INTERNATIONAL STANDARD

ISO/IEC 24709-1

First edition
2007-02-15

Information technology — Conformance testing for the biometric application programming interface (BioAPI) —

Part 1: Methods and procedures

Technologies de l'information — Essai de conformité pour l'interface de programmation d'applications biométriques (BioAPI) —

Partie 1: Méthodes et procédures

Reference number
ISO/IEC 24709-1:2007(E)

Contents

Page

Foreword vii
Introduction viii
1 Scope 1
2 Conformance 1
3 Normative references 2
4 Terms and definitions 2
5 Abbreviations 4
6 Conformance testing methodology 5
6.1 General 5
6.2 Conformance testing models 9
6.3 Abstract test engine 14
7 General properties of the assertion language 15
7.1 General 15
7.2 Variables 18
7.3 Built-in variables 18
7.4 Representation of integers 19
7.5 Representation of booleans 19
7.6 Representation of universally unique identifiers 19
7.7 Representation of octet strings 20
7.8 XML documents 20
8 Elements of the assertion language 20
8.1 Element <package> 20
8.2 Element <assertion> (child of <package>) 22
8.3 Element <input> (child of <assertion>) 23
8.4 Element <invoke> (child of <assertion>) 24
8.5 Element <input> (child of <invoke>) 25
8.6 Element <output> (child of <invoke>) 26
8.7 Element <return> (child of <invoke>) 27
8.8 Element <bind> (child of <assertion>) 28
8.9 Element <activity> (child of <package>) 29
8.10 Element <input> (child of <activity>) 35
8.11 Element <output> (child of <activity>) 36
8.12 Element <set> 36
8.13 Element <add> 37
8.14 Element <subtract> 38
8.15 Element <invoke> (child of <activity>) 39
8.16 Element <only_if> 42
8.17 Element <wait_until> 43
8.18 Element <assert_condition> 45
8.19 Element <and> 46
8.20 Element <or> 47
8.21 Element <xor> 47
8.22 Element <not> 48
8.23 Element <equal_to> 49
8.24 Element <not_equal_to> 50
8.25 Element <greater_than> 50

8.26 Element <greater_than_or_equal_to> 50
8.27 Element <less_than> 50
8.28 Element <less_than_or_equal_to> 51
8.29 Element <same_as> 51
8.30 Element <not_same_as> 52
8.31 Element <existing> 52
8.32 Element <not_existing> 52

9 Standard interface functions 53
9.1 General 53
9.2 Parameter groups 58
9.3 BioAPI_Init 87
9.4 BioAPI_Terminate 88
9.5 BioAPI_GetFrameworkInfo 89
9.6 BioAPI_EnumBSPs 90
9.7 BioAPI_BSPLoad 93
9.8 BioAPI_BSPUnload 94
9.9 BioAPI_BSPAttach 95
9.10 BioAPI_BSPDetach 98
9.11 BioAPI_QueryUnits 99
9.12 BioAPI_EnumBFPs 101
9.13 BioAPI_QueryBFPs 104
9.14 BioAPI_ControlUnit 106
9.15 BioAPI_FreeBIRHandle 108
9.16 BioAPI_GetBIRFromHandle 109
9.17 BioAPI_GetHeaderFromHandle 111
9.18 BioAPI_EnableEvents 113
9.19 BioAPI_SetGUICallbacks 114
9.20 BioAPI_Capture 116
9.21 BioAPI_CreateTemplate 118
9.22 BioAPI_Process 122
9.23 BioAPI_ProcessWithAuxBIR 124
9.24 BioAPI_VerifyMatch 126
9.25 BioAPI_IdentifyMatch 130
9.26 BioAPI_Enroll 133
9.27 BioAPI_Verify 137
9.28 BioAPI_Identify 141
9.29 BioAPI_Import 145
9.30 BioAPI_PresetIdentifyPopulation 147
9.31 BioAPI_DbOpen 149
9.32 BioAPI_DbClose 151
9.33 BioAPI_DbCreate 152
9.34 BioAPI_DbDelete 155
9.35 BioAPI_DbSetMarker 156
9.36 BioAPI_DbFreeMarker 157
9.37 BioAPI_DbStoreBIR 159
9.38 BioAPI_DbGetBIR 161
9.39 BioAPI_DbGetNextBIR 163
9.40 BioAPI_DbDeleteBIR 165
9.41 BioAPI_SetPowerMode 166
9.42 BioAPI_SetIndicatorStatus 168
9.43 BioAPI_GetIndicatorStatus 169
9.44 BioAPI_CalibrateSensor 171
9.45 BioAPI_Cancel 172
9.46 BioAPI_Free 173
9.47 BioAPI_Util_InstallBSP 174
9.48 BioAPI_Util_InstallBFP 176
9.49 BioSPI_BSPLoad 178
9.50 BioSPI_BSPUnload 180
9.51 BioSPI_BSPAttach 181

9.52 BioSPI_BSPDetach ... 183
9.53 BioSPI_QueryUnits ... 184
9.54 BioSPI_QueryBFPs ... 186
9.55 BioSPI_ControlUnit ... 187
9.56 BioSPI_FreeBIRHandle ... 189
9.57 BioSPI_GetBIRFromHandle ... 190
9.58 BioSPI_GetHeaderFromHandle ... 191
9.59 BioSPI_EnableEvents ... 192
9.60 BioSPI_SetGUICallbacks ... 193
9.61 BioSPI_Capture ... 194
9.62 BioSPI_CreateTemplate ... 195
9.63 BioSPI_Process ... 197
9.64 BioSPI_ProcessWithAuxBIR ... 198
9.65 BioSPI_VerifyMatch ... 200
9.66 BioSPI_IdentifyMatch ... 202
9.67 BioSPI_Enroll ... 204
9.68 BioSPI_Verify ... 206
9.69 BioSPI_Identify ... 209
9.70 BioSPI_Import ... 211
9.71 BioSPI_PresetIdentifyPopulation ... 212
9.72 BioSPI_DbOpen ... 213
9.73 BioSPI_DbClose ... 215
9.74 BioSPI_DbCreate ... 216
9.75 BioSPI_DbDelete ... 217
9.76 BioSPI_DbSetMarker ... 218
9.77 BioSPI_DbFreeMarker ... 219
9.78 BioSPI_DbStoreBIR ... 220
9.79 BioSPI_DbGetBIR ... 221
9.80 BioSPI_DbGetNextBIR ... 222
9.81 BioSPI_DbDeleteBIR ... 224
9.82 BioSPI_SetPowerMode ... 225
9.83 BioSPI_SetIndicatorStatus ... 226
9.84 BioSPI_GetIndicatorStatus ... 227
9.85 BioSPI_CalibrateSensor ... 228
9.86 BioSPI_Cancel ... 228
9.87 BioSPI_Free ... 229
9.88 BioAPI_EventHandler ... 230
9.89 BioAPI_GUI_STATE_CALLBACK ... 231
9.90 BioAPI_GUI_STREAMING_CALLBACK ... 234
9.91 BioSPI_EventHandler ... 235
9.92 BioSPI_GUI_STATE_CALLBACK ... 237
9.93 BioSPI_GUI_STREAMING_CALLBACK ... 238
9.94 BioSPI_BFP_ENUMERATION_HANDLER ... 240
9.95 BioSPI_MEMORY_FREE_HANDLER ... 242

10 Built-in variables ... 243
10.1 Variables whose value never changes ... 243
10.2 Variables whose value may change ... 251

11 Test log ... 251

12 Test report ... 254

13 BioAPI conformance test suite ... 255
13.1 General concepts ... 255
13.2 BioAPI conformance test suite structure ... 255

Annex A (normative) XML schema of the assertion language ... 257

Annex B (normative) ASN.1 schema of the assertion language ... 261

Annex C (normative) XML schema for the test log ... 265

Annex D (informative) Test method implementation guideline ... 267

Annex E (informative) **BioAPI conformity assessment program** **275**
Annex F (informative) **XML diagrams of the assertion language** **281**
Bibliography **303**

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 24709-1 was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information technology*, Subcommittee SC 37, *Biometrics*.

ISO/IEC 24709 consists of the following parts, under the general title *Information technology — Conformance testing for the biometric application programming interface (BioAPI)*:

— *Part 1: Methods and procedures*

— *Part 2: Test assertions for biometric service providers*

The following parts are under preparation:

— *Part 3: Test assertions for BioAPI frameworks*

— *Part 4: Test assertions for biometric applications*

Introduction

This part of ISO/IEC 24709 defines a conformance testing methodology for ISO/IEC 19784-1. It specifies three conformance testing models that enable conformance testing of each of the following BioAPI components: an application, a framework and a BSP. It also specifies an assertion language that is used for the definition of test assertions. Actual test assertions for each of the BioAPI components are defined in subsequent parts of ISO/IEC 24709.

This part of ISO/IEC 24709 also contains informative guidelines regarding general concepts related to establishing and administering a BioAPI conformance assessment and certification program. These informative guidelines identify the types of activities, responsibilities, services and documentation recommended for conducting conformity assessment and certification of BioAPI-conformant implementations. Further, this part of ISO/IEC 24709 provides informative guidelines for establishing a complete conformity assessment methodology for BioAPI specification.

Clause 6 describes the general test method and conformance testing models for BioAPI.

Clause 7 defines the assertion language, based on XML, used for definition of conformance test assertion.

Clause 8 defines the elements of the assertion language.

Clause 9 specifies the use of the standard BioAPI interface functions of BioAPI in conformance testing.

Clause 10 defines the built-in variables of the assertion language.

Clause 11 defines the test log using XML syntax.

Clause 12 defines the test report using XML syntax.

Clause 13 describes the general concept and structure of a BioAPI conformance test suite.

Annex A is normative, and defines the XML schema of the assertion language.

Annex B is normative, and defines the ASN.1 schema of the assertion language.

Annex C is normative, and defines the XML schema for the test log.

Annex D is informative, and describes a primer of a BioAPI test method implementation, including elements of the conformance test process and description of the test categories.

Annex E is informative, and describes a general framework for the overall BioAPI Conformity Assessment Process.

Annex F is informative, and provides the relationship diagrams for the assertion language.

The Bibliography references a number of standards organizatons, including ISO, IEC, NIST and IEEE, and other organizations that have published a number of documents and white papers related to conformity assessments in general and conformance testing in particular.1)

1) Rather than make normative references to these documents, this part of the ISO/IEC 24709 incorporates appropriate excerpts of their text, in some cases paraphrasing the text or adapting the provisions to the specific circumstances. Therefore, these documents are listed in the Bibliography or are referenced explicitly in the body text, as appropriate.

Information technology — Conformance testing for the biometric application programming interface (BioAPI) —

Part 1: Methods and procedures

1 Scope

1.1 This part of ISO/IEC 24709 specifies the concepts, framework, test methods and criteria required to test conformity of biometric products claiming conformance to BioAPI (see ISO/IEC 19784-1). Guidelines for specifying BioAPI conformance test suites, writing test assertions and defining procedures to be followed during the conformance testing are provided.

1.2 This part of ISO/IEC 24709 is concerned with conformance testing of biometric products claiming conformance to BioAPI (see ISO/IEC 19784-1). It is not concerned with testing other characteristics of biometric products or other types of testing of biometric products (i.e. acceptance, performance, robustness, security, etc.). Testing by means of test methods which are specific to particular biometric products are not the subject of ISO/IEC 24709.

1.3 This part of ISO/IEC 24709 is applicable to the development and use of conformance test method specifications, BioAPI conformance test suites and conformance testing programs for BioAPI-conformant products. It is intended primarily for use by testing organizations, but may be applied by developers and users of test assertions and test method implementations.

2 Conformance

2.1 A BioAPI conformance test suite conforming to this part of ISO/IEC 24709 shall support one or more conformance testing models (see 6.2) and shall be able to execute any valid test assertion for the testing model(s) that it supports, and that are written in the assertion language specified in Clauses 7 through 10.

NOTE There is no restriction on the form or structure of a BioAPI conformance test suite, in terms of the number of software components, the tasks performed by each software component, or the content and form of the information exchanged between software components.

2.2 A BioAPI conformance test suite shall be able to verify the syntactic correctness of any package (see 7.1.6) containing assertions or activities (or both) for any conformance testing model, including the testing models that the implementation does not support (if any).

2.3 For each supported conformance testing model, a BioAPI conformance test suite shall be able to perform the actions (specific to a computing platform) necessary to interact with an implementation under test, making function calls to the standard BioAPI interface functions exposed by the implementation under test and receiving function calls from it.

NOTE 1 In the conformance testing model for BioAPI applications, it is not required that the BioAPI conformance test suite be able to start or stop the execution of the implementation under test, but needs a mechanism to detect the starting or ending of the application under test.

NOTE 2 It is not required that a BioAPI conformance test suite be able to test all implementations of the base standard that claim conformance to the base standard. This includes, but is not limited to, the case when the implementation of the base standard was created for a computing platform different from the one for which the BioAPI conformance test suite was created, and the case where the implementation of the base standard depends on a hardware device that is not available on the computing system where the test is to be run.

2.4 A BioAPI conformance test suite shall produce a test log (see Clause 11) and test report (see Clause 12) for each implementation tested.

2.5 If a BioAPI conformance test suite is unable to perform the test of an implementation of the base standard, this shall be recorded in the test report in these terms rather than as non-conformance of the implementation under test.

2.6 A BioAPI conformance test suite shall provide a means for a user to enter all the data necessary as input to a test.

NOTE This includes the identification of the assertion to be processed (package name and assertion name), the list of all the input parameters of the assertion, and all the other information that is to be included in a test report (see Clause 12).

3 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 19784-1, *Information technology — Biometric application programming interface — Part 1: BioAPI specification*

4 Terms and definitions

For the purposes of this document, the terms and definitions given in ISO/IEC 19784-1 and the following apply.

4.1
abstract test engine
conceptual machine capable of performing conformance tests on an instance of a standard BioAPI component

4.2
base standard
standard for which a test method specification is written and/or a test method implementation is developed

4.3
BioAPI conformance test suite
test software used to ascertain conformance to a specification or a standard that is in conformance with ISO/IEC 19784-1

4.4
BioAPI conformity statement
statement that describes conformity of an Implementation Under Test against relevant BioAPI requirements

4.5
certification
acknowledgement that a validation has been completed and the criteria established by the certifying organization have been met

4.6
conformance
fulfillment by a product, process or service of all relevant specified conformance requirements [ISO/IEC 13210:1999]

4.7
conformance requirement
requirement stated in a base standard that identifies a specific requirement in a finite, measurable and unambiguous manner

NOTE 1 A conformance requirement by itself or in conjunction with other conformance requirements corresponds to an assertion.

NOTE 2 Adapted from ISO/IEC 13210:1999.

4.8
conformity assessment
any activity concerned with determining directly or indirectly that relevant requirements are fulfilled

4.9
implementation under test
software and hardware, located on one or more system, which implement the standard(s) being tested

4.10
standard BioAPI component
BioAPI application, BioAPI framework or biometric service provider, as specified in ISO/IEC 19784-1

NOTE 1 See 6.1.3.8.

NOTE 2 Although a biometric function provider is also a BioAPI component according to ISO/IEC 19784-1, it is not included in this definition because ISO/IEC 19784-1 does not specify any biometric function provider interfaces.

4.11
standard BioAPI interface
any one of the interfaces specified in ISO/IEC 19784-1, which one standard BioAPI component exposes to another standard BioAPI component

NOTE See 6.1.3.8.

4.12
test assertion
assertion
specification for testing a conformance requirement in an Implementation Under Test in the form of software or procedural methods that generate the test results (also named test outcomes or test verdicts) used for assessment of the conformance requirement

4.13
test case
specification of the actions required to achieve a specific test purpose or combination of test purposes

NOTE Adapted from the definition of “abstract test case” in ISO/IEC 9646-1:1994.

4.14
test method implementation
software, procedures or other means used to measure conformance

4.15
test method specification
document that expresses the required functionality and behavior of a base standard as assertions and provides the complete set of conforming test result codes

[ISO/IEC 13210:1999]

4.16
test purpose
prose description of a narrowly defined objective of testing, focusing on a single conformance requirement, as specified in the appropriate product specification

NOTE Adapted from ISO/IEC 13210:1999 and ISO/IEC 9646-1:1994.

4.17
test report
document that presents test results and other information relevant to the execution of the test methods against an Implementation Under Test

4.18
test result code
test verdict
value that describes the result of a test

NOTE Adapted from ISO/IEC 13210:1999.

4.19
validation
process of testing software for conformance to a specific specification

5 Abbreviations

For the purposes of this document, the following abbreviations apply.

API	Application Programming Interface
BCS	BioAPI Conformity Statement
BIR	Biometric Information Record
BSP	Biometric Service Provider
CBEFF	Common Biometric Exchange Formats Framework
CTS	BioAPI Conformance Test Suite
IUT	Implementation Under Test
SPI	Service Provider Interface
UUID	Universally Unique Identifier

6 Conformance testing methodology

6.1 General

6.1.1 Implementation under test

6.1.1.1 The Implementation Under Test (IUT) is the object that is being tested for conformity. For BioAPI specifications it is the software that has 'implemented' the specification. The software and supporting hardware constitute the IUT and shall be listed in both the test report and certificate of conformity.

6.1.1.2 Biometric products claiming to conform to the BioAPI specification are expected to conform to all the applicable conformity requirements in the Conformity clause of the BioAPI specification (Clause 5 of the specification). These requirements can be:

a) Mandatory requirements: these are to be observed in all cases;

b) Conditional requirements: these are to be observed if the conditions set out in the specification apply;

c) Optional requirements: these can be selected to suit the implementation, and are to be observed if selected.

6.1.1.3 To evaluate the conformity of a biometric product, it is necessary to have a statement of the capabilities that have been implemented in conformance with the BioAPI specification, so that the implementation can be tested for conformity against relevant requirements, and against those requirements only. Such a statement is called a BioAPI Conformity Statement (BCS), and shall be prepared by the IUT supplier prior to the beginning of the Conformance Testing. At a minimum, the BCS shall contain an itemized list of all mandatory, optional, and conditional conformity requirements of the BioAPI specification included in the IUT.

6.1.2 Test method

6.1.2.1 For conformance testing to be meaningful, all BioAPI implementations must be tested in the same manner. Conformance testing reflects the essence of technical requirements of BioAPI specifications and measures whether a Biometric product faithfully implements the specification.

6.1.2.2 For the purpose of this standard, conformance testing is "black box" testing of the functionality of a BioAPI implementation. Neither the internal structure nor source code of a candidate implementation is examined.

6.1.2.3 Considering the complexity of the BioAPI specification, and many possible ways of implementing BioAPI-conformant products, the feasible strategy is to use falsification testing methodology. This strategy as implemented in this Part of ISO/IEC 24709 includes the following steps:

a) Analyze the BioAPI specification, and develop documented test cases in form of test assertions. Test cases may be further grouped to form test scenarios. The test assertions are documented in other parts of ISO/IEC 24709.

b) The test assertions may be realized in the form of executable test scripts, which, in combination with applicable data files, will constitute a BioAPI conformance test suite (CTS)

c) An IUT will be subjected to various combinations of legal and illegal inputs, and compare the resulting output to a set of corresponding "expected results."

d) Test results will be evaluated using pass/fail criteria.

6.1.2.4 Falsification testing can only demonstrate non-conformity, i.e., if errors are found, non-conformance of the IUT shall be proven, but the absence of errors does not necessarily imply the converse. This test method is intended to provide a reasonable level of confidence and practical assurance that the IUT conforms to the standard. Use of this test method will not guarantee conformity of an implementation to the standard; that normally would require exhaustive testing, which is impractical for both technical and economic reasons.

6.1.2.5 A test method implementation shall document that it conforms to this Part of ISO/IEC 24709 and shall document any other test method specifications to which it claims to conform. Each test method implementation shall include the following:

a) BioAPI Conformance Test Suite (CTS), including documentation of the test suite, describing test categories, test objectives for each individual test, instructions on how to execute the test suite, and the expected results of executing the individual tests. The CTS shall be capable of executing the test script sets, capturing the returned results, evaluating the results, and reporting them in a human-readable form.

b) Documented test cases, which will sufficiently assure conformity to the standard. The test cases shall be formally represented in the form of test assertions using the assertion language, which can be submitted to the BioAPI Conformance test suite.

c) Conformance Testing Procedure, which shall identify and define all the activities necessary to prepare for Conformance Testing, perform the conformance testing, and report the test results. The procedure shall be detailed enough so that testing of a given IUT can be repeated with no significant change in test results. The procedure shall identify the administrative as well as testing processes.

6.1.2.6 A test method implementation shall use the required assertion definitions, types, syntax, and constructs specified in ISO/IEC 24709 as applicable. It shall use test result codes specified in this Part of ISO/IEC 24709 for test results defined by this Part of ISO/IEC 24709.

6.1.2.7 The conformance testing process is the complete process of accomplishing all conformance testing activities necessary to assess the conformity of an IUT to the BioAPI specification standard. The conformance testing process involves three phases:

a) Preparation for testing, which includes analysis of the BCS, preparation of the Conformance Test Plan, selection and configuration of the CTS, and preparation of the IUT and corresponding test environment (means of testing).

b) Test execution, which includes execution of the CTS and recording the observed test results in conformity test log(s). The results of conformance testing shall apply only to the IUT and test environment for which the tests are run.

c) Test Report production, which includes recording of all events occurred during the execution of each test case, including all test outcomes and test verdicts, as well as description of the test environment (operating system, hardware configuration, etc.)

6.1.2.8 In order to achieve the objective of credible conformance testing, the result of executing a test case on an IUT should be the same whenever it is performed. While it may not be possible to execute a complete BioAPI Conformance test suite and observe test outcomes that are identical to those obtained on another occasion, every effort should be made to ensure repeatability of test results and to minimize the possibility that a test case produces different outcomes on different occasions.

