigherOrEqual (int day, int month, int year, int hour, int minute, int second)	24
3.8.2.12	boolean isHigherOrEqual (Date date)	24
3.8.2.13	void setDayOfMonth(int dayOfMonth)	24
3.8.2.14	void setHour(int hour).....	24
3.8.2.15	void setMinute(int minute)	24

3.8.2.16	void setMonth(int month)	25
3.8.2.17	void setSecond(int second).....	25
3.8.2.18	void setYear(int year)	25
3.9	Class FrameworkSchema	25
3.9.1	Description	25
3.9.2	Method Summary.....	25
3.9.2.1	UUID getFrameworkUUID().....	25
3.9.2.2	String getFWDescription()	25
3.9.2.3	byte[] getFWProperty().....	25
3.9.2.4	UUID getFWPropertyID()	26
3.9.2.5	byte[] getPath()	26
3.9.2.6	String getProductVersion().....	26
3.9.2.7	String getSpecVersion().....	26
3.9.2.8	String getVendor()	26
3.10	Class GUIBitmap.....	26
3.10.1	Description	26
3.10.2	Method Summary.....	26
3.10.2.1	BIRSubtype getBIRSubtype()	26
3.10.2.2	int getHeight()	27
3.10.2.3	byte[][] getPixel()	27
3.10.2.4	int getWidth().....	27
3.11	Class IdentifyPopulation	27
3.11.1	Description	27
3.11.2	Method Summary.....	27
3.11.2.1	void addMember(PopulationMember member).....	27
3.11.2.2	void destroy().....	27
3.11.2.3	Vector<PopulationMember> getIdentifyPopulation()	28
3.11.2.4	bool isBound()	28
3.11.2.5	void unbind().....	28
3.12	Class PopulationMember	28
3.12.1	Description	28
3.12.2	Method Summary.....	28
3.12.2.1	UUID getKey().....	28
3.12.2.2	BIR getTemplate().....	28
3.12.2.3	void setKey(UUID key).....	28
3.12.2.4	void setTemplate(BIR template).....	29
3.13	Class RegistryID	29
3.13.1	Description	29
3.13.2	Method Summary.....	29
3.13.2.1	short getOwner().....	29
3.13.2.2	short getType()	29
3.13.2.3	void setOwner(short owner).....	29
3.13.2.4	void setType(short type).....	29
3.14	Class SecurityProfileType	29
3.14.1	Description	29
3.14.2	Method Summary.....	30
3.14.2.1	Vector<BSPSchemaOptions> acBioOption()	30
3.14.2.2	byte[] getENCInfo().....	30
3.14.2.3	byte[] getHASHAlgForACBio()	30
3.14.2.4	byte[] getMACAlgForACBio()	30
3.14.2.5	byte[] getMACInfo()	30
3.14.2.6	byte[] getSIGAlg()	30
3.14.2.7	byte[] getSIGAlgForACBio().....	31
3.14.2.8	Vector<SecurityOptionsType> getSupportedSecurityOptions()	31
3.15	Class UnitListElement	31
3.15.1	Description	31
3.15.2	Method Summary.....	31
3.15.2.1	UnitCategoryType getUnitCategory()	31
3.15.2.2	int getUnitID().....	31
3.15.2.3	void setUnitCategory(UnitCategoryType unitCategory)	31