6.1.2.9 It may be necessary to review the observed test outcomes in order to make sure that all procedures have been correctly followed. It is essential that all inputs, outputs, and other test events, as well as test environment and IUT configuration, be logged for each test case being executed, with sufficient information to produce a conformity log for each test execution, for future reference.

6.1.3 Standard BioAPI components and standard BioAPI interfaces

6.1.3.1 This Part of ISO/IEC 24709 specifies a methodology for assessing conformance of implementations of ISO/IEC 19784-1 to that International Standard. Three types of implementations of ISO/IEC 19784-1 are distinguished (see Figure 1):

a) BioAPI application;

b) BioAPI framework;

c) BioAPI biometric service provider (BSP).

6.1.3.2 A BioAPI application is a software component (or set of software components) that uses the BioAPI interface specified in ISO/IEC 19784-1, making one or more function calls to the BioAPI interface in the course of its execution.

6.1.3.3 A BioAPI framework is a software component (or set of software components) that implements the BioAPI interface specified in ISO/IEC 19784-1 and realizes its prescribed behavior, making one or more function calls to the BioSPI interface specified in ISO/IEC 19784-1 in the course of its execution.

6.1.3.4 A BioAPI biometric service provider is a software component (or set of software components) that implements the BioSPI interface specified in ISO/IEC 19784-1 and realizes its prescribed behavior.

6.1.3.5 A BioAPI biometric function provider (BFP) is another component of the BioAPI architecture, but its interface is not specified in ISO/IEC 19784-1. Therefore this Part of ISO/IEC 24709 contains no provisions relative to that component.

6.1.3.6 Besides the two main interfaces BioAPI and BioSPI, two other interfaces are specified in ISO/IEC 19784-1, as indicated in the five following sub-clauses.

6.1.3.6.1 BioAPI frameworks implement an auxiliary interface that supports the reception of the following information from BSPs:

a) notifications of events related to BioAPI units;

b) streaming data (grayscale bitmaps) sent during an operation performed by a BSP or by a BFP (on behalf of a BSP);

c) GUI state information sent during an operation performed by a BSP or by a BFP (on behalf of a BSP).

6.1.3.6.2 In this Part of ISO/IEC 24709, the interface mentioned in the previous sub-clause, implemented by BioAPI frameworks, is called the "framework callback interface."

6.1.3.6.3 BioAPI applications may implement an auxiliary interface that supports the reception of some or all of the following information from the BioAPI framework:

a) notifications of events related to BioAPI units, originating in a BSP or in a BFP and relayed by the framework to the application;

b) streaming data (grayscale bitmaps) sent during an operation performed by a BSP or by a BFP (on behalf of a BSP) and relayed by the framework to the application;

c) GUI state information sent during an operation performed by a BSP or by a BFP (on behalf of a BSP) and relayed by the framework to the application.

The BioAPI framework relays to the application(s) the information it receives from the BSPs, depending on which functions are supported by each application, and depending on whether the application is currently using the BSP that sends the information to the BioAPI framework.

6.1.3.6.4 In this Part of ISO/IEC 24709, the interface mentioned in the previous sub-clause, optionally implemented by BioAPI applications, is called the "application callback interface".

6.1.3.6.5 In addition, BioAPI frameworks include functions that support installation and deinstallation of BSPs, and installation and deinstallation of BFPs. In this Part of ISO/IEC 24709, those functions are included in the BioAPI interface.

6.1.3.7 BioAPI BSPs may implement an auxiliary interface that supports the reception of information from BFPs, but this interface is not specified in ISO/IEC 19784-1. Therefore this Part of ISO/IEC 24709 contains no provisions relative to that interface.

6.1.3.8 In this Part of ISO/IEC 24709, the three types of implementations of the BioAPI standard listed above are called the standard BioAPI components. The BioAPI interface, the BioSPI interface, the application callback interface, and the framework callback interface mentioned above are called the standard BioAPI interfaces. For the purposes of this Part of ISO/IEC 24709, each standard BioAPI component interacts with other standard BioAPI components by making function calls to their standard BioAPI interfaces (and in no other way).

6.1.3.9 This Part of ISO/IEC 24709 specifies a BioAPI conformance testing methodology in terms of the three standard BioAPI components and of the four standard BioAPI interfaces.

6.1.4 Physical architectures

6.1.4.1 ISO/IEC 19784-1 does not specify the details of loading in memory any one of the standard BioAPI components, nor does it specify a physical association between standard BioAPI components and nodes, nor does it limit the number of instances of each standard BioAPI component within a node. All this information is therefore platform-specific or implementation-specific.

6.1.4.2 Typically, BioAPI BSPs are implemented as dynamic libraries on the platforms that support this feature.

6.1.4.3 In one typical physical BioAPI architecture, the BioAPI framework is a dynamic library as well, and BioAPI applications are executable programs that load the BioAPI framework dynamic library in memory. In this architecture, there can only be one BioAPI application using an instance of the BioAPI framework at any given time. If multiple BioAPI applications are running concurrently in a node, each of them owns a separate instance of the BioAPI framework and a separate instance of each BSP that is loaded by more than one of them.

6.1.4.4 In another typical physical BioAPI architecture, the BioAPI framework is an independent executable program (for example, an operating system service) and any number of BioAPI applications may be launched and terminated at any time during the execution of the BioAPI framework, all of them using the same instance of the framework and the same instance of each BSP that is loaded by more than one of them simultaneously.

6.1.4.5 Given the abstractness of the BioAPI architecture specified in ISO/IEC 19784-1, the conformance testing methodology specified in this Part of ISO/IEC 24709 does not depend on any particular physical BioAPI architecture. The conformance testing methodology treats the standard BioAPI components as abstract components implementing the interfaces and the behavior specified in ISO/IEC 19784-1, with no assumptions being made on their physical realization.

6.1.4.6 However, specific implementations of the conformance testing methodology (BioAPI conformance test suites) run on specific platforms and are designed for particular physical BioAPI architectures. Therefore, each BioAPI conformance test suite may be designed for testing only a subset of the possible implementations of the standard BioAPI components. In Annex E of this this Part of ISO/IEC 24709, recommendations are provided in order to increase the level of consistency between different implementations of the conformance testing methodology created for different platforms or for different physical BioAPI architectures on the same platform.

6.2 Conformance testing models

6.2.1 The conformance testing methodology specified in this Part of ISO/IEC 24709 addresses each of the standard BioAPI components separately. Three conformance testing models are defined within the methodology, one for the testing of each standard BioAPI component.

6.2.2 The three conformance testing models are independent from one another, although they have many properties in common.

6.2.3 Implementations of the conformance testing methodology reflect the separation between these three testing models, resulting in testing procedures that are specialized for testing the conformance of only one of the following at a time:

a) a given implementation of the BioAPI application standard BioAPI component;

b) a given implementation of the BioAPI framework standard BioAPI component;

c) a given implementation of the BioAPI BSP standard BioAPI component.

6.2.4 Each testing model is based on a variation to the basic BioAPI architecture.

6.2.5 For the purposes of this clause, the basic BioAPI architecture is described as comprising a normal BioAPI application, a normal BioAPI framework, and one or more normal BioAPI BSPs. Each testing model makes additions and replacements to the basic architecture as specified in the three following sub-clauses.

6.2.5.1 In the conformance testing model for BioAPI applications, a special testing component (called the "application-testing framework") shall be inserted between the application under test and the normal framework (see Figure 2). This testing component shall implement the standard BioAPI interface on one side and the application callback interface on the other side. As a result, it shall appear to the application under test as a framework, and to a framework as an application. This testing component shall have the ability to relay application calls to the normal framework and framework calls to the application under test, and the ability to observe, analyze, and log the flow of incoming calls and to generate extra calls for testing purposes.

6.2.5.2 In the conformance testing model for BioAPI frameworks, a special testing component (called the "framework-testing application") shall replace the normal application, and another special testing component (called the "framework-testing BSP") shall replace the normal BSP (see Figure 3). Between these two testing components shall lie the framework under test. The framework-testing application shall implement the application callback interface and the framework-testing BSP shall implement the BioSPI interface. Therefore the framework under test cannot distinguish between them and the corresponding normal BioAPI components. In addition, each testing component shall have a special testing interface enabling them to interact with each other for the purpose of performing the tests.

6.2.5.3 In the conformance testing model for BioAPI BSPs, a special testing component (called the "BSP-testing application") shall replace the normal application and the normal framework (see Figure 4). This testing component shall act both as a BioAPI application and as a BioAPI framework, and shall implement the framework callback interface. As a result, it shall appear to the BSP under test as a framework. The testing component shall be able to make calls to the BioSPI interface of the BSP under test.

6.2.6 When a BioAPI implementation that comprises two or more standard BioAPI components is submitted to conformance testing, the procedures for each component shall be followed separately for each component, producing separate reports (see clause 12).

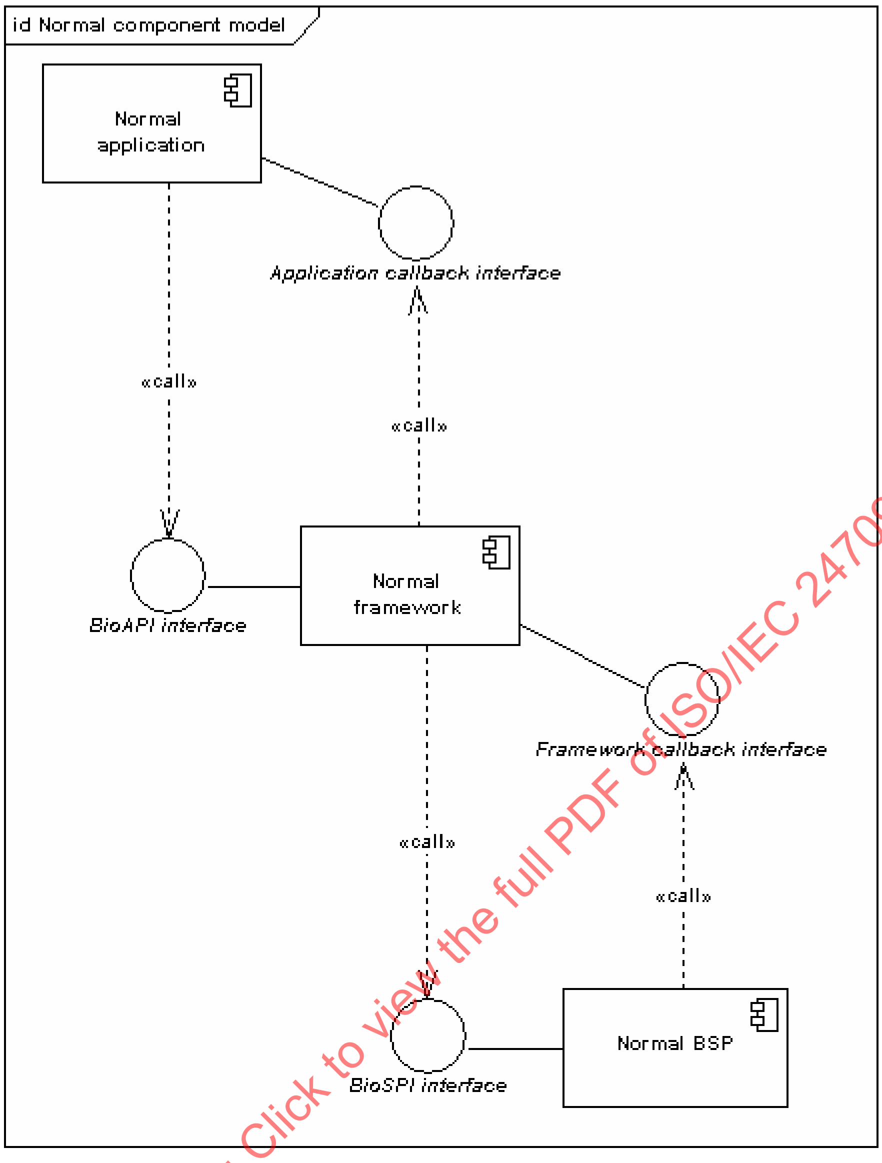

Figure 1 — Normal BioAPI component model

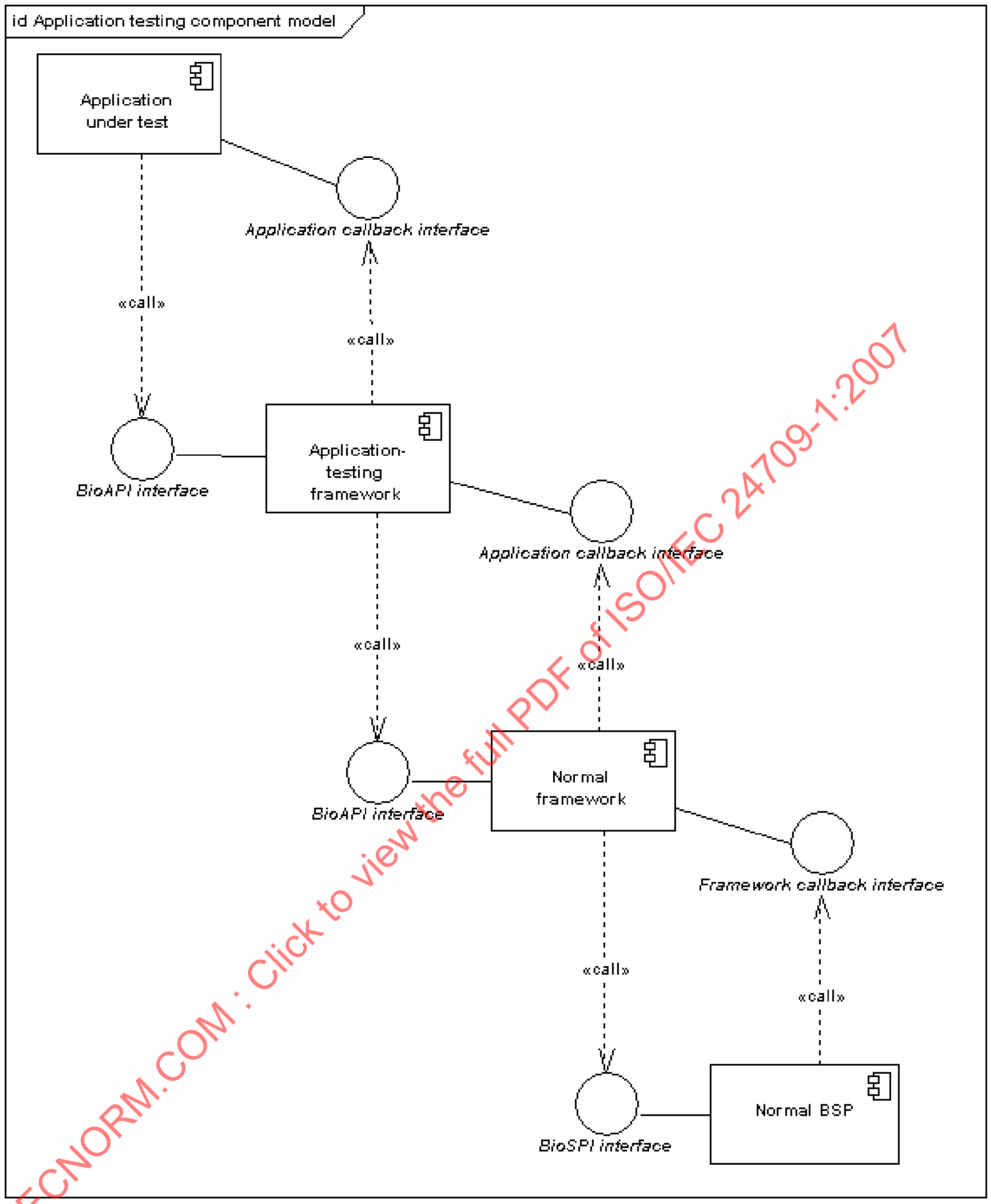
id Application testing component model
Application under test
Application callback interface
«call»
«call»
BioAPI interface
Application-testing framework
Application callback interface
«call»
«call»
BioAPI interface
Normal framework
Framework callback interface
«call»
«call»
BioSPI interface
Normal BSP

Figure 2 — Conformance testing model for BioAPI applications

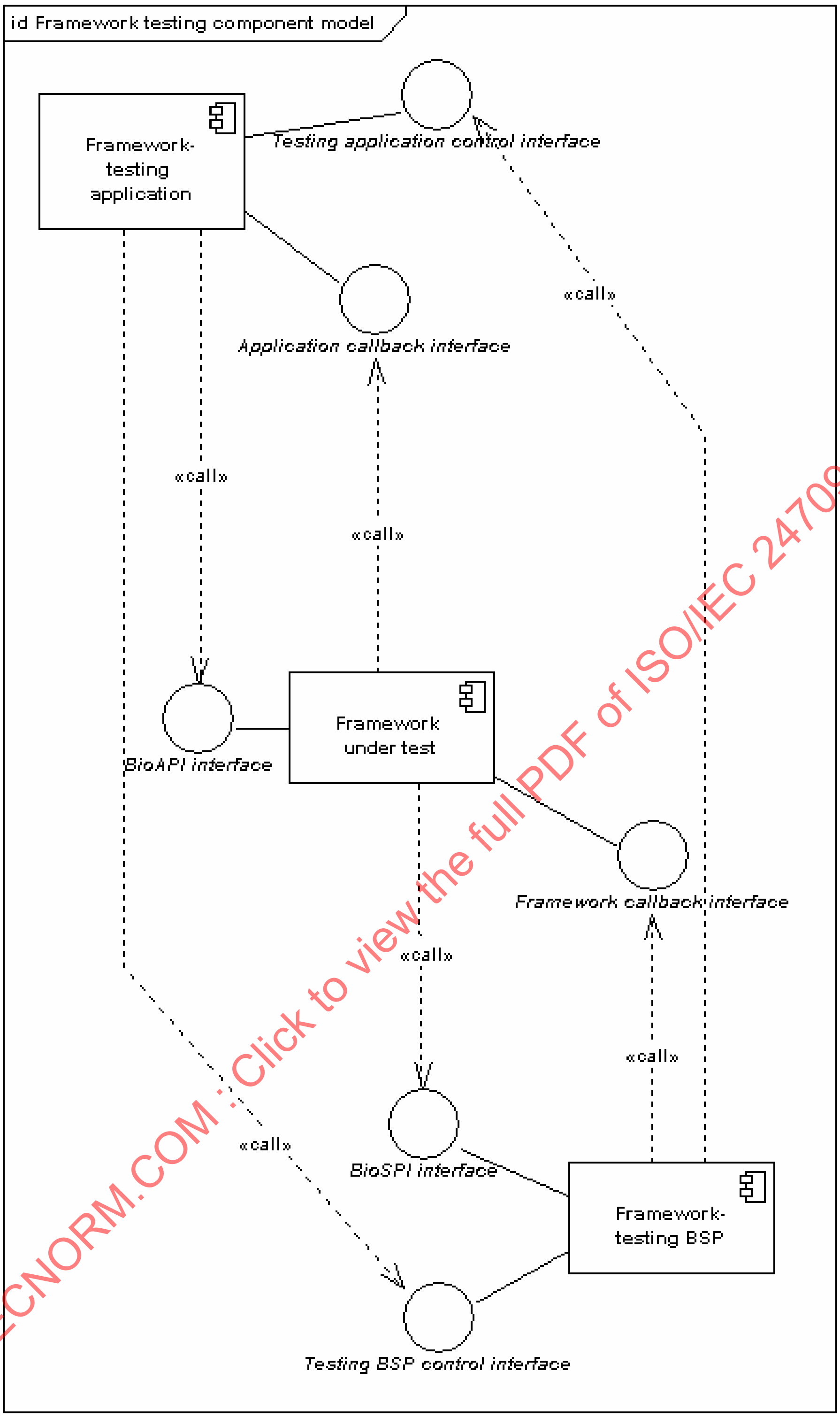
id Framework testing component model
Framework-testing application
Testing application control interface
Application callback interface
«call»
«call»
«call»
BioAPI interface
Framework under test
Framework callback interface
«call»
«call»
BioSPI interface
Framework-testing BSP
«call»
Testing BSP control interface

Figure 3 — Conformance testing model for BioAPI frameworks

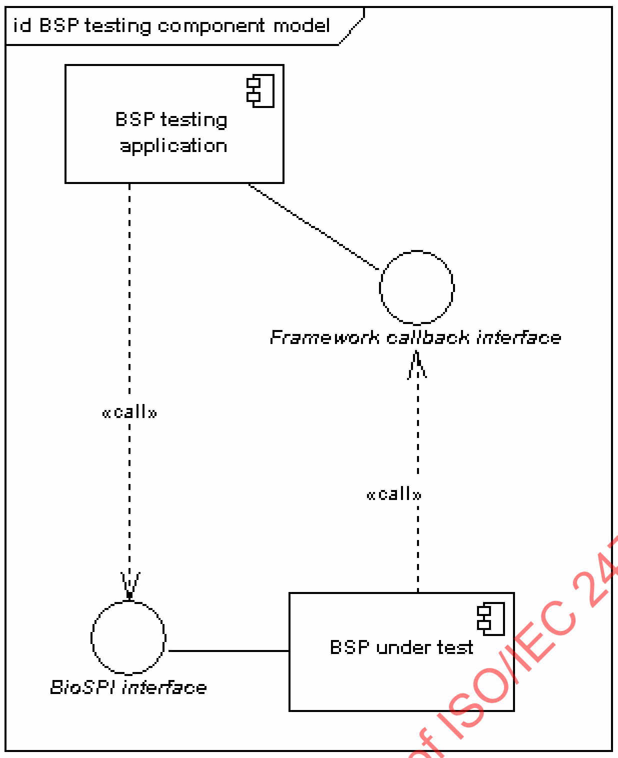

Figure 4 — Conformance testing model for BioAPI BSPs

6.3 Abstract test engine

6.3.1 The semantics of the assertion language is specified in terms of an abstract test engine.

6.3.2 The abstract test engine is a conceptual machine capable of performing conformance tests on an instance of a standard BioAPI component (application, framework, or BSP). The abstract test engine shall be able to do so by processing a formal assertion written in the assertion language specified in this Part of ISO/IEC 24709 (see clause 7).

6.3.3 The abstract test engine shall have the ability to operate according to any of the three conformance testing models (see 6.2) depending on the type of the standard BioAPI component under test. In each testing model, the abstract test engine shall be associated with the special testing components in the model, as follows:

a) in the conformance testing model for BioAPI applications (see 6.2.5.1), the abstract test engine shall be associated with the application-testing framework;

b) in the conformance testing model for BioAPI frameworks (see 6.2.5.2), the abstract test engine shall be associated both with the framework-testing application and with the framework-testing BSP at the same time;

c) in the conformance testing model for BioAPI BSPs (see 6.2.5.3), the abstract test engine shall be associated with the BSP-testing application.

6.3.4 During the processing of a test assertion, the special testing component(s) associated with the abstract test engine shall be able to make calls to the standard BioAPI interfaces of the component under test and of other components present in the testing model. The testing component(s) shall also be able to process incoming calls made by the other components to the standard BioAPI interfaces implemented by the testing components themselves.

6.3.5 The type of outgoing calls being made and of incoming calls being processed by the testing component(s) shall depend on the type of standard BioAPI component under test (and thus on the conformance testing model being used), while the actual calls being made and the way of processing the incoming calls shall depend on the content of the assertion being processed.

6.3.6 In the conformance testing model that contains two special testing components (the testing model for BioAPI frameworks), the testing components associated with the abstract test engine shall communicate with each other by using the special testing interfaces, so that the abstract test engine controls the behavior of both components at the same time. The communication between the two testing components is not standardized, and the special testing interfaces are not directly represented in the language.

6.3.7 The structure and operation of the abstract test engine shall not depend on platforms, physical BioAPI architecture, or the internal architecture of the BioAPI components. From the point of view of the abstract test engine, each component shall be a black box whose behavior depends solely on:

a) the calls being made to the entry points of its standard BioAPI interfaces; and

b) the passing of time or interaction with other actors (invisible to the abstract test engine) as they can cause well-defined activities to take place inside the component at undefined instants.

6.3.8 The activities described in 6.3.7 b) may be indirectly observed by the abstract test engine as they may affect:

a) one or more subsequent calls that the component will make to other components; or

b) the component's responses to one or more subsequent calls that other components will make to it.

6.3.9 BioAPI Conformance test suites shall include a concrete implementation of the abstract test engine. The structure of the abstract test engine does not have to be mirrored in such implementations, but its specified behavior shall be reflected in the implementation, so that the semantics of the assertion language is preserved.

6.3.10 Concrete activities such as loading and running an executable program, locating and loading a dynamic library, and so on, fall below the level of abstraction of the conformance testing models and of the abstract test engine. Therefore they are not formally described in this specification.

7 General properties of the assertion language

7.1 General

7.1.1 Clauses 7 to 10 of this Part of ISO/IEC 24709 specify a language whose purpose is to express assertions to be used in BioAPI conformance testing. The assertion language is an integral part of the BioAPI conformance testing methodology.

7.1.2 Other parts of ISO/IEC 24709 contain a set of assertions written in the assertion language. Implementations of the BioAPI conformance testing methodology (BioAPI conformance test suites) that claim conformance to ISO/IEC 24709 need to use (or implement) all the assertions provided in these parts and only those assertions.

7.1.3 The above does not preclude particular implementations of the conformance testing methodology from using a different set of assertions (be they derived or not from the assertions in other parts of ISO/IEC 24709), while still claiming conformance to this Part of ISO/IEC 24709. However, such implementations cannot also claim conformance to other parts of ISO/IEC 24709.

7.1.4 The assertion language has a syntax based on W3C XML 1.0. In the rest of this clause, the terms "element" and "attribute" are used with the meaning of "XML element" and "XML attribute", respectively.

7.1.5 Assertions are represented as `<assertion>` elements (see 8.2). Each assertion has a number of properties represented as attributes and as child elements of the `<assertion>` element. Among the properties of an assertion are its name (`name` attribute), its conformance testing model (`model` attribute), and its primary activity reference (`<invoke>` child element).

EXAMPLE (non-normative)

```
<assertion name="CreateTemplate1" model="BSPTesting">
   <description>
      Test the BioSPI_CreateTemplate function of a BSP.
      The UUID and version of the BSP must be provided
      as input to the test.
   </description>
   <input name="_uuid"/>
   <input name="_version"/>
   <invoke activity="CreateTemplate"
      package="7346D660-1583-13D0-A3A5-00C0FFD756E3">
      <input name="BSPUuid" var="_uuid"/>
      <input name="BSPVersion" var="_version"/>
      <input name="deviceIDOrNull" value="0"/>
      <input name="inserttimeouttime" value="15000"/>
      <input name="sourcepresenttimeouttime" value="10000"/>
      <input name="capturetimeouttime" value="20000"/>
   </invoke>
   <bind activity="EventHandler" function="BioAPI_EventHandler"/>
</assertion>
```

7.1.6 Assertions are grouped into packages. Each package, represented as a **`<package>`** element that is always the root element of an XML document, has a name (**`name`** attribute) and other identification properties. A package contains zero or more assertions (**`<assertion>`** elements) followed by zero or more activities (**`<activity>`** elements). Empty packages are permitted, as well as packages containing only assertions, or only activities, or both assertions and activities (where all the assertions precede the first activity in the package). Neither the **`<assertion>`** element nor the **`<activity>`** element can be the root element of an XML document.

EXAMPLE (non-normative)

```
<?xml version='1.0' encoding="utf-8"?>
<package name="73668660-1583-1AD0-A3A5-09C0FF4756E3">
   <author>
      ISO/IEC SC37
   </author>
   <description>
      Abcde abcde abcde
   </description>

   <assertion name="Capture2" model="BSPTesting">
      <description>
         Test the BioSPI_Capture function of a BSP.
      </description>
      <invoke activity="Capture"
         package="7346D660-1583-13D0-A3A5-00C0FFD756E3">
         <input name="BSPUuid" value=""/>
         <input name="BSPVersion" value="0"/>
         <input name="deviceIDOrNull" value="0"/>
         <input name="inserttimeouttime" value="15000"/>
         <input name="sourcepresenttimeouttime" value="10000"/>
         <input name="capturetimeouttime" value="20000"/>
      </invoke>
      <bind activity="EventHandler" function="BioAPI_EventHandler"/>
   </assertion>

   <assertion name="Capture5" model="BSPTesting">
      <description>
         Test the BioSPI_Capture function of a BSP
         with abcde abcde abcde.
      </description>
      <invoke activity="Capture"
         package="7346D660-1583-13D0-A3A5-00C0FFD756E3">
         <input name="BSPUuid" value=""/>
         <input name="BSPVersion" value="0"/>
         <input name="deviceIDOrNull" value="-1"/>
         <input name="inserttimeouttime" value="15000"/>
         <input name="sourcepresenttimeouttime" value="10000"/>
         <input name="capturetimeouttime" value="20000"/>
      </invoke>
      <bind activity="EventHandler" function="BioAPI_EventHandler"/>
   </assertion>
</package>
```

7.1.7 Assertions may have input parameters (but not output parameters). The input parameters are represented by **`<input>`** elements. At the time of testing, they are assigned actual values during the preparation of a test. For an assertion with parameters, the results of a test may depend on the values assigned to the parameters; therefore those values are an important part of the test being performed.

7.1.8 The means of assigning values to the input parameters of assertions is outside the scope of this Part of ISO/IEC 24709, but the values used in a test are documented in the standard test report (see clause 12).

7.1.9 Every assertion contains a primary activity reference. The primary activity of an assertion specifies the actions to be executed when the assertion is processed.

7.1.10 Activities are represented as `<activity>` elements (see 8.9). An activity is a sequence, usually parameterized, of actions that may include invocation of functions of the standard BioAPI interfaces (see clause 9). An activity may invoke other activities.

EXAMPLE (non-normative)

```
<?xml version='1.0' encoding="utf-8"?>
<package name="734ED660-1183-13D0-A3A5-0410FFD77AE9">
   <author>
      ISO/IEC SC37
   </author>
   <description>
      Abcde abcde abcde
   </description>

   <activity name="CreateTemplate">
      <input name="BSPUuid"/>
      <input name="BSPVersion"/>
      <input name="deviceIDOrNull"/>
      <input name="inserttimeouttime"/>
      <input name="sourcepresenttimeouttime"/>
      <input name="capturetimeouttime"/>

      <invoke activity="LoadAndAttach" break_on_break="true">
         <input name="BSPUuid" var="BSPUuid"/>
         <input name="BSPVersion" var="BSPVersion"/>
         <input name="deviceIDOrNull" var="deviceIDOrNull"/>
         <input name="BSP" value="1"/>
         <input name="eventtimeouttime"
            var="inserttimeouttime"/>
      </invoke>

      <wait_until
         timeout_var="sourcepresenttimeouttime"
         setvar="eventtimeoutflag"
         var="_sourcePresent"/>

      <assert_condition
            response_if_true="undecided"
            break_if_false="true">
         <description>
            We are testing
         </description>
         <not var="eventtimeoutflag"/>
      </assert_condition>

      <invoke function="BioSPI_Capture">
         <input name="BSPHandle" value="1"/>
         <input name="Purpose" var="__BioAPI_PURPOSE_ENROLL"/>
         <input name="Timeout" var="capturetimeouttime"/>
         <output name="CapturedBIR" setvar="bir"/>
         <output name="AuditData" setvar="auditbir"/>
         <return setvar="return"/>
      </invoke>
   </activity>

   <activity name="LoadAndAttach">
      ...
   </activity>

</package>
```

7.2 Variables

7.2.1 The names of variables in the assertion language shall consist of strings of ISO/IEC 10646 characters that match the "NCName" production in W3C XML Namespaces.

7.2.2 Variables whose name begins with a LOW LINE character ("_") are global variables. Any other variables are local variables.

7.2.3 Global variables shall have a lifetime that extends over the processing of an entire assertion. They may be created within any activity (see 8.6.2.3, 8.7.2.3, 8.12.2.5.1, and 8.17.2.9) but shall be associated with the entire assertion and shall not be destroyed until the processig of the assertion terminates. Global variables may also be created as input parameters of assertions (see 8.3.2.4).

7.2.4 Local variables may be created within any activity (see 8.5.2.5, 8.6.2.3, 8.7.2.3, 8.12.2.5.1, 8.17.2.9, shall be associated with that activity, and shall be destroyed as soon as that activity terminates.

7.2.5 Input and output parameters of activities are local variables of the activities. The only difference between input parameters (of assertions and activities) and ordinary variables is in the way they are created and acquire their initial value. The only difference between output parameters of activities and ordinary variables is in the way they are destroyed and in the disposition of their final value.

7.2.6 In the assertion language, variables do not have a data type. The values of all variables are ISO/IEC 10646 character strings of unlimited length.

7.2.7 A value shall be interpreted as an integer in two cases:

a) when it is evaluated by a numeric operation (see sub-clauses 8.23 to 8.28); or

b) when it is passed to a standard BioAPI interface function that accepts an integer as input.

7.2.8 In case a) of 7.2.7, a typical numeric operation gets one or more values (character strings representing integers) as input and returns a value. Although the values may exist in some native numeric representation for some time during the execution of the operation, this fact is never reflected outside the operation.

7.2.9 In case b) of 7.2.7, the conversion from character string to integer is performed as part of the invocation of the function, and does not manifest itself in the syntax of the assertion language.

7.2.10 Likewise, a variable may be set to a (character string) value resulting from a conversion of an integer to a character string. This happens when the variable is set from an output parameter of a standard BioAPI interface function or from its return value.

7.2.11 The conversion from integer to character string is performed as part of the invocation of the function, and does not manifest itself in the syntax of the assertion language.

7.2.12 The same is true of data types other than integers, which are used on the standard BioAPI interfaces. These native data types are enumerations, handles, addresses of functions, and addresses of data (see clause 9), and are all represented as character strings in the assertion language.

7.3 Built-in variables

7.3.1 A number of global variables are built into the assertion language. Their names all begin with two consecutive LOW LINE characters ("_"). These variables are specified in clause 10.

7.3.2 The abstract test engine shall create and assign all built-in variables before starting execution of the primary activity of an assertion and shall not destroy them until the execution of that activity terminates. Most of the built-in variables have a value that never changes and is specified in clause 10.1. The value of the other built-in variables may change as specified in clause 10.2.

NOTE Names of global variables beginning with two consecutive LOW LINE characters ("_") cannot appear as the value of the `setvar` attribute of the elements `<output>`, `<return>`, and `<wait_until>` (see 8.6.2.3, 8.7.2.3 and 8.17.2.9.1, respectively), or as the value of the `name` attribute of the elements `<input>`, `<set>`, `<add>`, and `<subtract>` (see 8.3.2.3, 8.12.2.2, 8.13.2.2, and 8.14.2.2, respectively). Therefore, it is not possible to explicitly modify the value of a built-in variable.

7.3.3 In the execution of any activity, built-in variables shall only be updated during the time intervals in which the activity is interruptible (see 8.9.2.21). At the end of any such interval, the values of all related built-in variables shall be consistent with one another.

NOTE It is always safe to reference two or more related built-in variables in a condition within an `<only_if>`, `<wait_until>`, and so on. If one wants to reference multiple built-in variables in a series of elements (for example, to copy two or more variables to ordinary variables using `<set>`), the elements need to be placed in an activity with the attribute `atomic="true"`.

7.4 Representation of integers

7.4.1 Non-negative integers shall be represented as strings of one or more ISO/IEC 10646 characters in the range DIGIT ZERO to DIGIT NINE ("`0`" to "`9`").

7.4.2 Negative integers shall be represented as the corresponding positive integer, preceded by a HYPHEN-MINUS character ("`-`").

7.4.3 The canonical representation of a positive integer is the one that contains no leading DIGIT ZERO characters. The canonical representation of the integer zero is the one consisting of a single DIGIT ZERO character. The canonical representation of a negative integer is the one consisting of a HYPHEN-MINUS character followed by the canonical representation of the corresponding positive integer.

7.5 Representation of booleans

7.5.1 The boolean TRUE shall be represented as the string of ISO/IEC 10646 characters "`true`". The boolean FALSE shall be represented as the string of ISO/IEC 10646 characters "`false`".

7.5.2 This representation is canonical.

7.6 Representation of universally unique identifiers

7.6.1 Universally unique identifiers (UUIDs) shall be represented as strings of ISO/IEC 10646 characters. Each string shall contain characters from the union of the following sets:

a) DIGIT ZERO to DIGIT NINE ("`0`" to "`9`"), each representing a hexadecimal digit 0 through 9;

b) LATIN CAPITAL LETTER A to LATIN CAPITAL LETTER F ("`A`" to "`F`"), each representing a hexadecimal digit A through F;

c) LATIN SMALL LETTER A to LATIN SMALL LETTER F ("`a`" to "`f`"), each representing a hexadecimal digit A through F;

d) HYPHEN-MINUS ("`-`").

7.6.2 There shall be exactly 32 hexadecimal digits. There shall also be four HYPHEN-MINUS characters inserted at the following positions:

a) between the 8th and the 9th hexadecimal digit;

b) between the 12th and the 13th hexadecimal digit;

c) between the 16th and the 17th hexadecimal digit; and

d) between the 20th and the 21st hexadecimal digit.

7.6.3 The canonical representation of a UUID is the one in which the characters listed in 7.6.1 b) are not used.

7.7 Representation of octet strings

7.7.1 Octet strings shall be represented as strings of ISO/IEC 10646 characters. Each string shall contain an even number (possibly zero) of characters from the union of the following sets:

a) DIGIT ZERO to DIGIT NINE ("`0`" to "`9`"), each representing a hexadecimal digit 0 through 9;

b) LATIN CAPITAL LETTER A to LATIN CAPITAL LETTER F ("`A`" to "`F`"), each representing a hexadecimal digit A through F;

c) LATIN SMALL LETTER A to LATIN SMALL LETTER F ("`a`" to "`f`"), each representing a hexadecimal digit A through F.

7.7.2 The canonical representation of an octet string is the one in which the characters listed in 7.7.1 b) are not used.

7.8 XML documents

7.8.1 Assertions and activities shall be contained in W3C XML 1.0 documents. The version of XML shall be "`1.0`". The character encoding shall be either "`utf-8`" or "`utf-16`".

7.8.2 The root element of all such XML documents shall be the `<package>` element (see 8.1).

7.8.3 The XML documents shall be valid according to the XML Schema specified in Annex A and according to the ASN.1 module specified in Annex B.

NOTE An implementation may use either the XML Schema or the ASN.1 Schema for validating assertions, because the two schemas are equivalent.

8 Elements of the assertion language

8.1 Element <package>

8.1.1 Syntax

8.1.1.1 This element shall have the following attribute:

— **`name`** (required) – the value of this attribute shall be a valid package name (see 8.1.2.5).

8.1.1.2 This element shall have a content consisting of the following (in order):

a) one **`<author>`** element – this element shall contain the name or description of the author of the package (a character string);

b) one **`<description>`** element – this element shall contain the description of the package (a character string);

c) zero or more **`<assertion>`** elements – this element represents an assertion and is specified in 8.2;

d) zero or more **`<activity>`** elements – this element represents an activity and is specified in 8.9.

8.1.2 Semantics

8.1.2.1 A package is a named container for assertions and activities.

8.1.2.2 Both assertions and activities may be referenced from a context outside their package.

8.1.2.3 Assertions are not referenced from any package, but may be referenced from contexts external to the assertion language. In particular, they are referenced when setting up a test and within test reports (see clause 12).

8.1.2.4 Activities are normally referenced from assertions (either in an **`<invoke>`** element or in a **`<bind>`** element) or from other activities. In both cases, they may be referenced from either the same package or a different package.

8.1.2.5 The names of packages shall consist of strings of ISO/IEC 10646 characters that represent universally unique identifiers (see 7.6). Each package name shall be globally unique.

8.1.2.6 Two package names shall be considered equal if and only if they contain the same sequence of digits and letters regardless of case.

8.1.2.7 The assertions in a package shall have distinct names. The activities in a package shall also have distinct names.

NOTE An assertion and an activity in a package may have identical names.

8.1.3 Example (non-normative)

```
<?xml version='1.0' encoding="utf-8"?>
<package name="734ED660-1183-13D0-A3A5-0410FFD77AE9">
   <author>
      ISO/IEC SC37
   </author>
   <description>
      Abcde abcde abcde
   </description>

   <assertion name="Capture2" model="BSPTesting">
      <description>
         Test the BioSPI_Capture function of a BSP.
      </description>
      <invoke activity="Capture"
         package="7346D660-1583-13D0-A3A5-00C0FFD756E3">
         <input name="BSPUuid" value=""/>
         <input name="BSPVersion" value="0"/>
         <input name="deviceIDOrNull" value="0"/>
         <input name="inserttimeouttime" value="15000"/>
```

```
            <input name="sourcepresenttimeouttime" value="10000"/>
            <input name="capturetimeouttime" value="20000"/>
        </invoke>
        <bind activity="EventHandler" function="BioAPI_EventHandler"/>
    </assertion>
</package>
```

8.2 Element `<assertion>` (child of `<package>`)

8.2.1 Syntax

8.2.1.1 This element shall have the following attributes:

a) **`name`** (required) – the value of this attribute shall be a valid assertion name (see 8.2.2.7);

b) **`model`** (required) – this attribute shall have one of the values "**`applicationTesting`**", "**`frameworkTesting`**", "**`BSPTesting`**"; it shall indicate the conformance testing model of the assertion.

8.2.1.2 This element shall have a content consisting of the following (in order):

a) one **`<description>`** element – this element shall contain a description of the assertion (a character string);

b) zero or more **`<input>`** elements – this element represents an input parameter of the assertion and is specified in 8.3;

c) one **`<invoke>`** element – this element represents the invocation of the primary activity of the assertion and is specified in 8.4;

d) zero or more **`<bind>`** elements – this element represents a binding between a standard BioAPI interface function and an activity, and is specified in 8.8.

8.2.2 Semantics

8.2.2.1 The **`model`** indicates which conformance testing model (see 6.2) shall be selected for the processing of the assertion, and thus determines which component(s) of the model shall be associated with the abstract test engine.

8.2.2.2 If there is only one testing component, all calls required to be made to standard BioAPI interface functions of other standard BioAPI components as a result of processing the assertion shall be made by this component, and all incoming calls to this component shall be processed according to the content of the assertion.

8.2.2.3 If there are two testing components, either component can make calls to standard BioAPI interface functions of other components, and all incoming calls to either testing component shall be processed according to the content of the assertion.

8.2.2.4 If **`model`** is "**`applicationTesting`**", the testing model for BioAPI applications (see 6.2.5.1) shall be selected for this assertion, and the abstract test engine shall be associated with the application-testing framework.

8.2.2.5 If **`model`** is "**`frameworkTesting`**", the testing model for BioAPI frameworks (see 6.2.5.2) shall be selected for this assertion, and the abstract test engine shall be associated with both the framework-testing application and the framework-testing BSP at the same time.

8.2.2.6 If `model` is "`BSPTesting`", the testing model for BioAPI BSPs (see 6.2.5.3) shall be selected for this assertion, and the abstract test engine shall be associated with the BSP-testing application.

8.2.2.7 The names of assertions shall consist of strings of ISO/IEC 10646 characters that match the "NCName" production in W3C XML Namespaces.