3.15.2.4	void setUnitID(int unitID)	31
3.16	Class UnitSchema	32
3.16.1	Description	32
3.16.2	Method Summary	32
3.16.2.1	UUID getBSPUUID()	32
3.16.2.2	String getFirmwareVersion()	32
3.16.2.3	String getHardwareSerialNumber()	32
3.16.2.4	String getHardwareVersion()	32
3.16.2.5	int getMaxBSPDbSize()	32
3.16.2.6	int getMaxIdentify()	32
3.16.2.7	Vector<SecurityProfileType> getSecurityProfile()	33
3.16.2.8	String getSoftwareVersion()	33
3.16.2.9	Vector<EventKind> getSupportedEvents()	33
3.16.2.10	UnitCategoryType getUnitCategory()	33
3.16.2.11	int getUnitID()	33
3.16.2.12	UUID getUnitManagerUUID()	33
3.16.2.13	UUID getUnitProperties()	33
3.16.2.14	byte[] getUnitProperty()	33
3.16.2.15	UUID getUnitPropertyID()	34
3.16.2.16	String getVendorInformation()	34
3.16.2.17	boolean isAuthenticatedHardware()	34
3.16.2.18	void setBSPUUID(UUID bspUUID)	34
3.16.2.19	void setUnitID(int unitID)	34
3.17	Class UUID	34
3.17.1	Description	34
4	Object oriented interfaces for supporting BioAPI_Units	35
4.1	Introduction	35
4.2	Interface Archive	35
4.2.1	Description	35
4.2.2	Method Summary	35
4.2.2.1	void closeDatabase ()	35
4.2.2.2	void deleteBIR (UUID key)	35
4.2.2.3	BIR getSingleBIR (UUID key)	35
4.2.2.4	Vector<UUID> listUUIDs ()	36
4.2.2.5	void openDatabase (byte[] databaseID, BIRDatabaseAccess access)	36
4.2.2.6	UUID storeBIR (BIR biometricReference)	36
4.2.2.7	UUID storeBIR (BIR biometricReference, byte[] auxiliaryData)	36
4.2.2.8	void storeBIR (BIR biometricReference, UUID key)	37
4.2.2.9	void storeBIR (BIR biometricReference, byte[] auxiliaryData, UUID key)	37
4.2.2.10	IdentifyPopulation newIdentifyPopulation ()	37
4.2.2.11	IdentifyPopulation newIdentifyPopulation (byte[] query)	37
4.2.2.12	IdentifyPopulation newIdentifyPopulation (Vector<UUID> uuidList)	38
4.3	Interface Comparison	38
4.3.1	Description	38
4.3.2	Method Summary	38
4.3.2.1	Vector<Candidate> identify (int maxFMRrequested, BIR processedBIR, boolean binning, int maxResults, int timeout)	38
4.3.2.2	Vector<Candidate> identify (int maxFMRrequested, BIR processedBIR, Vector<BIR> auxiliaryBIRs, boolean binning, int maxResults, int timeout)	38
4.3.2.3	void presetIdentifyPopulation (IdentifyPopulation population)	39
4.3.2.4	boolean verify (int maxFMRrequested, BIR processedBIR, BIR referenceTemplate, Vector<ResultOptions> options)	39
4.3.2.5	boolean verify (int maxFMRrequested, BIR processedBIR, BIR referenceTemplate, Vector<BIR> auxiliaryBIRs, Vector<ResultOptions> options)	39
4.3.2.6	BIR getAdaptedBIR()	40
4.3.2.7	int getFMRAchieved()	40
4.3.2.8	byte[] getAdditionalData()	40
4.4	Interface Processing	41

4.4.1	Description	41
4.4.2	Method Summary	41
4.4.2.1	BIR createTemplate (BIR capturedBIR, BIR referenceTemplate, RegistryID outputFormat, byte[] additionalData)	41
4.4.2.2	BIR createTemplate (Vector<BIR> capturedBIRs, BIR referenceTemplate, RegistryID outputFormat, byte[] additionalData)	41
4.4.2.3	BIR createTemplate (BIR capturedBIR, BIR referenceTemplate, Vector<BIR> auxBIRs, RegistryID outputFormat, byte[] additionalData)	41
4.4.2.4	BIR createTemplate (Vector<BIR> capturedBIRs, BIR referenceTemplate, Vector<BIR> auxBIRs, RegistryID outputFormat, byte[] additionalData)	41
4.4.2.5	BIR process (BIR capturedBIR, RegistryID outputFormat)	41
4.4.2.6	BIR process (BIR captureBIR, Vector<BIR> auxiliaryBIRs, RegistryID outputFormat)	41
4.5	Interface Sensor	42
4.5.1	Description	42
4.5.2	Method Summary	42
4.5.2.1	void calibrate (int timeout)	42
4.5.2.2	BIR capture (Vector<Purpose> purpose, BiometricSubtype subtype, RegistryID outputFormat, int timeout, byte[] options)	42
4.5.2.3	UnitIndicatorStatus getIndicatorStatus()	43
4.5.2.4	void setIndicatorStatus (UnitIndicatorStatus indicatorStatus)	43
5	BFP level	44
5.1	Interface BFP	44
5.1.1	Description	44
5.1.2	Imported Interfaces	44
5.1.3	Method Summary	44
5.1.3.1	void bfpLoad (BFPEventListener bfpEventListener, BFPGUIProgressListener bfpGUIProgressListener)	44
5.1.3.2	byte[] controlUnit (int unitID, int controlCode, byte[] inputData)	45
5.1.3.3	byte[] getACBioInstance(int unitID)	45
5.1.3.4	byte[] getAuxiliaryData(int unitID)	45
5.1.3.5	BFPSchema getBFPSchema()	46
5.1.3.6	Vector<UnitSchema> queryUnits ()	46
5.1.3.7	void setPowerMode (int unitID, UnitPowerMode powerMode)	46
5.1.3.8	void unitAttach (int unitID)	46
5.1.3.9	void unitDetach (int unitID)	47
6	BSP level	48
6.1	Interface BSP	48
6.1.1	Description	48
6.1.2	Imported Interfaces	48
6.1.3	Method Summary	48
6.1.3.1	void bspAttach (String bioAPIVersion, ACBioParameters acBioParameters, Vector<UnitListElement> unitList, Vector<SecurityProfileType> securityProfileList)	48
6.1.3.2	void bspDetach ()	49
6.1.3.3	void bspLoad (BSPEventListener bspEventListener, BFPEventListener bfpEventListener, BFPGUIProgressListener bfpGUIProgressListener, BFPEnumerationListener bfpEnumerationListener)	49
6.1.3.4	void bspUnload ()	50
6.1.3.5	byte checkQuality (BIR inputBIR, RegistryID qualityAlgorithmID)	50
6.1.3.6	byte[] controlUnit (int unitID, int controlCode, byte[] inputData)	50
6.1.3.7	UUID enrol (int archiveUnitID, int processingUnitID, BIR capturedBIR, BIR referenceTemplate, Vector<Purpose> purpose, BiometricSubtype subtype, RegistryID outputFormat, byte[] additionalData, int timeout, Vector<ResultOptions> options)	51
6.1.3.8	UUID enrol (int archiveUnitID, int processingUnitID, Vector<BIR> capturedBIRs, BIR referenceTemplate, Vector<Purpose> purpose, BiometricSubtype subtype, RegistryID outputFormat, byte[] additionalData, int timeout, Vector<ResultOptions> options)	51