8.2.2.8 The names of assertions shall be unique within a package.

8.2.3 Example (non-normative)

```
<assertion name="CreateTemplate1" model="BSPTesting">
   <description>
      Test the BioSPI_CreateTemplate function of a BSP.
      The UUID and version of the BSP must be provided
      as input to the test.
   </description>
   <input name="_uuid"/>
   <input name="_version"/>
   <invoke activity="CreateTemplate"
      package="7346D660-1583-13D0-A3A5-00C0FFD756E3">
      <input name="BSPUuid" var="_uuid"/>
      <input name="BSPVersion" var="_version"/>
      <input name="deviceIDOrNull" value="0"/>
      <input name="inserttimeouttime" value="15000"/>
      <input name="sourcepresenttimeouttime" value="10000"/>
      <input name="capturetimeouttime" value="20000"/>
   </invoke>
   <bind activity="EventHandler" function="BioAPI_EventHandler"/>
</assertion>
```

8.3 Element `<input>` (child of `<assertion>`)

8.3.1 Syntax

8.3.1.1 This element shall have the following attribute:

— **name** (required) – the value of this attribute shall be a valid global variable name (see 7.2), which is to be the name of an input parameter of the assertion.

8.3.1.2 The content of this element shall be empty.

8.3.2 Semantics

8.3.2.1 This element represents an input parameter of an assertion.

NOTE Assertions cannot have output parameters.

8.3.2.2 Assertions with parameters cannot be processed unless a value is assigned to each parameter. The means of assigning values to input parameters of assertions is outside the scope of this Part of ISO/IEC 24709.

8.3.2.3 The value of the `name` attribute shall be a valid global variable name (see 7.2) and shall not begin with two consecutive LOW LINE characters ("_").

8.3.2.4 Input parameters of assertions shall be created and set to the values provided as input to the test.

8.3.2.5 Assigning values to assertion parameters is an essential part of the preparation of a test. For some implementations under test, the results of a test may depend on the values assigned to the parameters. The values used in a given test are documented in the standard test report (see Clause 12).

8.3.2.6 The names of the input parameters of an assertion shall be valid global variable names (see 7.2). The names of all global variables (including input parameters) shall be distinct.

NOTE The values assigned to assertion parameters cannot be changed during the processing of an assertion, because all the language elements that assign or modify a global variable (see 8.6.2.3, 8.7.2.3, 8.12.2.2, 8.13.2.2, 8.14.2.2, and 8.17.2.9.1) do not permit the use of an assertion parameter as the variable to be modified.

8.3.3 Example (non-normative)

```
<input name="_uuid"/>
```

8.4 Element `<invoke>` (child of `<assertion>`)

8.4.1 Syntax

8.4.1.1 This element shall have the following attributes:

a) `activity` (required) – the value of this attribute shall be the name of an activity;

b) `package` (optional) – if this attribute is present, its value shall be the name of the package that contains the activity named in a).

8.4.1.2 This element shall have a content consisting of the following:

— zero or more `<input>` elements – this element provides a value for an input parameter of the activity, and is specified in 8.5.

8.4.2 Semantics

8.4.2.1 This element designates the primary activity of the assertion and specifies the actual input parameters of its invocation. The processing of the assertion shall result in the execution of that activity with the provided input parameters. There shall be exactly one such activity for each assertion.

8.4.2.2 The activity may contain `<assert_condition>` elements with a `break_if_false` attribute (see 8.18.2.3). If a break occurs during the execution of the activity, the processing of the entire assertion shall terminate.

8.4.2.3 The set of `<input>` elements of the invocation shall match the input parameters of the activity as specified in 8.5.2.2.

8.4.2.4 The `package` attribute is mandatory if the activity is in a different package from the one that contains the assertion, and is optional otherwise.

8.4.2.5 The activity may or may not have output parameters. If the primary activity of an assertion has output parameters, the values of all the output parameters are discarded when the invoked activity terminates.

NOTE This is similar to invoking an activity with output parameters from another activity without providing any `<output>` elements matching the output parameters. That is allowed, and is useful when the invoking activity is not interested in the values of the output parameters of the invoked activity.

8.4.2.6 The `<invoke>` element of assertions shall not have a `break_on_break` attribute (see 8.15.2.6.5) and shall not have an `<only_if>` child element (see 8.16).

8.4.3 Example (non-normative)

```
<invoke activity="CreateTemplate"
   package="7346D660-1583-13D0-A3A5-00C0FFD756E3">
   <input name="BSPUuid" var="_uuid"/>
   <input name="BSPVersion" var="_version"/>
   <input name="deviceIDOrNull" value="0"/>
   <input name="inserttimeouttime" value="15000"/>
   <input name="sourcepresenttimeouttime" value="10000"/>
   <input name="capturetimeouttime" value="20000"/>
</invoke>
```

8.5 Element `<input>` (child of `<invoke>`)

8.5.1 Syntax

8.5.1.1 This element shall have the following attributes:

a) `name` (required) – the value of this attribute shall be the name of an input parameter of the activity or standard BioAPI interface function being invoked;

b) `value` (optional) – if this attribute is present, its value shall be the value to be assigned to the input parameter named in a);

c) `var` (optional) – if this attribute is present, its value shall be a valid name of a variable (see 7.2), whose value is to be assigned to the input parameter.

8.5.1.2 Exactly one of the attributes `value` and `var` shall be present.

8.5.1.3 The content of this element shall be empty.

8.5.2 Semantics

8.5.2.1 This element represents the assignment of a value to an input parameter of an activity or a standard BioAPI interface function (see clause 9) in an invocation.

8.5.2.2 The set of `<input>` elements of an invocation shall match the input parameters of the activity or of the standard BioAPI interface function being invoked, as specified in the two following sub-clauses.

8.5.2.2.1 In the case of activity invocation, for each `<input>` element of the activity, there shall be at most one `<input>` element in the invocation whose `name` attribute equals the `name` attribute of the former, and vice versa. For each `<input>` element in the invocation, there shall be an `<input>` element of the activity whose `name` attribute equals the `name` attribute of the former. The order of the `<input>` elements in the activity and in the invocation need not be the same.

8.5.2.2.2 In the case of function invocation, for each input parameter of the function, there shall be at most one `<input>` element in the invocation whose `name` attribute equals the name of the input parameter of the function, and vice versa. For each `<input>` element in the invocation, there shall be an input parameter of the function whose name equals the `name` attribute of the former. The `<input>` elements in the invocation may appear in any order.

NOTE Clause 9 specifies the names of all standard BioAPI interface functions and the names of their input and output parameters, as they are to appear in the invocation of those functions.

8.5.2.3 If the **`var`** attribute is present, its value shall be the name of a variable (see 7.2) that already exists prior to the invocation. The value of the variable shall be assigned to the input parameter. The variable may be either global or local.

8.5.2.4 If the **`value`** attribute is present, its value shall be assigned to the input parameter.

8.5.2.5 In the case of activity invocation, the input parameter shall be created as a local variable of the activity being invoked, and shall be set to the specified value.

8.5.2.6 In the case of function invocation, the specified value shall be converted into a native form suitable for the corresponding parameter of the underlying standard BioAPI interface function as specified in clause 9 and its individual sub-clauses.

8.5.3 Example (non-normative)

```
<input name="BSPUuid" var="_uuid"/>
```

8.6 Element `<output>` (child of `<invoke>`)

8.6.1 Syntax

8.6.1.1 This element shall have the following attributes:

a) **`name`** (required) – the value of this attribute shall be the name of an output parameter of the activity or standard BioAPI interface function being invoked;

b) **`setvar`** (required) – the value of this attribute shall be a valid name of a variable (see 7.2), to which the value of the output parameter is to be assigned.

8.6.1.2 The content of this element shall be empty.

8.6.2 Semantics

8.6.2.1 This element represents the assignment of the value of an output parameter of an activity or standard BioAPI interface function to a variable (see clause 9) in an invocation.

8.6.2.2 The set of **`<output>`** elements of an invocation shall match the output parameters of the activity or standard BioAPI interface function being invoked, as specified in the two following sub-clauses.

8.6.2.2.1 In the case of activity invocation, for each **`<output>`** element of the activity, there shall be at most one **`<output>`** element in the invocation whose **`name`** attribute equals the **`name`** attribute of the former. For each **`<output>`** element in the invocation, there shall be an **`<output>`** element of the activity whose **`name`** attribute equals the **`name`** attribute of the former. The order of the **`<output>`** elements in the activity and in the invocation need not be the same.

8.6.2.2.2 In the case of function invocation, for each output parameter of the function, there shall be at most one **`<output>`** element in the invocation whose **`name`** attribute equals the name of the output parameter of the function. For each **`<output>`** element in the invocation, there shall be an output parameter of the function whose name equals the **`name`** attribute of the former. The **`<output>`** elements in the invocation may appear in any order.

NOTE Clause 9 specifies the names of all standard BioAPI interface functions and the names of their input and output parameters, as they are to appear in the invocation of those functions.

8.6.2.3 The value of the `setvar` attribute shall be a valid global or local variable name (see 7.2), shall not be an input parameter of the assertion being processed (see 8.3), and shall not begin with two consecutive LOW LINE characters ("_"). The variable may already exist prior to the invocation, or may be a new variable that is to be created when the invoked activity or function terminates.

8.6.2.4 The values of the setvar attributes of the <output> elements in an invocation shall all be distinct.

8.6.2.5 When the invoked activity or function terminates, if the variable does not exist, it shall be created. If the variable is local, it shall be associated with the current (invoking) activity, so that it will be destroyed when that activity terminates.

8.6.2.6 In the case of activity invocation, the variable shall be set to the final value of the output parameter of the invoked activity, which is the value that the parameter has immediately before being destroyed. The output parameter shall exist when the invoked activity terminates.

NOTE The output parameter (as a local variable) may be created at any time during the execution of the invoked activity.

8.6.2.7 In the case of function invocation, the value of the corresponding output parameter of the underlying standard BioAPI interface function shall be converted from its native form into a character string as specified in the individual sub-clauses of clause 9, and the variable shall be set to the resulting character string.

8.7 Element `<return>` (child of `<invoke>`)

8.7.1 Syntax

8.7.1.1 This element shall have the following attribute:

— **setvar** (required) – the value of this attribute shall be a valid name of a variable (see 7.2), to which the return value is to be assigned.

8.7.1.2 The content of this element shall be empty.

8.7.2 Semantics

8.7.2.1 This element represents the assignment of the return value of a standard BioAPI interface function to a variable (see clause 9 and its individual sub-clauses) in an invocation.

8.7.2.2 There shall be at most one `<return>` element in the invocation of any standard BioAPI interface function.

8.7.2.3 The value of the `setvar` attribute shall be a valid global or local variable name (see 7.2), shall not be an input parameter of the assertion being processed (see 8.3), and shall not begin with two consecutive LOW LINE characters ("_"). This variable may already exist prior to the invocation, or may be a new variable that is to be created when the invoked activity or function terminates.

8.7.2.4 When the invoked function terminates, if the variable does not exist, it shall be created. If the variable is local, it shall be associated with the current (invoking) activity, so that it will be destroyed when the current activity terminates.

8.7.2.5 The return value of the standard BioAPI interface function shall be converted from its native form into a character string as specified in clause 9, and the variable shall be set to the resulting character string.

8.8 Element `<bind>` (child of `<assertion>`)

8.8.1 Syntax

8.8.1.1 This element shall have the following attributes:

a) **`function`** (required) – the value of this attribute shall be the name of a standard BioAPI interface function (see clause 9);

b) **`activity`** (required) – the value of this attribute shall be the name of an activity;

c) **`package`** (optional) – if this attribute is present, its value shall be the name of the package that contains the activity named in b).

8.8.1.2 The content of this element shall be empty.

8.8.2 Semantics

8.8.2.1 This element represents an association between a standard BioAPI interface function (exposed by a testing component) and an activity, whereby the activity is automatically invoked in response to an incoming call to the standard BioAPI interface function.

8.8.2.2 The association (binding) shall be in effect through the processing of the assertion.

NOTE Bindings are established before the primary activity of the assertion starts execution, and cannot be disabled or changed.

8.8.2.3 The standard BioAPI interface function identified by the **`function`** attribute shall be one of the functions exposed by the testing component(s).

NOTE The conformance testing model of an assertion, specified by the `model` attribute of the assertion, determines which testing component(s) exist for that assertion (see 6.2), and thus which functions may be referenced in a binding.

8.8.2.4 The same standard BioAPI interface function shall not be referenced in two different **`<bind>`** elements in the same assertion.

8.8.2.5 The bound activity shall have input and output parameters determined from the input and output parameters of the function invocation scheme (specified in clause 9) of the standard BioAPI interface function to which it is bound, as follows:

a) in the conformance testing model for BioAPI applications, all the input and output parameters of the function invocation shall be input parameters of the bound activity; in addition, there shall be an input parameter named **`return`**;

b) in the conformance testing model for BioAPI frameworks and in the conformance testing model for BioAPI BSPs, all the input parameters of the function invocation shall be output parameters of the bound activity, and all the output parameters of the function invocation shall be input parameters of the bound activity; in addition, there shall be an output parameter named **`return`**.

8.8.2.6 The bound activity shall be invoked each time a testing component receives an incoming call to the standard BioAPI interface function to which the activity is bound, as specified in 8.9.2.7 and 8.9.2.8.

8.9 Element `<activity>` (child of `<package>`)

8.9.1 Syntax

8.9.1.1 This element shall have the following attributes:

a) `name` (required) – the value of this attribute shall be a valid activity name (see 8.9.2.22);

b) `atomic` (optional) – this attribute shall indicate whether the execution of the activity is not interruptible (the default is "`false`").

8.9.1.2 This element shall have a content consisting of the following (in order):

a) zero or more `<input>` elements – this element represents an input parameter of the activity and is specified in 8.10;

b) zero or more `<output>` elements – this element represents an output parameter of the activity and is specified in 8.11;

c) zero or more occurrences of any of the following elements in any order:

1) `<set>` – this element represents the assignment of a value to a new variable or to an existing variable, and is specified in 8.12;

2) `<add>` – this element represents the addition of an integer to the value (representing an integer) of an existing variable, and is specified in 8.13;

3) `<subtract>` – this element represents the subtraction of an integer from the value (representing an integer) of an existing variable, and is specified in 8.14;

4) `<wait_until>` – this element represents a suspension of the execution of the current activity until a condition is verified, and is specified in 8.17;

5) `<assert_condition>` – this element represents the issuance of a conformity response based on a given condition, and is specified in 8.18;

6) `<invoke>` – this element represents the invocation of an activity or of a standard BioAPI interface function, and is specified in 8.15.

8.9.2 Semantics

8.9.2.1 Activities are the executable units of the assertion language and consist of ordered sequences of zero or more elements representing the following actions:

a) assigning a variable;

b) suspending execution of the activity until a condition is verified;

c) invoking a standard BioAPI interface function;

d) issuing a conformity response based on a condition;

e) invoking an activity.

8.9.2.2 The abstract test engine shall perform the actions of an activity in order.

8.9.2.3 An activity shall have a priority, which can be low, medium, or high. At most one activity may be in execution at any given time during the processing of an assertion. The abstract test engine shall have the ability to interrupt the execution of an activity in order to execute an activity with a higher priority (see 8.9.2.21).

8.9.2.4 When an assertion is submitted for processing, the abstract test engine shall invoke the primary activity of the assertion as a high-priority activity.

8.9.2.5 When an activity invokes another activity, the invoked activity shall be given the same priority as the invoking activity.

8.9.2.6 When one of the following elements:

a) a `<wait_until>` element; or

b) an `<invoke>` element invoking a standard BioAPI interface function

occurs for the first time in the primary activity of an assertion or in an activity that has been (directly or indirectly) invoked by it, the priority of the activity in which the element occurs shall be changed from high to low, immediately before the element is processed. If the activity in which the element occurs is not the primary activity of an assertion, the priority of every activity (including the primary activity) that has (directly or indirectly) invoked that activity shall be changed to low as well.

8.9.2.7 In the conformance testing model for BioAPI applications (see 6.2.5.1), when the testing component receives an incoming call to a standard BioAPI interface function, the actions specified in the four following sub-clauses shall be performed.

8.9.2.7.1 If the function being called belongs to the BioAPI interface, the incoming call shall be relayed to the same standard BioAPI interface function exposed by the normal framework. The new (relayed) call shall have the same native parameter values as the original incoming call.

8.9.2.7.2 If the function being called belongs to the application callback interface, the incoming call shall be relayed to the same standard BioAPI interface function exposed by the application under test. The new (relayed) call shall have the same native parameter values as the original incoming call.

8.9.2.7.3 When the relayed call returns but before returning from the original incoming call, if there is an activity bound to the standard BioAPI interface function being called (see 8.8), the abstract test engine shall interrupt the execution of the current low-priority activity and shall invoke the bound activity as a high-priority activity (but see 8.9.2.9).

8.9.2.7.4 The input parameters of the bound activity invocation shall be set from the native input parameters of the incoming call and from the native output parameters and return value of the relayed call that has just returned, as specified in the sub-clause of clause 9 with the heading "Bound Activity Invocation Input" relative to the standard BioAPI interface function.

8.9.2.8 In the conformance testing model for BioAPI frameworks (see 6.2.5.2) and in the conformance testing model for BioAPI BSPs (see 6.2.5.3), when the testing component receives an incoming call, the actions specified in the four following sub-clauses shall be performed.

8.9.2.8.1 If there is an activity bound to the function being called (see 8.8), the abstract test engine shall interrupt the execution of the current low-priority activity and shall invoke the bound activity (but see 8.9.2.9) with its priority set as follows:

a) if the function is `BioSPI_Cancel`, the priority shall be high;

b) if the function belongs to the BioSPI interface but is not `BioSPI_Cancel`, the priority shall be medium;

c) otherwise, the priority shall be high.

8.9.2.8.2 The input parameters of the bound activity invocation shall be set from the native input parameters of the incoming call as specified in the sub-clause of clause 9 with the heading "Bound Activity Invocation Input" relative to the standard BioAPI interface function.

8.9.2.8.3 When the bound activity terminates, the native output parameters and the return value of the incoming call shall be set from the output parameters of the activity as specified in the sub-clause of clause 9 with the heading "Bound Activity Invocation Output" relative to the standard BioAPI interface function.

8.9.2.8.4 If there is no activity bound to the function being called, then the abstract test engine shall set the native output parameters and return value of the incoming call as specified in the individual sub-clauses of clause 9 with the heading "Default Output".

8.9.2.9 If an incoming call to a function that has a bound activity (with a given priority) is received while the abstract test engine is executing another activity with the same or a higher priority, the new activity invocation shall be added to a queue, using a separate queue for each priority.

8.9.2.10 Each time an activity terminates or the priority of the current activity is changed from high to low, the abstract test engine shall determine which queue, among those that are not empty, holds the activity invocations with the highest priority. The oldest activity invocation in that queue shall be removed from the queue and processed.

8.9.2.11 The execution of any activity shall start with the creation of an execution context that is relative to one particular invocation of the activity. The execution context shall act as a container for the local variables of the activity (including its input and output parameters) and shall include an indication of the current execution position within the activity.

8.9.2.12 After the execution context is created, each of the input parameters of the activity shall be created (as a local variable) in the execution context and shall be set to the following value:

a) if the invocation of the activity provides an `<input>` element for that output parameter, the provided value (see 8.5.2.5);

b) otherwise, an empty string.

8.9.2.13 The elements of the activity shall then be processed in order, from the first element to the last element, unless a break occurs as specified in 8.9.2.17.

8.9.2.14 If no break occurs, then after the last element of the activity has been processed, one of the following actions shall be performed for each output parameter of the activity:

a) if the invocation of the activity provides an `<output>` element for that output parameter, the final value of the output parameter shall be assigned to the variable referenced in the `<output>` element as specified in 8.6.2.3;

b) otherwise, the final value of the output parameter shall be discarded.

8.9.2.15 The execution context relative to the present invocation of the activity shall then be destroyed.

8.9.2.16 If the activity was invoked by an activity, the execution of the invoking activity shall resume. If the activity was the primary activity of an assertion, the processing of the entire assertion shall terminate.

8.9.2.17 A break is said to occur within the execution of an activity in the following two cases:

a) if an activity contains an `<assert_condition>` element with a `break_if_false` attribute with a value of "`true`", when the condition in the `<assert_condition>` element evaluates to "`false`", a break is said to occur within that activity; and

b) if an activity contains an **`<invoke>`** element with a **`break_on_break`** attribute with a value of "**`true`**", when the invoked activity terminates because of a break (see 8.9.2.18), a break is said to occur within the invoking activity.

8.9.2.18 When a break occurs within an activity (because of either 8.9.2.17 a) or b)), the abstract test engine shall:

a) skip the processing of the remaining elements of the activity;

b) skip the assignment of the values of the output parameters to the variables referenced in the corresponding **`<output>`** elements in the invocation of the activity (if any);

c) proceed with the destruction of the execution context and resume execution of the invoking activity (if any).

NOTE Item b) implies that, if a variable corresponding to an output parameter did not exist before the invocation, it is not created when the invoking activity is resumed.

NOTE If 8.9.2.17 b) applies to the invoking activity, the execution of the invoking activity will be abandoned immediately after being resumed.

8.9.2.19 Activities may create new global variables or change the value of existing ones.

8.9.2.20 An activity may not be interrupted by another activity with the same or a lower priority (see 8.9.2.9).

8.9.2.21 An activity may be interrupted by an activity with a higher priority. Interruptions may occur at any time, either before or after each element of the activity, or during the processing of an element of the activity, with the following exceptions:

a) no interruption shall occur during the processing of a **`<set>`**, **`<add>`**, **`<subtract>`**, or **`<assert_condition>`** element;

b) during the processing of an **`<invoke>`** of a standard BioAPI interface function, no interruption shall occur during the evaluation of the condition (if any) and the assignment of the input parameters, and during the disposition of the output parameters and return value.

NOTE Interruptions are allowed while the function is in execution.

c) during the processing of a **`<wait_until>`**, no interruption shall occur during each evaluation of the condition;

NOTE Interruptions are allowed between two consecutive evaluations of the condition.

d) an activity that has an **`atomic`** attribute with the value "**`true`**" shall not be interrupted at any time (from the creation of the input parameters until after the destruction of the output parameters).

8.9.2.22 The names of activities shall consist of strings of ISO/IEC 10646 characters that match the "NCName" production in W3C XML Namespaces.

8.9.2.23 The names of activities shall be unique within a package.

8.9.3 Examples (non-normative)

```
<?xml version='1.0' encoding="utf-8"?>
<package name="73668660-1583-1AD0-A3A5-09C0FF4756E3">
   <author>
      ISO/IEC SC37
   </author>
   <description>
      Abcde abcde abcde
   </description>

<activity name="LoadAndAttach">
  <input name="BSPUuid"/>
  <input name="BSPVersion"/>
  <input name="deviceIDOrNull"/>
  <input name="BSP"/>
  <input name="eventtimeouttime"/>

  <set name="_deviceIDOrNull" var="deviceIDOrNull"/>

  <invoke function="BioSPI_BSPLoad">
    <input name="Reserved" value=""/>
    <input name="BSPUuid" var="BSPUuid"/>
    <input name="eventHandler" value="1"/>
    <input name="context" value="1"/>
    <return setvar="return"/>
  </invoke>

  <assert_condition
      response_if_true="undecided"
      break_if_false="true">
    <equal_to var1="return" var2="__BioAPI_OK"/>
  </assert_condition>

  <wait_until
      timeout_var="eventtimeouttime"
      setvar="eventtimeoutflag"
      var="_insert"/>

  <assert_condition
      response_if_true="undecided"
      break_if_false="true">
      <not var="eventtimeoutflag"/>
  </assert_condition>

  <invoke function="BioSPI_BSPAttach">
    <input name="BSPUuid" var="BSPUuid"/>
    <input name="Version" var="BSPVersion"/>
    <input name="DeviceID" var="_deviceID"/>
    <input name="Reserved1" value="0"/>
    <input name="Reserved2" value="0"/>
    <input name="BSPHandle" var="BSP"/>
    <input name="Reserved3" value="0"/>
    <input name="Reserved4" value=""/>
    <input name="Reserved5" value=""/>
    <input name="Reserved6" value=""/>
    <return setvar="return"/>
  </invoke>

  <assert_condition
      response_if_true="undecided"
      break_if_false="true">
    <equal_to var1="return" var2="__BioAPI_OK"/>
  </assert_condition>

</activity>
```

```
<activity name="EventHandler">
  <input name="BSPUuid"/>
  <input name="context"/>
  <input name="deviceID"/>
  <input name="reserved"/>
  <input name="eventType"/>

  <set name="_BSPUuid" var="BSPUuid"/>
  <set name="_context" var="context"/>

  <set name="_deviceID" var="deviceID">
    <only_if>
      <not>
        <existing var="_deviceID"/>
      </not>
      <or>
        <equal_to var1="eventType" var2="__BioAPI_NOTIFY_INSERT"/>
        <equal_to var1="eventType"
             var2="__BioAPI_NOTIFY_SOURCE_PRESENT"/>
      </or>
      <or>
        <equal_to var1="_deviceIDOrNull" value2="0"/>
        <equal_to var1="_deviceIDOrNull" var2="deviceID"/>
      </or>
    </only_if>
  </set>

  <set name="_insert" value="true">
    <only_if>
      <equal_to var1="eventType" var2="__BioAPI_NOTIFY_INSERT"/>
      <equal_to var1="_deviceID" var2="deviceID"/>
    </only_if>
  </set>

  <set name="_insert" value="false">
    <only_if>
      <equal_to var1="eventType" var2="__BioAPI_NOTIFY_REMOVE"/>
      <equal_to var1="_deviceID" var2="deviceID"/>
    </only_if>
  </set>

  <set name="_sourcePresent" value="true">
    <only_if>
      <equal_to var1="eventType"
            var2="__BioAPI_NOTIFY_SOURCE_PRESENT"/>
      <equal_to var1="_deviceID" var2="deviceID"/>
    </only_if>
  </set>

  <set name="_sourcePresent" value="false">
    <only_if>
      <equal_to var1="eventType"
            var2="__BioAPI_NOTIFY_SOURCE_REMOVED"/>
      <equal_to var1="_deviceID" var2="deviceID"/>
    </only_if>
  </set>

  <set name="_eventtype" var="eventType">
    <only_if>
      <equal_to var1="_deviceID" var2="deviceID"/>
    </only_if>
  </set>

  <assert_condition
      response_if_false="fail">
    <equal_to var1="reserved" value2="0"/>
  </assert_condition>
```

```
</activity>


<activity name="CreateTemplate">
  <input name="BSPUuid"/>
  <input name="BSPVersion"/>
  <input name="deviceIDOrNull"/>
  <input name="inserttimeouttime"/>
  <input name="sourcepresenttimeouttime"/>
  <input name="capturetimeouttime"/>

  <invoke activity="LoadAndAttach" break_on_break="true">
    <input name="BSPUuid" var="BSPUuid"/>
    <input name="BSPVersion" var="BSPVersion"/>
    <input name="deviceIDOrNull" var="deviceIDOrNull"/>
    <input name="BSP" value="1"/>
    <input name="eventtimeouttime" var="inserttimeouttime"/>
  </invoke>

  <wait_until
      timeout_var="sourcepresenttimeouttime"
      setvar="eventtimeoutflag"
      var="_sourcePresent"/>

  <assert_condition
      response_if_true="undecided"
      break_if_false="true">
   <not var="eventtimeoutflag"/>
  </assert_condition>

  <invoke function="BioSPI_Capture">
    <input name="BSPHandle" value="1"/>
    <input name="Purpose" var="__BioAPI_PURPOSE_ENROLL"/>
    <input name="Timeout" var="capturetimeouttime"/>
    <output name="CapturedBIR" setvar="bir"/>
    <output name="AuditData" setvar="auditbir"/>
    <return setvar="return"/>
  </invoke>

  ...
</activity>

</package>
```

8.10 Element `<input>` (child of `<activity>`)

8.10.1 Syntax

8.10.1.1 This element shall have the following attribute:

— **name** (required) – the value of this attribute shall be a valid local variable name (see 7.2), which is to be the name of an input parameter of the activity.

8.10.1.2 The content of this element shall be empty.

8.10.2 Semantics

8.10.2.1 This element represents an input parameter of an activity.

8.10.2.2 Input parameters of activities are local variables (see 7.2).

8.10.2.3 The names of the input parameters of an activity shall be valid local variable names (see 7.2).

8.10.2.4 The names of all local variables of an activity (including the input and output parameters of the activity) shall be distinct.

8.11 Element `<output>` (child of `<activity>`)

8.11.1 Syntax

8.11.1.1 This element shall have the following attribute:

— **name** (required) – the value of this attribute shall be a valid local variable name (see 7.2), which is to be the name of an output parameter of the activity.

8.11.1.1 The content of this element shall be empty.

8.11.2 Semantics

8.11.2.1 This element represents an output parameter of an activity.

8.11.2.2 Output parameters of activities are local variables (see 7.2).

8.11.2.3 The names of the output parameters of an activity shall be valid local variable names.

8.11.2.4 The names of all local variables of an activity (including the input and output parameters of the activity) shall be distinct.

8.12 Element `<set>`

8.12.1 Syntax

8.12.1.1 This element shall have the following attributes:

a) `name` (required) – the value of this attribute shall be a valid name of a variable (see 7.2), which is to be assigned a value (or created and then assigned a value);

b) `value` (optional) – if this attribute is present, its value shall be the value to be assigned to the variable referenced in a);

c) `var` (optional) – if this attribute is present, its value shall be a valid name of a variable (see 7.2), whose value is to be assigned to the variable referenced in a);

8.12.1.2 One and only one of the attributes `value` and `var` shall be present.

8.12.1.3 This element shall have a content consisting of the following:

— an optional `<only_if>` element – this element represents a condition and is specified in 8.16.

8.12.2 Semantics

8.12.2.1 This element represents the assignment of a value to a variable, possibly subject to a condition.

8.12.2.2 The value of the `name` attribute shall be a valid global or local variable name (see 7.2) , shall not be an input parameter of the assertion being processed (see 8.3), and shall not begin with two consecutive LOW

LINE characters ("_"). The variable may exist prior to the assignment or may be a new variable created by the assignment.

8.12.2.3 If there is an `<only_if>` child element, the condition shall be evaluated (see 8.16).

8.12.2.4 The result of the evaluation shall be a valid representation of a boolean (see 7.5).

8.12.2.5 If the result of the evaluation is "`true`", the variable whose name is the value of the `name` attribute shall be set as specified in the three following sub-clauses.

8.12.2.5.1 If the variable does not exist, it shall be created.

8.12.2.5.2 If the `var` attribute is present, its value shall be the name of an existing variable. The value of that variable shall be assigned to the variable whose name is the value of the `name` attribute.

8.12.2.5.3 If the `value` attribute is present, its value shall be assigned to the variable whose name is the value of the `name` attribute.

8.13 Element <add>

8.13.1 Syntax

8.13.1.1 This element shall have the following attributes:

a) `name` (required) – the value of this attribute shall be a valid name of a variable (see 7.2) whose value is to be modified;

b) `value` (optional) – if this attribute is present, its value shall represent an integer to be added to the current value (representing an integer) of the variable referenced in a);

c) `var` (optional) – if this attribute is present, its value shall be a valid name of a variable (see 7.2) whose value represents an integer to be added to the current value of the variable referenced in a);

8.13.1.2 One and only one of the attributes `value` and `var` shall be present.

8.13.1.3 This element shall have a content consisting of the following:

— an optional `<only_if>` element – this element represents a condition and is specified in 8.16.

8.13.2 Semantics

8.13.2.1 This element represents the addition of an integer to the value (representing an integer) of a variable (see 7.4), possibly subject to a condition.

8.13.2.2 The value of the `name` attribute shall be a valid global or local variable name (see 7.2), shall not be an input parameter of the assertion being processed (see 8.3), and shall not begin with two consecutive LOW LINE characters ("_"). The variable shall exist prior to the assignment and its value shall be a valid representation of an integer (see 7.4).

8.13.2.3 If there is an `<only_if>` child element, the condition shall be evaluated (see 8.16).

8.13.2.4 The result of the evaluation shall be a valid representation of a boolean (see 7.5).

8.13.2.5 If the result of the evaluation is "`true`", the variable whose name is the value of the `name` attribute shall be set as specified in the three following sub-clauses.

8.13.2.5.1 The variable shall be set to the canonical representation of an integer which is the sum of two operands. The first operand is the integer represented by the current value of the variable. The second operand shall be determined as follows.

8.13.2.5.2 If the `var` attribute is present, its value shall be the name of an existing variable. The value of that variable shall be a valid representation of an integer (see 7.4). This integer is the second operand.

8.13.2.5.3 If the `value` attribute is present, its value shall be a valid representation of an integer (see 7.4). This integer is the second operand.

8.14 Element `<subtract>`

8.14.1 Syntax

8.14.1.1 This element shall have the following attributes:

a) `name` (required) – the value of this attribute shall be a valid name of a variable (see 7.2) whose value is to be modified;

b) `value` (optional) – if this attribute is present, its value shall represent an integer to be subtracted from the current value of the variable referenced in a);

c) `var` (optional) – if this attribute is present, its value shall be a valid name of a variable (see 7.2) whose value represents an integer to be subtracted from the current value of the variable referenced in a);

8.14.1.2 One and only one of the attributes `value` and `var` shall be present.

8.14.1.3 This element shall have a content consisting of the following:

— an optional `<only_if>` element – this element represents a condition and is specified in 8.16.

8.14.2 Semantics

8.14.2.1 This element represents the subtraction of an integer from the value (representing an integer) of a variable (see 7.4), possibly subject to a condition.

8.14.2.2 The value of the `name` attribute shall be a valid global or local variable name (see 7.2), shall not be an input parameter of the assertion being processed (see 8.3), and shall not begin with two consecutive LOW LINE characters ("_"). The variable shall exist prior to the assignment and its value shall be a valid representation of an integer (see 7.4).

8.14.2.3 If there is an `<only_if>` child element, the condition shall be evaluated (see 8.16).

8.14.2.4 The result of the evaluation shall be a valid representation of a boolean (see 7.5).

8.14.2.5 If the result of the evaluation is "`true`", the variable whose name is the value of the `name` attribute shall be set as specified in the three following sub-clauses.

8.14.2.5.1 The variable shall be set to the canonical representation of an integer which is the difference between two operands. The first operand is the integer represented by the current value of the variable. The second operand shall be determined as follows.

8.14.2.5.2 If the `var` attribute is present, its value shall be the name of an existing variable. The value of that variable shall be a valid representation of an integer (see 7.4). This integer is the second operand.

8.14.2.5.3 If the `value` attribute is present, its value shall be a valid representation of an integer (see 7.4). This integer is the second operand.

8.15 Element `<invoke>` (child of `<activity>`)

8.15.1 Syntax

8.15.1.1 This element shall have the following attributes:

a) `activity` (optional) – the value of this attribute shall be the name of an activity;

b) `package` (optional) – if this attribute is present, its value shall be the name of the package that contains the activity named in a);

c) `break_on_break` (optional) – if this attribute is present, it shall have one of the values "`false`", "`true`"; it indicates whether a break occurring in the invoked activity shall cause the current activity to be abandoned as well; the default is "`false`";

d) `function` (optional) – the value of this attribute shall be the name of a standard BioAPI interface function.

e) `timeout_value` (optional) – if this attribute is present, its value indicates the maximum duration of time (in milliseconds) allowed for the invocation;

f) `timeout_var` (optional) – if this attribute is present, its value shall be a valid name of a variable (see 7.2), whose value indicates the maximum duration of time (in milliseconds) allowed for the invocation;

g) `setvar` (optional) – if this attribute is present, its value shall be a valid variable name (see 7.2).

8.15.1.2 Exactly one of the attributes `activity` and `function` shall be present.

8.15.1.3 The attributes `package` and `break_on_break` shall not be present unless the attribute `activity` is present.

8.15.1.4 The attributes `timeout_var`, `timeout_value`, and `setvar` shall not be present unless the attribute `function` is present.

8.15.1.5 At most one of the attributes `timeout_var` and `timeout_value` shall be present.

8.15.1.6 The `setvar` attribute shall not be present unless either the `timeout_value` attribute or the `timeout_var` attribute is present.

8.15.1.7 This element shall have a content consisting of the following (in order):

a) an optional `<only_if>` element – this element represents a condition and is specified in 8.16;

b) zero or more `<input>` elements – this element provides a value for an input parameter of the activity or standard BioAPI interface function being invoked, and is specified in 8.5;

c) zero or more `<output>` elements – this element references a variable which is to be assigned from an output parameter of the activity or function being invoked, and is specified in 8.6;

d) an optional `<return>` element – this element references a variable which is to be assigned from the return value of the function being invoked, and is specified in 8.7.

8.15.2 Semantics

8.15.2.1 This element represents an invocation of an activity or of a standard BioAPI interface function from an activity.

8.15.2.2 If there is an `<only_if>` child element, the condition shall be evaluated (see 8.16).

8.15.2.3 The result of the evaluation shall be a valid representation of a boolean (see 7.5).

8.15.2.4 If the result of the evaluation is "`true`", the invocation shall be performed as specified in the following sub-clauses, otherwise no further action shall be performed for this `<invoke>` element.

8.15.2.5 In case of activity invocation, the activity being invoked shall be an existing activity that may be either in the same package as the invoking activity, or in a different package (see 8.15.2.6.4). In case of function invocation, the function being invoked shall be one of the standard BioAPI interface functions specified in clause 9.

8.15.2.6 In the case of activity invocation, the five following sub-clauses apply.

8.15.2.6.1 The set of `<input>` elements of the invocation shall match the input parameters of the activity as specified in 8.5.2.2.

8.15.2.6.2 The set of `<output>` elements of the invocation shall match the output parameters of the activity as specified in 8.6.2.2.

8.15.2.6.3 The `<return>` element shall not be present.

8.15.2.6.4 The `package` attribute, if present, shall indicate the package in which the activity being invoked is. This attribute is mandatory if the activity being invoked is in a different package from the one that contains the invoking activity, and is optional otherwise.

8.15.2.6.5 If the attribute `break_on_break` is present and its value is "`true`", then if the activity being invoked is abandoned because of a break (see 8.9.2.17), the current activity shall be abandoned as well.

8.15.2.7 In the case of function invocation, the three following sub-clauses apply.

8.15.2.7.1 The set of `<input>` elements of the invocation shall match the input parameters of the function as specified in 8.5.2.2.

8.15.2.7.2 The set of `<output>` elements of the invocation shall match the output parameters of the function as specified in 8.6.2.2.

8.15.2.7.3 The `<return>` element may be present (see 8.7.2.2).

8.15.2.8 If the `timeout_value` attribute is present, its value shall be a valid representation of an integer (see 7.4) that specifies the maximum duration allowed for the invocation (in milliseconds). If the `timeout_var` attribute is present, its value shall be the name of an existing variable and the value of that variable shall be a valid representation of an integer (see 7.4) that specifies the maximum duration. Otherwise, there shall be no maximum duration.

NOTE An implementation of this Part of ISO/IEC 24709 may abandon the execution of a test that has lasted beyond a reasonable amount of time. Such a behavior would not be in violation of this sub-clause.

8.15.2.9 The invocation shall proceed regardless of the specified maximum duration, even if this is zero or negative. If it is zero or negative, the abstract test engine shall immediately determine that the maximum duration has been exceeded.

8.15.2.10 If the `setvar` attribute is present, the four following sub-clauses apply.

8.15.2.10.1 The value of the `setvar` attribute shall be a valid global or local variable name (see 7.2), shall not be an input parameter of the assertion being processed (see 8.3), and shall not begin with two consecutive LOW LINE characters ("_"). The variable may exist prior to the assignment or may be a new variable created by the assignment.

8.15.2.10.2 If the maximum duration is exceeded before the invocation returns, the variable shall be created (if it does not exist) and shall be set to "`true`".

8.15.2.10.3 If the invocation returns before the maximum duration is exceeded, then the variable shall be created (if it does not exist) and shall be set to "`false`".

8.15.2.10.4 The variable shall not be created (if it does not exist before the invocation) and shall not be assigned except as specified in the two previous sub-clauses.

8.15.2.11 When an activity or a function is invoked, the execution of the current activity shall be suspended until the invoked activity or function terminates or until the maximum duration has been exceeded.

8.15.2.12 Before the activity or function being invoked starts, all input parameters for which an `<input>` element was provided in the invocation (if any) shall be set from the variables or values provided in those `<input>` elements, as specified in 8.5.2.5. The remaining input parameters shall be assigned from an empty string.

8.15.2.13 If the invoked activity or function terminates without exceeding the maximum duration, the two following sub-clauses apply.

8.15.2.13.1 The values of each output parameter for which an `<output>` element was provided in the invocation (if any) shall be assigned to the variable referenced in the `<output>` element as specified in 8.6.2.3.

8.15.2.13.2 In case of function invocation, if there is a `<return>` element in the invocation, the return value shall be assigned to the variable referenced in the `<return>` element as specified in 8.7.2.3.

8.15.2.14 If the maximum duration is exceeded before the invoked function returns, the two following sub-clauses apply.

8.15.2.14.1 No assignment of output parameters and return value to variables shall be performed. In addition, if a variable corresponding to an output parameter or return value did not exist before the invocation, it shall not be created when the invoking activity resumes.

8.15.2.14.2 An implementation shall not (attempt to) interrupt the execution of the native function, even if it determines that the execution of the native function may last forever.

8.15.2.15 The execution of the invoking activity (if any) shall then resume (but see 8.9.2.17).

8.15.2.16 A native function that has exceeded the maximum duration may eventually return. When it returns, its native output parameters (if any) and return value shall be discarded.

8.15.3 Examples (non-normative)

```
<invoke activity="LoadAndAttach">
   <only_if var="myflag"/>
   <output name="BSP" setvar="BSP"/>
</invoke>

<invoke function="BioSPI_CreateTemplate">
   <input name="BSPHandle" var="BSP"/>
   <input name="CapturedBIR" var="inputbir"/>
   <input name="StoredTemplate" value=""/>
   <input name="Payload" value=""/>
   <output name="NewTemplate" setvar="template"/>
   <return setvar="return"/>
</invoke>
```

8.16 Element `<only_if>`

8.16.1 Syntax

8.16.1.1 This element shall have the following attribute:

— **var** (optional) **–** if this attribute is present, its value shall be a valid variable name (see 7.2).

8.16.1.2 This element shall have a content consisting of the following:

a) an optional **`<description>`** element – this element (if present) shall contain a description of the condition (a character string);

b) zero or more occurrences of any of the following elements in any order:

1) **`<and>`** – this element represents the logical operator AND, and is specified in 8.19;

2) **`<or>`** – this element represents the logical operator OR, and is specified in 8.20;

3) **`<xor>`** – this element represents the logical operator XOR, and is specified in 8.21;

4) **`<not>`** – this element represents the logical operator NOT, and is specified in 8.22;

5) **`<equal_to>`** – this element represents the numeric operator "equal_to", and is specified in 8.23;

6) **`<not_equal_to>`** – this element represents the numeric operator "not equal to", and is specified in 8.24;

7) **`<greater_than>`** – this element represents the numeric operator "greater than", and is specified in 8.25;

8) **`<greater_than_or_equal_to>`** – this element represents the numeric operator "greater than or equal to", and is specified in 8.26;

9) **`<less_than>`** – this element represents the numeric operator "less than", and is specified in 8.27;

10) **`<less_than_or_equal_to>`** – this element represents the numeric operator "less than or equal to", and is specified in 8.28;

11) **`<same_as>`** – this element represents the character-string operator "equals", and is specified in 8.29;

12) **<not_same_as>** – this element represents the character-string operator "not equal to", and is specified in 8.30;

13) **<existing>** – this element expresses the condition that a variable exists, and is specified in 8.31;

14) **<not_existing>** – this element expresses the condition that a variable does not exist, and is specified in 8.32.