6.1.3.9	UUID enrol (int numberOfSamples, int sensorUnitID, int archiveUnitID, int processingUnitID, BIR referenceTemplate, Vector<Purpose> purpose, BiometricSubtype subtype, RegistryID outputFormat, byte[] additionalData, int timeout, Vector<ResultOptions> options)	51
6.1.3.10	UUID enrol (int archiveUnitID, int processingUnitID, BIR capturedBIR, UUID referenceID, Vector<Purpose> purpose, BiometricSubtype subtype, RegistryID outputFormat, byte[] additionalData, int timeout, Vector<ResultOptions> options)	51
6.1.3.11	UUID enrol (int archiveUnitID, int processingUnitID, Vector<BIR> capturedBIRs, UUID referenceID, Vector<Purpose> purpose, BiometricSubtype subtype, RegistryID outputFormat, byte[] additionalData, int timeout, Vector<ResultOptions> options)	51
6.1.3.12	UUID enrol (int numberOfSamples, int sensorUnitID, int archiveUnitID, int processingUnitID, UUID referenceID, Vector<Purpose> purpose, BiometricSubtype subtype, RegistryID outputFormat, byte[] additionalData, int timeout, Vector<ResultOptions> options)	51
6.1.3.13	byte[] getAuxiliaryData(int unitID)	52
6.1.3.14	BSPSchema getBSPSchema()	52
6.1.3.15	Vector<Candidate> identifyAggregated (int sensorUnitID, int archiveUnitID, int processingUnitID, int comparisonUnitID, int int maxFMRrequested, BiometricSubtype subtype, boolean binning, int maxResults, int timeout, Vector<ResultOptions> options)	52
6.1.3.16	Vector<BFPListElement> queryBFPs ()	53
6.1.3.17	Vector<BFPListElement> queryBFPs (Vector<UnitCategoryType> unitCategories)	53
6.1.3.18	Vector<UnitSchema> queryUnits ()	53
6.1.3.19	Vector<UnitSchema> queryUnits (Vector<UnitCategoryType> unitCategories)	53
6.1.3.20	void setPowerMode (int unitID, UnitPowerMode powerMode)	54
6.1.3.21	void subscribeToGUIEvents (GUISelectEventListener guiSelectEventListener, GUIStateEventListener guiStateEventListener, GUIProgressEventListener guiProgressEventListener)	54
6.1.3.22	void unsubscribeFromGUIEvents ()	54
6.1.3.23	boolean verifyAggregated (int sensorUnitID, int archiveUnitID, int processingUnitID, int comparisonUnitID, int maxFMRrequested, BIR referenceTemplate, BiometricSubtype subtype, int timeout, Vector<ResultOptions> options)	54
6.1.3.24	boolean verifyAggregated (int archiveUnitID, int comparisonUnitID, int maxFMRrequested, BIR processedBIR, UUID referenceKey, BiometricSubtype subtype, int timeout, Vector<ResultOptions> options)	54
6.1.3.25	boolean verifyAggregated (int sensorUnitID, int archiveUnitID, int processingUnitID, int comparisonUnitID, int maxFMRrequested, UUID referenceKey, BiometricSubtype subtype, int Timeout, Vector<ResultOptions> options)	54
7	Framework Level	56
7.1	Interface ComponentRegistry	56
7.1.1	Description	56
7.1.2	Method Summary	56
7.1.2.1	void installBFP (BFPSchema bfpSchema, boolean update)	56
7.1.2.2	void installBSP (BSPSchema bspSchema, boolean update)	56
7.1.2.3	void uninstallBFP (UUID bfpUUID)	56
7.1.2.4	void uninstallBSP (UUID bspUUID)	57
7.2	Interface Framework	57
7.2.1	Description	57
7.2.2	Inherited interfaces	57
7.2.3	Method Summary	57
7.2.3.1	AttachedSession bspAttach (UUID bspUUID, String bioAPIVersion, ACBioParameters acBioParameters, Vector<UnitListElement> unitList, Vector<SecurityProfileType> securityProfileList)	57