8.16.1.3 If the **var** attribute is present, the content of the element shall be empty.

8.16.2 Semantics

8.16.2.1 This element represents a condition based on either a single variable or a combination of variables, values, logical operators, numeric operators, and other operators.

8.16.2.2 If the **var** attribute is present, its value shall be the name of an existing variable, and the value of that variable shall be a valid representation of a boolean (see 7.5). The result of the evaluation shall be the same as the value of the variable.

8.16.2.3 If there is no **var** attribute and there are no child elements, the result of the evaluation shall be "**true**".

8.16.2.4 If there are child elements, the result of the evaluation of each child element shall be a valid representation of a boolean (see 7.5). The result of the evaluation of the condition shall be "**true**" if all the child elements evaluate to "**true**", and shall be "**false**" otherwise.

8.16.2.5 The condition is evaluated during the processing of the parent element (see 8.12.2.3 and 8.15.2.2). If the condition evaluates to "**true**", then the containing element is processed normally (as though it did not have an **<only_if>** child element), otherwise its processing has no effect.

8.17 Element <wait_until>

8.17.1 Syntax

8.17.1.1 This element shall have the following attributes:

a) **timeout_value** (optional) – if this attribute is present, its value indicates the maximum duration of time (in milliseconds) allowed for the wait;

b) **timeout_var** (optional) – if this attribute is present, its value shall be a valid name of a variable (see 7.2), whose value indicates the maximum duration of time (in milliseconds) allowed for the wait;

c) **setvar** (optional) – if this attribute is present, its value shall be a valid variable name (see 7.2);

d) **var** (optional) – if this attribute is present, its value shall be a valid variable name (see 7.2).

8.17.1.2 At most one of the attributes **timeout_var** and **timeout_value** shall be present.

8.17.1.3 The **setvar** attribute shall not be present unless either the **timeout_value** attribute or the **timeout_var** attribute is present.

8.17.1.4 This element shall have a content consisting of the same child elements as specified for the element **<only_if>** (see 8.16).

8.17.1.5 If the **var** attribute is present, the content of the element shall be empty.

8.17.2 Semantics

8.17.2.1 This element specifies a suspension of the execution of the current activity until a certain condition is verified, or until a certain maximum duration of time has been reached.

8.17.2.2 The condition in the `<wait_until>` element is based on either a single variable or a combination of variables, values, logical operators, numeric operators, and other operators.

8.17.2.3 If the `var` attribute is present, its value shall be the name of an existing variable, and the value of that variable shall be a valid representation of a boolean (see 7.5). The result of the evaluation shall be the same as the value of the variable.

8.17.2.4 If there is no `var` attribute and there are no child elements, the result of the evaluation shall be "`true`".

8.17.2.5 If there are child elements, the result of the evaluation of each child element shall be a valid representation of a boolean (see 7.5). The result of the evaluation of the condition shall be "`true`" if all the child elements evaluate to "`true`", and shall be "`false`" otherwise.

8.17.2.6 If the `timeout_value` attribute is present, its value shall be a valid representation of an integer (see 7.4) that specifies the maximum duration allowed for the wait (in milliseconds). If the `timeout_var` attribute is present, its value shall be the name of an existing variable and the value of that variable shall be a valid representation of an integer (see 7.4) that specifies the maximum duration. Otherwise, the wait shall last (conceptually) for an indefinite time.

NOTE The NOTE to 8.15.2.8 applies.

8.17.2.7 Conceptually, the condition shall be evaluated repeatedly and continuously during the wait, until its result becomes "`true`" or until the maximum duration (if any) has been reached.

NOTE In practice, it is only necessary to evaluate the condition at discrete points in time, whenever there is a possibility that an incoming call made to the testing component, or even the passing of time, have affected the value of the condition. Implementors of this Part of ISO/IEC 24709 need to ensure that any transition of the condition from "`false`" to "`true`" during a wait is detected, even if the condition remains "`true`" for a very short time and then becomes "`false`" again.

8.17.2.8 The condition shall be evaluated at least once. If the first evaluation produces a value of "`true`", then the maximum duration shall not be considered exceeded.

NOTE This is required even if the maximum duration is zero or negative.

8.17.2.9 If the `setvar` attribute is present, the four following sub-clauses apply.

8.17.2.9.1 The value of the **setvar** attribute shall be a valid global or local variable name (see 7.2) , shall not be an input parameter of the assertion being processed (see 8.3), and shall not begin with two consecutive LOW LINE characters ("_"). The variable may exist prior to the assignment or may be a new variable created by the assignment.

8.17.2.9.2 If the maximum duration is exceeded while the condition is still "**false**", the variable shall be created (if it does not exist) and shall be set to "**true**".

8.17.2.9.3 If the condition is already "**true**" at the beginning of the wait or becomes "**true**" during the wait, then the variable shall be created (if it does not exist) and shall be set to "**false**".

8.17.2.9.4 The variable shall not be created (if it does not exist before the wait) and shall not be assigned except as specified in the two previous sub-clauses.

NOTE This implies that the variable cannot be initialized at the beginning of the wait.

8.18 Element `<assert_condition>`

8.18.1 Syntax

8.18.1.1 This element shall have the following attributes:

a) `response_if_true` (optional) – if this attribute is present, it shall have one of the values "`pass`", "`undecided`"; it indicates which conformity response shall be issued when the condition is "`true`"; the default is "`pass`";

b) `response_if_false` (optional) – if this attribute is present, it shall have one of the values "`fail`", "`undecided`"; it shall indicate which conformity response shall be issued when the condition is "`false`"; the default is "`fail`";

c) `break_if_false` (optional) – if this attribute is present, it shall have one of the values "`false`", "`true`"; it indicates whether the execution of the current activity shall be abandoned in case the condition is "`false`"; the default is "`false`";

d) `var` (optional) – if this attribute is present, its value shall be a valid variable name (see 7.2).

8.18.1.2 This element shall have a content consisting of the same child elements as specified for the element `<only_if>` (see 8.16).

8.18.1.3 If the `var` attribute is present, the content of the element shall be empty.

8.18.2 Semantics

8.18.2.1 This element specifies:

a) a condition relative to an aspect of the behavior of the implementation under test;

b) the conformity response ("`pass`", "`undecided`") to be issued when the condition is verified; and

c) the conformity response ("`fail`", "`undecided`") to be issued when the condition is not verified.

8.18.2.2 This feature enables the specification of conformance criteria by which one wants to state that a certain condition shall produce, say, a "`pass`" response but the opposite condition shall produce "`undecided`"; or by which one wants to state that a certain condition shall produce, say, a "`fail`" response but the opposite condition shall produce "`undecided`".

8.18.2.3 The `break_if_false` attribute indicates whether a result of "`false`" for the condition shall cause the abstract test engine to abandon (break) the execution of the current activity (see 8.9.2.18). It supports those common cases in which a series of actions – following the issuance of a certain conformity response – are meaningless or inappropriate.

8.18.2.4 The condition in the `<assert_condition>` element is based on either a single variable or a combination of variables, values, logical operators, numeric operators, and other operators.

8.18.2.5 If the `var` attribute is present, its value shall be the name of an existing variable, and the value of that variable shall be a valid representation of a boolean (see 7.5). The result of the evaluation shall be the same as the value of the variable.

8.18.2.6 If there is no `var` attribute and there are no child elements, the result of the evaluation shall be "`true`".

8.18.2.7 If there are child elements, the result of the evaluation of each child element shall be a valid representation of a boolean (see 7.5). The result of the evaluation of the condition shall be "**`true`**" if all the child elements evaluate to "**`true`**", and shall be "**`false`**" otherwise.

8.18.2.8 If the condition evaluates to "**`true`**", then the corresponding conformity response (see 8.18.2.1 b)) shall be issued as specified in clause 11.

8.18.2.9 If the condition evaluates to "**`false`**", then the corresponding conformity response (see 8.18.2.1 c)) shall be issued as specified in clause 11; in addition, if the **`break_if_false`** attribute is "**`true`**", the current activity shall be abandoned (see 8.9.2.18).

8.19 Element **`<and>`**

8.19.1 Syntax

8.19.1.1 This element shall have the following attributes:

a) **`var1`** (optional) – if this attribute is present, its value shall be a valid variable name (see 7.2);

b) **`var2`** (optional) – if this attribute is present, its value shall be a valid variable name (see 7.2).

8.19.1.2 This element shall have a content consisting of the same child elements as specified for the element **`<only_if>`** (see 8.16), but with the following constraint.

8.19.1.3 If either the **`var1`** attribute or the **`var2`** attribute is present, then both shall be present and the content of the element shall be empty, otherwise there shall be two or more child elements.

8.19.2 Semantics

8.19.2.1 This element represents a condition based on either two variables or a combination of variables, values, logical operators, numeric operators, and other operators. The condition is evaluated during the processing of the parent element.

8.19.2.2 This element represents a logical AND operator, whose operands may be provided either as attributes of this element or as child elements. If attributes are used, the number of operands shall be two, otherwise it can be two or more.

8.19.2.3 If the **`var1`** attribute is present, its value shall be the name of an existing variable, and the value of that variable shall be a valid representation of a boolean (see 7.5).

8.19.2.4 If the **`var2`** attribute is present, its value shall be the name of an existing variable, and the value of that variable shall be a valid representation of a boolean (see 7.5).

8.19.2.5 If the attributes **`var1`** and **`var2`** are present, the operands shall be the values of the two variables specified in these attributes, in order. Otherwise, the operands shall be the result of the evaluation of each child element, in order.

8.19.2.6 Each operand shall be a valid representation of a boolean (see 7.5). The result of the evaluation shall be "**`true`**" if the operands are all "**`true`**", otherwise it shall be "**`false`**".

8.20 Element `<or>`

8.20.1 Syntax

8.20.1.1 This element shall have the following attributes:

a) `var1` (optional) – if this attribute is present, its value shall be a valid variable name (see 7.2);

b) `var2` (optional) – if this attribute is present, its value shall be a valid variable name (see 7.2).

8.20.1.2 This element shall have a content consisting of the same child elements as specified for the element `<only_if>` (see 8.16), but with the following constraint.

8.20.1.3 If either the `var1` attribute or the `var2` attribute is present, then both shall be present and the content of the element shall be empty, otherwise there shall be two or more child elements.

8.20.2 Semantics

8.20.2.1 This element represents a condition based on either two variables or a combination of variables, values, logical operators, numeric operators, and other operators. The condition is evaluated during the processing of the parent element.

8.20.2.2 This element represents a logical OR operator, whose operands may be provided either as attributes of this element or as child elements. If attributes are used, the number of operands shall be two, otherwise it can be two or more.

8.20.2.3 If the `var1` attribute is present, its value shall be the name of an existing variable, and the value of that variable shall be a valid representation of a boolean (see 7.5).

8.20.2.4 If the `var2` attribute is present, its value shall be the name of an existing variable, and the value of that variable shall be a valid representation of a boolean (see 7.5).

8.20.2.5 If the attributes `var1` and `var2` are present, the operands shall be the values of the two variables specified in these attributes, in order. Otherwise, the operands shall be the result of the evaluation of each child element, in order.

8.20.2.6 Each operand shall be a valid representation of a boolean (see 7.5). The result of the evaluation shall be "`false`" if the operands are all "`false`", otherwise it shall be "`true`".

8.21 Element `<xor>`

8.21.1 Syntax

8.21.1.1 This element shall have the following attributes:

a) `var1` (optional) – if this attribute is present, its value shall be a valid variable name (see 7.2);

b) `var2` (optional) – if this attribute is present, its value shall be a valid variable name (see 7.2).

8.21.1.2 This element shall have a content consisting of the same child elements as specified for the element `<only_if>` (see 8.16), but with the following constraint.

8.21.1.3 If either the `var1` attribute or the `var2` attribute is present, then both shall be present and the content of the element shall be empty, otherwise there shall be exactly two child elements.

8.21.2 Semantics

8.21.2.1 This element represents a condition based on either two variables or a combination of variables, values, logical operators, numeric operators, and other operators. The condition is evaluated during the processing of the parent element.

8.21.2.2 This element represents a logical XOR operator, whose operands may be provided either as attributes of this element or as child elements. The number of operands shall be two.

8.21.2.3 If the `var1` attribute is present, its value shall be the name of an existing variable, and the value of that variable shall be a valid representation of a boolean (see 7.5).

8.21.2.4 If the `var2` attribute is present, its value shall be the name of an existing variable, and the value of that variable shall be a valid representation of a boolean (see 7.5).

8.21.2.5 If the attributes `var1` and `var2` are present, the operands shall be the values of the two variables specified in these attributes, in order. Otherwise, the operands shall be the result of the evaluation of each child element, in order.

8.21.2.6 Each operand shall be a valid representation of a boolean (see 7.5). The result of the evaluation shall be "`true`" if the two operands are different, otherwise it shall be "`false`".

8.22 Element `<not>`

8.22.1 Syntax

8.22.1.1 This element shall have the following attribute:

— **var** (optional) – if this attribute is present, its value shall be a valid variable name (see 7.2).

8.22.1.2 This element shall have a content consisting of the same child elements as specified for the element `<only_if>` (see 8.16), but with the following constraint.

8.22.1.3 If the `var` attribute is present, then the content of the element shall be empty, otherwise there shall be exactly one child element.

8.22.2 Semantics

8.22.2.1 This element represents a condition based on either a single variable or a combination of variables, values, logical operators, numeric operators, and other operators. The condition is evaluated during the processing of the parent element.

8.22.2.2 This element represents a logical NOT operator, whose operand may be provided either as an attribute of this element or as a child element. The number of operands shall be exactly one.

8.22.2.3 If the `var` attribute is present, its value shall be the name of an existing variable, and the value of that variable shall be a valid representation of a boolean (see 7.5).

8.22.2.4 If the `var` attribute is present, the operand shall be the value of the variable specified in this attribute. Otherwise, the operand shall be the result of the evaluation of the child element.

8.22.2.5 The operand shall be a valid representation of a boolean (see 7.5). The result of the evaluation shall be "false" if the operand is "true", and shall be "true" otherwise.

8.23 Element `<equal_to>`

8.23.1 Syntax

8.23.1.1 This element shall have the following attributes:

a) `var1` (optional) – if this attribute is present, its value shall be a valid variable name (see 7.2);

b) `var2` (optional) – if this attribute is present, its value shall be a valid variable name (see 7.2);

c) `value1` (optional) – if this attribute is present, its value shall be a valid representation of an integer;

d) `value2` (optional) – if this attribute is present, its value shall be a valid representation of an integer.

8.23.1.2 This element shall have a content consisting of the same child elements as specified for the element `<only_if>` (see 8.16), but with the following constraints.

8.23.1.3 At most one of the attributes `var1` and `value1` shall be present.

8.23.1.4 At most one of the attributes `var2` and `value2` shall be present.

8.23.1.5 If either `var1` or `value1` is present and either `var2` or `value2` is also present, then the content of the element shall be empty. If no attribute is present, then there shall be exactly two child elements. Otherwise, there shall be exactly one child element.

8.23.2 Semantics

8.23.2.1 This element represents a condition based on either two variables or a combination of variables, values, logical operators, numeric operators, and other operators. The condition is evaluated during the processing of the parent element.

8.23.2.2 This element represents an "equals" numeric operation, whose operands may be provided either as attributes of this element or as child elements. The number of operands shall be two.

8.23.2.3 If the `var1` attribute is present, its value shall be the name of an existing variable, and the value of that variable shall be a valid representation of an integer (see 7.4).

8.23.2.4 If the `var2` attribute is present, its value shall be the name of an existing variable, and the value of that variable shall be a valid representation of an integer (see 7.4).

8.23.2.5 If the **var1** attribute is present, then the first operand shall be the value of the variable specified in this attribute. If the attribute **value1** is present, then the first operand shall be the value of this attribute. Otherwise, the first operand shall be the result of the evaluation of the first (or only) child element.

8.23.2.6 If the **var2** attribute is present, then the second operand shall be the value of the variable specified in this attribute. If the attribute **value2** is present, then the second operand shall be the value of this attribute. Otherwise, the second operand shall be the result of the evaluation of the second (or only) child element.

8.23.2.7 Each operand shall be a valid representation of an integer (see 7.4). The result of the evaluation shall be "**true**" if the integer represented by the first operand is equal to the integer represented by the second, otherwise it shall be "**false**".

8.24 Element `<not_equal_to>`

8.24.1 Syntax

This element has the same syntax as the element `<equal_to>` (see 8.23).

8.24.2 Semantics

8.24.2.1 This element has the same semantics as the element `<equal_to>`, except for the following.

8.24.2.2 This element represents a "not equal to" numeric operation. The result of the evaluation shall be "`true`" if the integer represented by the first operand is not equal to the integer represented by the second, otherwise it shall be "`false`".

8.25 Element `<greater_than>`

8.25.1 Syntax

This element has the same syntax as the element `<equal_to>` (see 8.23).

8.25.2 Semantics

8.25.2.1 This element has the same semantics as the element `<equal_to>`, except for the following.

8.25.2.2 This element represents a "greater than" numeric operation. The result of the evaluation shall be "`true`" if the integer represented by the first operand is greater than the integer represented by the second, otherwise it shall be "`false`".

8.26 Element `<greater_than_or_equal_to>`

8.26.1 Syntax

This element has the same syntax as the element `<equal_to>` (see 8.23).

8.26.2 Semantics

8.26.2.1 This element has the same semantics as the element `<equal_to>`, except for the following.

8.26.2.2 This element represents a "greater than or equal to" numeric operation. The result of the evaluation shall be "`true`" if the integer represented by the first operand is greater than or equal to the integer represented by the second, otherwise it shall be "`false`".

8.27 Element `<less_than>`

8.27.1 Syntax

This element has the same syntax as the element `<equal_to>` (see 8.23).

8.27.2 Semantics

8.27.2.1 This element has the same semantics as the element `<equal_to>`, except for the following.

8.27.2.2 This element represents a "less than" numeric operation. The result of the evaluation shall be "`true`" if the integer represented by the first operand is less than the integer represented by the second, otherwise it shall be "`false`".

8.28 Element `<less_than_or_equal_to>`

8.28.1 Syntax

This element has the same syntax as the element `<equal_to>` (see 8.23).

8.28.2 Semantics

8.28.2.1 This element has the same semantics as the element `<equal_to>`, except for the following.

8.28.2.2 This element represents a "less than or equal to" numeric operation. The result of the evaluation shall be "`true`" if the integer represented by the first operand is less than or equal to the integer represented by the second, otherwise it shall be "`false`".

8.29 Element `<same_as>`

8.29.1 Syntax

8.29.1.1 This element shall have the following attributes:

a) `var1` (optional) – if this attribute is present, its value shall be a valid variable name (see 7.2);

b) `var2` (optional) – if this attribute is present, its value shall be a valid variable name (see 7.2);

c) `value1` (optional);

d) `value2` (optional).

8.29.1.2 This element shall have a content consisting of the same child elements as specified for the element `<only_if>` (see 8.16), but with the following constraints.

8.29.1.3 At most one of the attributes `var1` and `value1` shall be present.

8.29.1.4 At most one of the attributes `var2` and `value2` shall be present.

8.29.1.5 If either `var1` or `value1` is present and either `var2` or `value2` is also present, then the content of the element shall be empty. If no attribute is present, then there shall be exactly two child elements. Otherwise, there shall be exactly one child element.

8.29.2 Semantics

8.29.2.1 This element represents a condition based on either two variables or a combination of variables, values, logical operators, numeric operators, and other operators. The condition is evaluated during the processing of the parent element.

8.29.2.2 This element represents a character-string "equals" operation, whose operands may be provided either as attributes of this element or as child elements. The number of operands shall be two.

8.29.2.3 If the `var1` attribute is present, its value shall be the name of an existing variable.

8.29.2.4 If the `var2` attribute is present, its value shall be the name of an existing variable.

8.29.2.5 If the **var1** attribute is present, then the first operand shall be the value of the variable specified in this attribute. If the attribute **value1** is present, then the first operand shall be the value of this attribute. Otherwise, the first operand shall be the result of the evaluation of the first (or only) child element.

8.29.2.6 If the **var2** attribute is present, then the second operand shall be the value of the variable specified in this attribute. If the attribute **value2** is present, then the second operand shall be the value of this attribute. Otherwise, the second operand shall be the result of the evaluation of the second (or only) child element.

8.29.2.7 The result of the evaluation shall be "`true`" if the first operand is equal to the second operand character-by-character, otherwise it shall be "`false`".

NOTE This element is normally used to compare two character strings or two booleans.

8.30 Element `<not_same_as>`

8.30.1 Syntax

This element has the same syntax as the element `<same_as>` (see 8.29).

8.30.2 Semantics

8.30.2.1 This element has the same semantics as the element `<same_as>`, except for the following.

8.30.2.2 This element represents a "not equal to" character-string operation. The result of the evaluation shall be "`true`" if the first operand is not equal to the second operand character-by-character, otherwise it shall be "`false`".

8.31 Element `<existing>`

8.31.1 Syntax

8.31.1.1 This element shall have the following attribute:

— **var** (mandatory) – the value of this attribute shall be a valid variable name (see 7.2).

8.31.1.2 The content of this element shall be empty.

8.31.2 Semantics

8.31.2.1 This element represents a condition based on a variable. The condition is evaluated during the processing of the parent element.

8.31.2.2 If the value of the `var` attribute is the name of an existing global or local variable, the result of the evaluation shall be "`true`", otherwise it shall be "`false`".

8.32 Element `<not_existing>`

8.32.1 Syntax

This element has the same syntax as the element `<existing>` (see 8.31).

8.32.2 Semantics

8.32.2.1 This element has the same semantics as the element **`<existing>`**, except for the following.

8.32.2.2 If the value of the **`var`** attribute is the name of an existing global or local variable, the result of the evaluation shall be "**`false`**", otherwise it shall be "**`true`**".

9 Standard interface functions

9.1 General

9.1.1 This clause 9 specifies the use of the standard BioAPI interface functions of BioAPI in conformance testing. Subcluases 9.2 specifies parameter groups with more specific restrictions which are to be considered in addition to the parameter constraints specified in the subsequent subcluases. Each major subclause of this clause (from subclause 9.3 to the end of the clause) specifies the following properties of a standard BioAPI interface function:

1) function invocation scheme - the name of the standard BioAPI interface function and the names of its input and output parameters to be used in an invocation of the function;

2) constraints on the parameters;

3) function invocation input - setting of the native input parameters of the underlying function from the input parameters of a function invocation;

4) function invocation output - setting of the output parameters and return value of a function invocation from the native output parameters and return value of the underlying function;

5) bound activity invocation input - setting of the input parameters of an activity bound to the standard BioAPI interface function from the native input parameters of an incoming call;

6) bound activity invocation output - setting of the native output parameters and return value of the incoming call from the output parameters of an activity bound to the standard BioAPI interface function.

7) default output - setting of the native output parameters and return value of an incoming call when there is no activity bound to a standard BioAPI interface function.

NOTE Some variables and parameter groups of the assertion language support a wider range of values than are permitted by BioAPI for the corresponding native variables and function parameters. The purpose of this is to allow the creation of assertions that assess the response of the IUT to values outside the permitted range and assess the validity of the responses from the IUT.

9.1.2 The "outbound value" (for an input parameter of a function) is:

a) if the corresponding **`<input>`** element contains a **`value`** attribute, the value of that attribute;

b) if the corresponding **`<input>`** element contains a **`var`** attribute, the value of the variable whose name is the value of that attribute.

9.1.3 The "inbound value" (for an output parameter of a function) is the value to be assigned to the variable whose name is the value of the **`setvar`** attribute of the corresponding **`<output>`** element (if this element is provided in the invocation).

9.1.4 The "return value" of a function is the value to be assigned to the variable whose name is the value of the **`setvar`** attribute of the **`<return>`** element (if this element is provided in the invocation).

9.1.5 The "inbound value" (for an input parameter of a bound activity) is:

a) if the corresponding `<input>` element contains a `value` attribute, the value of that attribute;

b) if the corresponding `<input>` element contains a `var` attribute, the value of the variable whose name is the value of that attribute.

9.1.6 The "outbound value" (for an output parameter of a bound activity) is the value to be assigned to the variable whose name is the value of the `setvar` attribute of the corresponding `<output>` element (if this element is provided in the invocation).

9.1.7 In all function invocations, the return (inbound) value shall be set to the canonical representation of the integer in the range 0 to 4294967295 returned by the underlying function (see 7.4.3).

9.1.8 In each conformance testing model, standard BioAPI interface functions shall be used as specified in Table 1.

Table 1 — Use of standard BioAPI interface functions in conformance testing models

	Conformance testing model for BioAPI applications (see 6.2.5.1)	**Conformance testing model for BioAPI frameworks (see 6.2.5.2)**	**Conformance testing model for BioAPI BSPs (see 6.2.5.3)**
Standard BioAPI interface functions of the BioAPI interface	the application-testing framework shall expose the function the normal framework exposes the function the application under test may call the function exposed by the application-testing framework any incoming calls to the function exposed by the application-testing framework shall be relayed to the same function exposed by the normal framework no other ways of calling the function are supported	the framework under test exposes the function neither the framework-testing application nor the framework-testing BSP shall expose the function the framework-testing application is able to call the function exposed by the framework under test no other ways of calling the function are supported	the BSP-testing application shall not expose the function no ways of calling the function are supported

Standard BioAPI interface functions of the application callback interface	the application under test may expose the function the application-testing framework shall expose the function the normal framework may call the function exposed by the application-testing framework any incoming calls to the function exposed by the application-testing framework shall be relayed to the same function exposed by the application under test (if any) no other ways of calling the function are supported	the framework-testing application shall expose the function the framework under test may call the function exposed by the framework-testing application no other ways of calling the function are supported	the BSP-testing application shall not expose the function no ways of calling the function are supported
Framework callback interface	the normal framework exposes the function the application-testing framework shall not expose the function the normal BSP may call the function exposed by the normal framework no other ways of calling the function are supported	the framework under test exposes the function neither the framework-testing application nor the framework-testing BSP shall expose the function the framework-testing BSP is able to call the function exposed by the framework under test no other ways of calling the function are supported	the BSP-testing application shall expose the function the BSP under test may call the function exposed by the BSP-testing application no other ways of calling the function are supported
Standard BioAPI interface functions of the BioSPI interface	the normal BSP may expose the function the application-testing framework shall not expose the function the normal framework may call the function exposed by the normal BSP no other ways of calling the function are supported	the framework-testing BSP shall expose the function the framework-testing application shall not expose the function the framework under test may call the function exposed by the framework-testing BSP no other ways of calling the function are supported	the BSP under test may expose the function the BSP-testing application is able to call the function exposed by the BSP under test no other ways of calling the function are supported

9.1.9 The BioAPI interface consists of the following standard BioAPI interface functions:

- **BioAPI_Init**
- **BioAPI_Terminate**
- **BioAPI_GetFrameworkInfo**
- **BioAPI_EnumBSPs**
- **BioAPI_BSPLoad**
- **BioAPI_BSPUnload**
- **BioAPI_BSPAttach**
- **BioAPI_BSPDetach**
- **BioAPI_QueryUnits**
- **BioAPI_EnumBFPs**
- **BioAPI_QueryBFPs**
- **BioAPI_ControlUnit**
- **BioAPI_FreeBIRHandle**
- **BioAPI_GetBIRFromHandle**
- **BioAPI_GetHeaderFromHandle**
- **BioAPI_EnableEvents**
- **BioAPI_SetGUICallbacks**
- **BioAPI_Capture**
- **BioAPI_CreateTemplate**
- **BioAPI_Process**
- **BioAPI_ProcessWithAuxBIR**
- **BioAPI_VerifyMatch**
- **BioAPI_IdentifyMatch**
- **BioAPI_Enroll**
- **BioAPI_Verify**
- **BioAPI_Identify**
- **BioAPI_Import**
- **BioAPI_PresetIdentifyPopulation**
- **BioAPI_DbOpen**
- **BioAPI_DbClose**
- **BioAPI_DbCreate**
- **BioAPI_DbDelete**
- **BioAPI_DbSetMarker**
- **BioAPI_DbFreeMarker**
- **BioAPI_DbStoreBIR**
- **BioAPI_DbGetBIR**
- **BioAPI_DbGetNextBIR**
- **BioAPI_DbDeleteBIR**
- **BioAPI_SetPowerMode**
- **BioAPI_SetIndicatorStatus**
- **BioAPI_GetIndicatorStatus**
- **BioAPI_CalibrateSensor**
- **BioAPI_Cancel**
- **BioAPI_Free**

- **BioAPI_Util_InstallBSP**
- **BioAPI_Util_InstallBFP**

9.1.10 The BioSPI interface consists of the following standard BioAPI interface functions:

- **BioSPI_BSPLoad**
- **BioSPI_BSPUnload**
- **BioSPI_BSPAttach**
- **BioSPI_BSPDetach**
- **BioSPI_QueryUnits**
- **BioSPI_QueryBFPs**
- **BioSPI_ControlUnit**
- **BioSPI_FreeBIRHandle**
- **BioSPI_GetBIRFromHandle**
- **BioSPI_GetHeaderFromHandle**
- **BioSPI_EnableEvents**
- **BioSPI_SetGUICallbacks**
- **BioSPI_Capture**
- **BioSPI_CreateTemplate**
- **BioSPI_Process**
- **BioSPI_ProcessWithAuxBIR**
- **BioSPI_VerifyMatch**
- **BioSPI_IdentifyMatch**
- **BioSPI_Enroll**
- **BioSPI_Verify**
- **BioSPI_Identify**
- **BioSPI_Import**
- **BioSPI_PresetIdentifyPopulation**
- **BioSPI_DbOpen**
- **BioSPI_DbClose**
- **BioSPI_DbCreate**
- **BioSPI_DbDelete**
- **BioSPI_DbSetMarker**
- **BioSPI_DbFreeMarker**
- **BioSPI_DbStoreBIR**
- **BioSPI_DbGetBIR**
- **BioSPI_DbGetNextBIR**
- **BioSPI_DbDeleteBIR**
- **BioSPI_SetPowerMode**
- **BioSPI_SetIndicatorStatus**
- **BioSPI_GetIndicatorStatus**

- **BioSPI_CalibrateSensor**
- **BioSPI_Cancel**
- **BioSPI_Free**

9.1.11 The application callback interface consists of the following standard BioAPI interface functions:

- **BioAPI_EventHandler**
- **BioAPI_GUI_STATE_CALLBACK**
- **BioAPI_GUI_STREAMING_CALLBACK**

9.1.12 The framework callback interface consists of the following standard BioAPI interface functions:

- **BioSPI_EventHandler**
- **BioSPI_GUI_STATE_CALLBACK**
- **BioSPI_GUI_STREAMING_CALLBACK**
- **BioSPI_BFP_ENUMERATION_HANDLER**
- **BioSPI_MEMORY_FREE_HANDLER**

NOTE In ISO/IEC 19784-1, the standard BioAPI interface functions of the framework callback interface do not have function names, but are specified as function pointer types (which have names). In this Part of ISO/IEC 24709, these standard BioAPI interface functions are given names, both for ease of reference in the clauses of this Part of ISO/IEC 24709 and for use in the assertion language.

9.2 Parameter groups

The following subclauses specify groups of parameters (of functions or activities). Each group is referenced either by one or more subsequent groups or by one or more subclauses of this clause 9 relative to standard BioAPI interface functions.

9.2.1 Parameter group "Biometric type"

9.2.1.1 The parameter group "Biometric type" consists of the following parameters:

- **TypeMultiple**
- **TypeFacialFeatures**
- **TypeVoice**
- **TypeFingerprint**
- **TypeIris**
- **TypeRetina**
- **TypeHandGeometry**
- **TypeSignatureDynamics**
- **TypeKeystrokeDynamics**
- **TypeLipMovement**
- **TypeThermalFaceImage**
- **TypeThermalHandImage**
- **TypeGait**
- **TypeOther**
- **TypePassword**

9.2.1.2 This parameter group supports the representation of the values of the native type **`BioAPI_BIR_BIOMETRIC_TYPE`** (which is an integer type).

9.2.1.3 An outbound value of each of these parameters shall be either a valid representation of a boolean (see 7.5), or an empty string. An inbound value shall be the canonical representation of a boolean (see 7.5.2).

9.2.1.4 The value (say, *V*) of the native type **`BioAPI_BIR_BIOMETRIC_TYPE`** represented by an outbound value of this parameter group shall be determined as follows:

1) the value *V* shall be initially set to zero;

2) for each member of the parameter group whose outbound value is "**`true`**", an integer shall be added to *V* according to the following table.

`TypeMultiple`	$1=2^0$
`TypeFacialFeatures`	$2=2^1$
`TypeVoice`	$4=2^2$
`TypeFingerprint`	$8=2^3$
`TypeIris`	$16=2^4$
`TypeRetina`	$32=2^5$
`TypeHandGeometry`	$64=2^6$
`TypeSignatureDynamics`	$128=2^7$
`TypeKeystrokeDynamics`	$256=2^8$
`TypeLipMovement`	$512=2^9$
`TypeThermalFaceImage`	$1024=2^{10}$
`TypeThermalHandImage`	$2048=2^{11}$
`TypeGait`	$4096=2^{12}$
`TypeOther`	$1073741824=2^{30}$
`TypePassword`	$2147483648=2^{31}$

NOTE If TypeMultiple is set to false, only one of the other types may be set to true.

9.2.1.5 Given an integer (say, *V*) that is a value of the native type `BioAPI_BIR_BIOMETRIC_TYPE`, the inbound value of this parameter group that canonically represents *V* shall be determined as follows:

1) the integer *V* shall be decomposed into a sum of powers of two, each occurring at most once;

2) for each power of two listed in the table above that occurs in the decomposition of *V*, the inbound value of the parameter in the corresponding row of the table shall be "`true`";

3) the inbound value of all the remaining parameters (if any) shall be "`false`".

4) an even value of V not being a single power of two indicates an invalid value.

9.2.2 Parameter group "Operations"

9.2.2.1 The parameter group "Operations" consists of the following parameters:

- **OperationEnableEvents**
- **OperationSetGUICallbacks**
- **OperationCapture**
- **OperationCreateTemplate**
- **OperationProcess**
- **OperationProcessWithAuxBIR**
- **OperationVerifyMatch**
- **OperationIdentifyMatch**
- **OperationEnroll**
- **OperationVerify**
- **OperationIdentify**
- **OperationImport**
- **OperationPresetIdentifyPopulation**
- **OperationDatabaseOperations**
- **OperationSetPowerMode**
- **OperationSetIndicatorStatus**
- **OperationGetIndicatorStatus**
- **OperationCalibrateSensor**
- **OperationUtilities**
- **OperationQueryUnits**
- **OperationQueryBFPs**

9.2.2.2 This parameter group supports the representation of the values of the native type `BioAPI_OPERATIONS_MASK` (which is an integer type).

9.2.2.3 An outbound value of each of these parameters shall be either a valid representation of a boolean (see 7.5) or an empty string. An inbound value shall be the canonical representation of a boolean (see 7.5.2).

9.2.2.4 The value (say, *V*) of the native type `BioAPI_OPERATIONS_MASK` represented by an outbound value of this parameter group shall be determined as follows:

1) the value *V* shall be initially set to zero;

2) for each member of the parameter group whose outbound value is "`true`", an integer shall be added to *V* according to the following table.

`OperationEnableEvents`	$1=2^0$
`OperationSetGUICallbacks`	$2=2^1$
`OperationCapture`	$4=2^2$
`OperationCreateTemplate`	$8=2^3$
`OperationProcess`	$16=2^4$
`OperationProcessWithAuxBIR`	$32=2^5$
`OperationVerifyMatch`	$64=2^6$
`OperationIdentifyMatch`	$128=2^7$
`OperationEnroll`	$256=2^8$
`OperationVerify`	$512=2^9$
`OperationIdentify`	$1024=2^{10}$
`OperationImport`	$2048=2^{11}$
`OperationPresetIdentifyPopulation`	$4096=2^{12}$
`OperationDatabaseOperations`	$8192=2^{13}$
`OperationSetPowerMode`	$16384=2^{14}$
`OperationSetIndicatorStatus`	$32768=2^{15}$
`OperationGetIndicatorStatus`	$65536=2^{16}$
`OperationCalibrateSensor`	$131072=2^{17}$
`OperationUtilities`	$262144=2^{18}$
`OperationQueryUnits`	$1048576=2^{20}$

`OperationQueryBFPs`	$2097152=2^{21}$
`OperationControlUnit`	$4194304=2^{22}$

9.2.2.5 Given an integer (say, *V*) that is a value of the native type BioAPI_`OPERATIONS_MASK`, the inbound value of this parameter group that canonically represents *V* shall be determined as follows:

1) the integer *V* shall be decomposed into a sum of powers of two, each occurring at most once;

2) for each power of two listed in the table above that occurs in the decomposition of *V*, the inbound value of the parameter in the corresponding row of the table shall be "`true`";

3) the inbound value of all the remaining parameters (if any) shall be "`false`".

9.2.3 Parameter group "Options"

9.2.3.1 The parameter group "Options" consists of the following parameters:

- **OptionRaw**
- **OptionQualityRaw**
- **OptionQualityIntermediate**
- **OptionQualityProcessed**
- **OptionAppGUI**
- **OptionSourcePresent**
- **OptionPayload**
- **OptionBIRSign**
- **OptionBIREncrypt**
- **OptionTemplateUpdate**
- **OptionAdaptation**
- **OptionBinning**
- **OptionSelfContainedDevice**
- **OptionMOC**
- **OptionSubtypeToCapture**
- **OptionSensorBFP**
- **OptionArchiveBFP**
- **OptionMatchingBFP**
- **OptionProcessingBFP**
- **OptionCoarseScores**

9.2.3.2 This parameter group supports the representation of the values of the native type `BioAPI_OPTIONS_MASK` (which is an integer type).

9.2.3.3 An outbound value of each of these parameters shall be either a valid representation of a boolean (see 7.5) or an empty string. An inbound value shall be the canonical representation of a boolean (see 7.5.2).

9.2.3.4 The value (say, *V*) of the native type `BioAPI_OPTIONS_MASK` represented by an outbound value of this parameter group shall be determined as follows:

1) the value *V* shall be initially set to zero;

2) for each member of the parameter group whose outbound value is "`true`", an integer shall be added to *V* according to the following table.

`OptionRaw`	$1=2^0$
`OptionQualityRaw`	$2=2^1$
`OptionQualityIntermediate`	$4=2^2$
`OptionQualityProcessed`	$8=2^3$
`OptionAppGUI`	$16=2^4$
`OptionSourcePresent`	$64=2^6$
`OptionPayload`	$128=2^7$
`OptionBIRSign`	$256=2^8$
`OptionBIREncrypt`	$512=2^9$
`OptionTemplateUpdate`	$1024=2^{10}$
`OptionAdaptation`	$2048=2^{11}$
`OptionBinning`	$4096=2^{12}$
`OptionSelfContainedDevice`	$8192=2^{13}$
`OptionMOC`	$16384=2^{14}$
`OptionSubtypeToCapture`	$32768=2^{15}$
`OptionSensorBFP`	$65536=2^{16}$
`OptionArchiveBFP`	$131072=2^{17}$
`OptionMatchingBFP`	$262144=2^{18}$
`OptionProcessingBFP`	$524288=2^{19}$
`OptionCoarseScores`	$1048576=2^{20}$

9.2.3.5 Given an integer (say, *V*) that is a value of the native type BioAPI_OPTIONS_MASK, the inbound value of this parameter group that canonically represents *V* shall be determined as follows:

1) the integer *V* shall be decomposed into a sum of powers of two, each occurring at most once;

2) for each power of two listed in the table above that occurs in the decomposition of *V*, the inbound value of the parameter in the corresponding row of the table shall be "`true`";

3) the inbound value of all the remaining parameters (if any) shall be "`false`".

9.2.4 Parameter group "Events"

9.2.4.1 The parameter group "Events" consists of the following parameters:

- **EventNotifyInsert**
- **EventNotifyRemove**
- **EventNotifyFault**
- **EventNotifySourcePresent**
- **EventNotifySourceRemoved**

9.2.4.2 This parameter group supports the representation of the values of the native type `BioAPI_EVENT_MASK` (which is an integer type).

9.2.4.3 An outbound value of each of these parameters shall be either a valid representation of a boolean (see 7.5) or an empty string. An inbound value shall be the canonical representation of a boolean (see 7.5.2).

9.2.4.4 The value (say, *V*) of the native type `BioAPI_EVENT_MASK` represented by an outbound value of this parameter group shall be determined as follows:

1) the value *V* shall be initially set to zero;

2) for each member of the parameter group whose outbound value is "`true`", an integer shall be added to *V* according to the following table.

`EventNotifyInsert`	$1=2^0$
`EventNotifyRemove`	$2=2^1$
`EventNotifyFault`	$4=2^2$
`EventNotifySourcePresent`	$8=2^3$
`EventNotifySourceRemoved`	$16=2^4$

9.2.4.5 Given an integer (say, *V*) that is a value of the native type `BioAPI_EVENT_MASK`, the inbound value of this parameter group that canonically represents *V* shall be determined as follows.

1) the integer *V* shall be decomposed into a sum of powers of two, each occurring at most once;

2) for each power of two listed in the table above that occurs in the decomposition of *V*, the inbound value of the parameter in the corresponding row of the table shall be "`true`";

3) the inbound value of all the remaining parameters (if any) shall be "`false`".

9.2.5 Parameter group "Biometric data type"

9.2.5.1 The parameter group "Biometric data type" consists of the following parameters:

- **ProcessedLevel**
- **Encrypted**
- **Signed**
- **IndexPresent**

9.2.5.2 This parameter group supports the representation of the values of the native type `BioAPI_BIR_DATA_TYPE` (which is an integer type).

9.2.5.3 The members of this parameter group shall have inbound and outbound values as specified in the two following subclauses.

9.2.5.3.1 An outbound value of `ProcessedLevel` shall be either a valid representation of an integer (see 7.4) in the range 0 to 15, or an empty string. An inbound value shall be the canonical representation of an integer (see 7.4.3) in the same range.

9.2.5.3.2 An outbound value of `Encrypted`, `Signed`, and `IndexPresent` shall be either a valid representation of a boolean (see 7.5) or an empty string. An inbound value shall be the canonical representation of a boolean (see 7.5.2).

9.2.5.4 The value (say, *V*) of the native type `BioAPI_BIR_DATA_TYPE` represented by an outbound value of this parameter group shall be determined as follows:

1) the value *V* shall be initially set to the integer represented by the outbound value of `ProcessedLevel` (or zero, if the outbound value is an empty string);

2) for each member of the parameter group (apart from `ProcessedLevel`) whose outbound value is "`true`", an integer shall be added to *V* according to the following table.

`Encrypted`	$16=2^4$
`Signed`	$32=2^5$
`IndexPresent`	$128=2^7$

9.2.5.5 Given an integer (say, *V*) that is a value of the native type `BioAPI_BIR_DATA_TYPE`, the inbound value of this parameter group that canonically represents *V* shall be determined as follows:

1) the integer *V* shall be decomposed into a sum of:

   i) an integer less than 16;

   ii) the value 16;

   iii) the value 32; and

   iv) the value 128.

each occurring at most once;

2) for each power of two listed in the table above (16, 32, and 128) that occurs in the decomposition of *V*, the inbound value of the parameter in the corresponding row of the table shall be "`true`";

3) the inbound value of `ProcessedLevel` shall be the canonical representation of the integer in 1) i) above (see 7.4.3);

4) the inbound value of all remaining parameters (if any) shall be "`false`".