7.2.3.2	void bspDetach (AttachedSession sessionAttached).....	59
7.2.3.3	void bspLoad (UUID bspID, BFPEventListener bfpEventListener, BFGUIProgressListener bfpGUIProgressListener, BFPEnumerationListener bfpEnumerationListener, String context).....	59
7.2.3.4	void bspUnload (UUID bspID, String context).....	60
7.2.3.5	void enableEventNotifications (UUID bspID, Vector<EventKind> events).....	61
7.2.3.6	Vector<BFPSchema> enumBFPs ().....	61
7.2.3.7	Vector<BSPSchema> enumBSPs ().....	61
7.2.3.8	Vector< AttachedSession > getAttachedSessionList().....	62
7.2.3.9	FrameworkSchema getFrameworkSchema().....	62
7.2.3.10	void init (String version).....	62
7.2.3.11	Vector<BFPListElement> queryBFPs (UUID bspUUID).....	62
7.2.3.12	Vector<UnitSchema> queryUnits (UUID bspUUID).....	63
7.2.3.13	void terminate ().....	63
8	Application interaction.....	64
8.1	class BioAPIException extends Exception.....	64
8.1.1	Description.....	64
8.1.2	Constructor Summary.....	64
8.1.2.1	public BioAPIException (Facility source, int code).....	64
8.1.2.2	public BioAPIException (Facility source, int code, String message).....	64
8.1.2.3	public BioAPIException (Facility source, int code, String message, Exception cause).....	64
8.1.2.4	public BioAPIException (Facility source, int code, Exception cause).....	64
8.1.3	Method Summary.....	65
8.1.3.1	public int getErrorCode ().....	65
8.1.3.2	public Facility getSource ().....	65
8.2	GUI callback functions.....	65
8.2.1	Description.....	65
8.2.2	Callback interface specification.....	65
8.2.2.1	Interface GUIProgressEventListener.....	65
8.2.2.2	Interface GUISelectEventListener.....	66
8.2.2.3	Interface GUIStateEventListener.....	67
9	BSP Interaction.....	69
9.1	Interface BSPEventListener.....	69
9.1.1	Method Summary.....	69
9.1.1.1	boolean bspEventCallback (UUID bfpUUID, int unitID, UnitSchema unitSchema, EventKind event).....	69
10	BFP Interaction.....	70
10.1	Interface BFPEnumerationListener.....	70
10.1.1	Method Summary.....	70
10.1.1.1	Vector<BFPSchema> bfpEnumerationCallback().....	70
10.2	Interface BFPEventListener.....	70
10.2.1	Method Summary.....	70
10.2.1.1	boolean bfpEventCallback (UUID bfpUUID, int unitID, UnitSchema unitSchema, EventKind event).....	70
10.3	Interface BFGUIProgressEventListener.....	70
10.3.1	Method Summary.....	70
10.3.1.1	void bfpGUIProgressEventCallback (int unitID, String context, Vector<GUIBitmap> bitmaps, byte response).....	70
Annex A (informative)	Java Requirements.....	72
Annex B (informative)	Calling Sequence Examples and Sample Code.....	73
B.1	Reference Implementation.....	73
B.2	API Architecture.....	73
Annex C (informative)	Current Code Status.....	74
Annex D (informative)	Mapping of ISO/IEC 19784-1 to BioAPI Java.....	75

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 30106-2 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 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 is the best way to write desktop and web applications/services and this specification provides an advanced and well designed remote framework.

NOTE 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.

iTeh STANDARD PREVIEW
(standards.iteh.ai)
Full standard:
<https://standards.iteh.ai/catalog/standards/sist/a8b47ef-8b70-4d46-831b-b943383b28b1/iso-iec-30106-2-2016>

Information Technology — Object Oriented BioAPI — Part 2: Java Implementation

1 Scope

The proposed standard will specify 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 document.

2 BioAPI Java Package Structure

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

- Package org.bioapi: Contains functions for the attach sessions and managing units.
- Package org.bioapi.data: Contains all the data structures.

2.1 Package org.bioapi

2.1.1 Package description

This package 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 package.

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.