9.2.6 Parameter group "Biometric subtype"

9.2.6.1 The parameter group "Biometric subtype" consists of the following parameters:

- **Left**
- **Right**
- **Thumb**
- **PointerFinger**
- **MiddleFinger**
- **RingFinger**
- **LittleFinger**
- **Multiple**

9.2.6.2 This parameter group supports the representation of the values of the native type `BioAPI_BIR_SUBTYPE` (which is an integer type).

9.2.6.3 An outbound value of each of these parameters shall be either a valid representation of a boolean (see 7.5) or an empty string. An inbound value shall be the canonical representation of a boolean (see 7.5.2).

9.2.6.4 The value (say, *V*) of the native type `BioAPI_BIR_SUBTYPE` represented by an outbound value of this parameter group shall be determined as follows:

1) the value *V* shall be initially set to zero;

2) for each member of the parameter group whose outbound value is "`true`", an integer shall be added to *V* according to the following table.

`Left`	$1=2^0$
`Right`	$2=2^1$
`Thumb`	$4=2^2$
`PointerFinger`	$8=2^3$
`MiddleFinger`	$16=2^4$
`RingFinger`	$32=2^5$
`LittleFinger`	$64=2^6$
`Multiple`	$128=2^7$

9.2.6.5 Given an integer (say, *V*) that is a value of the native type `BioAPI_BIR_SUBTYPE`, the inbound value of this parameter group that canonically represents *V* shall be determined as follows:

1) the integer *V* shall be decomposed into a sum of powers of two, each occurring at most once;

2) for each power of two listed in the table above that occurs in the decomposition of *V*, the inbound value of the parameter in the corresponding row of the table shall be "`true`";

3) the inbound value of all the remaining parameters (if any) shall be "`false`".

9.2.7 Parameter group "Date"

9.2.7.1 The parameter group "Date" consists of the following parameters:

- **Year**
- **Month**
- **Day**

9.2.7.2 This parameter group supports the representation of the values of the native type `BioAPI_DATE`.

9.2.7.3 The members of this parameter group shall have inbound and outbound values as specified in the two following subclauses.

9.2.7.3.1 An outbound value of `Year` shall be either a valid representation of an integer (see 7.4) in the range 0 to 65535, or an empty string. An inbound value shall be the canonical representation of an integer (see 7.4.3) in the same range.

9.2.7.3.2 An outbound value of `Month` and `Day` shall be either a valid representation of an integer (see 7.4) in the range 0 to 255, or an empty string. An inbound value shall be the canonical representation of an integer (see 7.4.3) in the same range.

9.2.7.4 The value (say, *V*) of the native type `BioAPI_DATE` represented by an outbound value of this parameter group shall be determined as follows: The integer represented by the outbound value of `Year`, `Month`, and `Day` (or zero if the outbound value is an empty string) shall be written to the fields `Year`, `Month`, and `Day` (respectively) of the field `Date` of *V*.

9.2.7.5 Given a value (say, *V*) of the native type `BioAPI_DATE`, the inbound value of this parameter group that canonically represents *V* shall be determined as follows: The inbound value of `Year`, `Month`, and `Day` shall be the canonical representation of the integer (see 7.4.3) in the fields `Year`, `Month`, and `Day` (respectively) of the field `Date` of *V*.

9.2.8 Parameter group "Date and time"

9.2.8.1 The parameter group "Date and time" consists of the following parameters:

- **Year**
- **Month**
- **Day**
- **Hour**
- **Minute**
- **Second**

9.2.8.2 This parameter group supports the representation of the values of the native type `BioAPI_DTG`.

9.2.8.3 The members of this parameter group shall have inbound and outbound values as specified in the two following subclauses.

9.2.8.3.1 An outbound value of `Year` shall be either a valid representation of an integer (see 7.4) in the range 0 to 65535, or an empty string. An inbound value shall be the canonical representation of an integer (see 7.4.3) in the same range.

9.2.8.3.2 An outbound value of `Month`, `Day`, `Hour`, `Minute`, and `Second` shall be either a valid representation of an integer (see 7.4) in the range 0 to 255, or an empty string. An inbound value shall be the canonical representation of an integer (see 7.4.3) in the same range.

9.2.8.4 The value (say, *V*) of the native type `BioAPI_DTG` represented by an outbound value of this parameter group shall be determined as specified in the two following subclauses.

9.2.8.4.1 The integer represented by the outbound value of `Year`, `Month`, and `Day` (or zero if the outbound value is an empty string) shall be written to the fields `Year`, `Month`, and `Day` (respectively) of the field `Date` of *V*.

9.2.8.4.2 The integer represented by the outbound value of `Hour`, `Minute`, and `Second` (or zero if the outbound value is an empty string) shall be written to the fields `Hour`, `Minute`, and `Second` (respectively) of the field `Time` of *V*.

9.2.8.5 Given a value (say, *V*) of the native type `BioAPI_DTG`, the inbound value of this parameter group that canonically represents *V* shall be determined as specified in the two following subclauses.

9.2.8.5.1 The inbound value of `Year`, `Month`, and `Day` shall be the canonical representation of the integer (see 7.4.3) in the fields `Year`, `Month`, and `Day` (respectively) of the field `Date` of *V*.

9.2.8.5.2 The inbound value of `Hour`, `Minute`, and `Second` shall be the canonical representation of the integer (see 7.4.3) in the fields `Hour`, `Minute`, and `Second` (respectively) of the field `Time` of *V*.

9.2.9 Parameter group "Framework schema"

9.2.9.1 The parameter group "Framework schema" consists of the following parameters:

- **FrameworkUuid**
- **Description**
- **Path**
- **SpecVersion**
- **ProductVersion**
- **Vendor**
- **FwPropertyId**
- **FwProperty**

9.2.9.2 This parameter group supports the representation of the values of the native type `BioAPI_FRAMEWORK_SCHEMA`.

9.2.9.3 The members of this parameter group shall have inbound and outbound values as specified in the five following subclauses.

9.2.9.3.1 An outbound value of `FrameworkUuid` and `FwPropertyId` shall be either a valid representation of a UUID (see 7.6), or an empty string. An inbound value shall be the canonical representation of a UUID (see 7.6.3).

9.2.9.3.2 An outbound value of `SpecVersion` shall be either a valid representation of an integer in the range 0 to 255, or an empty string. An inbound value shall be the canonical representation of an integer in the same range.

9.2.9.3.3 An outbound or inbound value of `ProductVersion`, `Vendor`, and `Description` shall be a character string whose UTF-8 encoding is no longer than 268 octets, and which does not contain any NUL (0) characters.

9.2.9.3.4 An outbound value of `FwProperty` shall be a valid representation of an octet string (see 7.7). An inbound value shall be the canonical representation of an octet string (see 7.7.2).

9.2.9.3.5 An outbound or inbound value of `Path` shall be a character string that does not contain any NUL (0) characters.

9.2.9.4 The value (say, *V*) of the native type `BioAPI_FRAMEWORK_SCHEMA` represented by an outbound value of this parameter group shall be determined as specified in the five following subclauses.

9.2.9.4.1 The UUID represented by the outbound value of `FrameworkUuid` and `FwPropertyId` (or the UUID "`00000000-0000-0000-0000-000000000000`" if the outbound value is an empty string) shall be written to the field of *V* with the same name.

9.2.9.4.2 The integer represented by the outbound value of `SpecVersion` (or zero if the outbound value is an empty string) shall be written to the field `SpecVersion` of *V*.

9.2.9.4.3 The character string that is the outbound value of `Description`, `ProductVersion`, and `Vendor` shall be written to the field of *V* with the same name, terminated by a NUL (0) character.

9.2.9.4.4 The octet string represented by the outbound value of `FwProperty` shall be written to a memory block of sufficient size. The address and length of that memory block shall be written to the fields `Data` and

`Length` (respectively) of a variable of type `BioAPI_DATA`, whose address shall be written to the field `FwProperty` of *V*.

9.2.9.4.5 The character string that is the outbound value of `Path` shall be written, with a NUL (0) character appended to it, to a memory block of sufficient size, whose address shall be written to the field `Path` of *V*.

9.2.9.5 Given a value (say, *V*) of the native type `BioAPI_FRAMEWORK_SCHEMA`, the inbound value of this parameter group that canonically represents *V* shall be determined as specified in the five following subclauses.

9.2.9.5.1 The inbound value of `FrameworkUuid` and `FwPropertyId` shall be the canonical representation (see 7.6.3) of the UUID in the field of *V* with the same name.

9.2.9.5.2 The inbound value of `SpecVersion` shall be the canonical representation of the integer in the field of *V* with the same name.

9.2.9.5.3 The inbound value of `Description`, `ProductVersion`, and `Vendor` shall be the character string in the field of *V* with the same name, without the final NUL (0) character.

9.2.9.5.4 The inbound value of `FwProperty` shall be the canonical representation of the octet string in the memory block whose address and length are in the fields `Data` and `Length` (respectively) of the variable of type `BioAPI_DATA` pointed to by the field `FwProperty` of *V*.

9.2.9.5.5 The inbound value of `Path` shall be the character string pointed to by the field of *V* with the same name, without the final NUL (0) character.

9.2.10 Parameter group "BSP schema"

9.2.10.1 The parameter group "BSP schema" consists of the following parameters:

- **BSPUuid**
- **Description**
- **Path**
- **SpecVersion**
- **ProductVersion**
- **Vendor**
- **Format_1_FormatOwner**
- **Format_1_FormatType**
- **Format_2_FormatOwner**
- **Format_2_FormatType**
- **Format_3_FormatOwner**
- **Format_3_FormatType**
- **Format_4_FormatOwner**
- **Format_4_FormatType**
- **NumSupportedFormats**
- the members of the parameter group "Biometric type" (see 9.2.1)
- the members of the parameter group "Operations" (see 9.2.2)
- the members of the parameter group "Options" (see 9.2.3)

- **PayloadPolicy**
- **MaxPayloadSize**
- **DefaultVerifyTimeout**
- **DefaultIdentifyTimeout**
- **DefaultCaptureTimeout**
- **DefaultEnrollTimeout**
- **DefaultCalibrateTimeout**
- **MaxBSPDbSize**
- **MaxIdentify**

9.2.10.2 This parameter group supports the representation of the values of the native type **`BioAPI_BSP_SCHEMA`**.

9.2.10.3 The members of this parameter group shall have inbound and outbound values as specified in the ten following subclauses.

9.2.10.3.1 An outbound value of **`BSPUuid`** shall be either a valid representation of a UUID (see 7.6), or an empty string. An inbound value shall be the canonical representation of a UUID (see 7.6.3).

9.2.10.3.2 An outbound or inbound value of **`Description`**, **`ProductVersion`**, and **`Vendor`** shall be a character string whose UTF-8 encoding is no longer than 268 octets, and which does not contain any NUL (0) characters.

9.2.10.3.3 An outbound or inbound value of **`Path`** shall be a character string that does not contain any NUL (0) characters.

9.2.10.3.4 An outbound value of **`SpecVersion`** shall be either a valid representation of an integer (see 7.4) in the range 0 to 255, or an empty string. An inbound value shall be the canonical representation of an integer (see 7.4.3) in the same range.

9.2.10.3.5 An outbound value of **`Format_`*X*`_FormatOwner`** and **`Format_`*X*`_FormatType`** (with *X*=1, 2, 3, or 4), shall be either a valid representation of an integer (see 7.4) in the range 0 to 65535, or an empty string. An inbound value shall be either the canonical representation of an integer (see 7.4.3) in the same range, or an empty string.

9.2.10.3.6 An outbound value of **`NumSupportedFormats`**, **`MaxPayloadSize`**, **`MaxBSPDbSize`**, and **`MaxIdentify`** shall be either a valid representation of an integer (see 7.4) in the range 0 to 4294967295, or an empty string. An inbound value shall be the canonical representation of an integer (see 7.4.3) in the same range.

9.2.10.3.7 An outbound value of the parameter group "Biometric type" shall be a valid representation of a value of the native type **`BioAPI_BIR_DATA_TYPE`** (see 9.2.5.4). An inbound value shall be the canonical representation of a value of that type (see 9.2.5.5).

9.2.10.3.8 An outbound value of the parameter group "Operations" shall be a valid representation of a value of the native type **`BioAPI_OPERATIONS_MASK`** (see 9.2.2.4). An inbound value shall be the canonical representation of a value of that type (see 9.2.2.5).

9.2.10.3.9 An outbound value of the parameter group "Options" shall be a valid representation of a value of the native type **`BioAPI_OPTIONS_MASK`** (see 9.2.3.4). An inbound value shall be the canonical representation of a value of that type (see 9.2.3.5).

9.2.10.3.10 An outbound value of `PayloadPolicy`, `DefaultVerifyTimeout`, `DefaultIdentifyTimeout`, `DefaultCaptureTimeout`, `DefaultEnrollTimeout`, and `DefaultCalibrateTimeout` shall be either a valid representation of an integer (see 7.4) in the range -2147483648 to 2147483647, or an empty string. An inbound value shall be the canonical representation of an integer (see 7.4.3) in the same range.

9.2.10.4 The value (say, *V*) of the native type `BioAPI_BSP_SCHEMA` represented by an outbound value of this parameter group shall be determined as specified in the eight following subclauses.

9.2.10.4.1 The UUID represented by the outbound value of `BSPUuid` (or the UUID "`00000000-0000-0000-0000-000000000000`" if the outbound value is an empty string) shall be written to the field of *V* with the same name.

9.2.10.4.2 The character string that is the outbound value of `Description`, `ProductVersion`, and `Vendor` and shall be written to the field of *V* with the same name, terminated by a NUL (0) character.

9.2.10.4.3 The character string that is the outbound value of `Path` shall be written to a memory block of sufficient size, whose address shall be written to the field `Path` of *V*.

9.2.10.4.4 The integer represented by the outbound value of `SpecVersion`, `NumSupportedFormats`, `PayloadPolicy`, `MaxPayloadSize`, `DefaultVerifyTimeout`, `DefaultIdentifyTimeout`, `DefaultCaptureTimeout`, `DefaultEnrollTimeout`, `DefaultCalibrateTimeout`, `MaxBSPDbSize`, and `MaxIdentify` (or zero if the outbound value is an empty string) shall be written to the field of *V* with the same name.

9.2.10.4.5 The integer represented by the outbound value of `Format_`*X*`_FormatOwner` and `Format_`*X*`_FormatType` (with *X*=1, 2, 3, or 4), or zero if the outbound value is an empty string, shall be written to the fields `FormatOwner` and `FormatType` (respectively) of the element at position *X* of an array of four elements of type `BioAPI_BIR_BIOMETRIC_DATA_FORMAT`. The address of that array shall be written to the field `BSPSupportedFormats` of *V*.

9.2.10.4.6 The value of type `BioAPI_BIR_BIOMETRIC_TYPE` represented (see 9.2.1.4) by the outbound value of the parameter group "Biometric type" shall be written to the field `FactorsMask` of *V*.

9.2.10.4.7 The value of type `BioAPI_OPERATIONS` represented (see 9.2.2.4) by the outbound value of the parameter group "Operations" shall be written to the field `Operations` of *V*.

9.2.10.4.8 The value of type `BioAPI_OPTIONS` represented (see 9.2.3.4) by the outbound value of the parameter group "Options" shall be written to the field `Options` of *V*.

9.2.10.5 Given a value (say, *V*) of the native type `BioAPI_BSP_SCHEMA`, the inbound value of this parameter group that canonically represents *V* shall be determined as specified in the eight following subclauses.

9.2.10.5.1 The inbound value of `BSPUuid` shall be the canonical representation (see 7.6.3) of the UUID in the field of *V* with the same name.

9.2.10.5.2 The inbound value of `Description`, `ProductVersion`, and `Vendor` shall be the character string in the field of *V* with the same name.

9.2.10.5.3 The inbound value of `Path` shall be the character string pointed to by the field of *V* with the same name.

9.2.10.5.4 The inbound value of `SpecVersion`, `NumSupportedFormats`, `PayloadPolicy`, `MaxPayloadSize`, `DefaultVerifyTimeout`, `DefaultIdentifyTimeout`, `DefaultCaptureTimeout`, `DefaultEnrollTimeout`, `DefaultCalibrateTimeout`, `MaxBSPDbSize`, and `MaxIdentify` shall be the canonical representation of the integer in the field of *V* with the same name.

9.2.10.5.5 The inbound value of `Format_`*X*`_FormatOwner` and `Format_`*X*`_FormatType` (with *X*=1, 2, 3, or 4) shall be determined as follows. If the field `BSPSupportedFormats` of *V* is NULL or the field `NumSupportedFormats` of *V* is less than *X*, then the inbound value shall be an empty string. Otherwise, the inbound value shall be the canonical representation of the integer in the fields `FormatOwner` and `FormatType` (respectively) of the element at position *X* of the array of type `BioAPI_BIR_BIOMETRIC_DATA_FORMAT` pointed to by the field `BSPSupportedFormats` of *V*.

9.2.10.5.6 The inbound value of the parameter group "Biometric type" shall be the canonical representation (see 9.2.1.5) of the value of type `BioAPI_BIR_BIOMETRIC_TYPE` in the field `FactorsMask` of *V*.

9.2.10.5.7 The inbound value of the parameter group "Operations" shall be the canonical representation (see 9.2.2.5) of the value of type `BioAPI_OPERATIONS` in the field `Operations` of *V*.

9.2.10.5.8 The inbound value of the parameter group "Options" shall be the canonical representation (see 9.2.3.5) of the value of type `BioAPI_OPTIONS` in the field `Options` of *V*.

9.2.11 Parameter group "BFP schema"

9.2.11.1 The parameter group "BFP schema" consists of the following parameters:

- **BFPUuid**
- **BFPCategory**
- **BFPDescription**
- **Path**
- **SpecVersion**
- **ProductVersion**
- **Vendor**
- **Format_1_FormatOwner**
- **Format_1_FormatType**
- **Format_2_FormatOwner**
- **Format_2_FormatType**
- **Format_3_FormatOwner**
- **Format_3_FormatType**
- **Format_4_FormatOwner**
- **Format_4_FormatType**
- **NumSupportedFormats**
- the members of the parameter group "Biometric type" (see 9.2.1)
- **BFPPropertyID**
- **BFPProperty**

9.2.11.2 This parameter group supports the representation of the values of the native type `BioAPI_BFP_SCHEMA`.

9.2.11.3 The members of this parameter group shall have inbound and outbound values as specified in the eight following subclauses.

9.2.11.3.1 An outbound value of **`BFPUuid`** and **`BFPPropertyID`** shall be either a valid representation of a UUID (see 7.6), or an empty string. An inbound value shall be the canonical representation of a UUID (see 7.6.3).

9.2.11.3.2 An outbound value of **`BFPCategory`** and **`NumSupportedFormats`** shall be either a valid representation of an integer in the range 0 to 4294967295, or an empty string. An inbound value shall be the canonical representation of an integer in the same range.

9.2.11.3.3 An outbound or inbound value of **`BFPDescription`**, **`ProductVersion`**, and **`Vendor`** shall be a character string whose UTF-8 encoding is no longer than 268 octets, and which does not contain any NUL (0) characters.

9.2.11.3.4 An outbound or inbound value of **`Path`** shall be a character string that does not contain any NUL (0) characters.

9.2.11.3.5 An outbound value of **`SpecVersion`** shall be either a valid representation of an integer in the range 0 to 255, or an empty string. An inbound value shall be the canonical representation of an integer in the same range.

9.2.11.3.6 An outbound value of **`Format_`*`X`*`_FormatOwner`** and **`Format_`*`X`*`_FormatType`** (with *X*=1, 2, 3, or 4), shall be either a valid representation of an integer (see 7.4) in the range 0 to 65535, or an empty string. An inbound value shall be either the canonical representation of an integer (see 7.4.3) in the same range, or an empty string.

9.2.11.3.7 An outbound value of the parameter group "Biometric type" shall be a valid representation of a value of the native type **`BioAPI_BIR_DATA_TYPE`** (see 9.2.5.4). An inbound value shall be the canonical representation of a value of that type (see 9.2.5.5).

9.2.11.3.8 An outbound value of **`BFPProperty`** shall be a valid representation of an octet string (see 7.7). An inbound value shall be the canonical representation of an octet string (see 7.7.2).

9.2.11.4 The value (say, *V*) of the native type **`BioAPI_BFP_SCHEMA`** represented by an outbound value of this parameter group shall be determined as specified in the eight following subclauses.

9.2.11.4.1 The UUID represented by the outbound value of **`BFPUuid`** and **`BFPPropertyID`** (or the UUID "`00000000-0000-0000-0000-000000000000`" if the outbound value is an empty string) shall be written to the field of *V* with the same name.

9.2.11.4.2 The integer represented by the outbound value of **`BFPCategory`** and **`NumSupportedFormats`** (or zero if the outbound value is an empty string) shall be written to the field of *V* with the same name.

9.2.11.4.3 The character string that is the outbound value of **`BFPDescription`**, **`ProductVersion`**, and **`Vendor`** shall be written to the field of *V* with the same name, terminated by a NUL (0) character.

9.2.11.4.4 The character string that is the outbound value of **`Path`** shall be written, with a NUL (0) character appended to it, to a memory block of sufficient size, whose address shall be written to the field **`Path`** of *V*.

9.2.11.4.5 The integer represented by the outbound value of **`SpecVersion`** (or zero if the outbound value is an empty string) shall be written to the field of *V* with the same name.

9.2.11.4.6 The integer represented by the outbound value of **`Format_`*`X`*`_FormatOwner`** and **`Format_`*`X`*`_FormatType`** (with *X*=1, 2, 3, or 4), or zero if the outbound value is an empty string, shall be written to the fields **`FormatOwner`** and **`FormatType`** (respectively) of the element at position *X* of an array of four elements of type **`BioAPI_BIR_BIOMETRIC_DATA_FORMAT`**. The address of that array shall be written to the field **`BFPSupportedFormats`** of *V*.

9.2.11.4.7 The value of type **`BioAPI_BIR_BIOMETRIC_TYPE`** represented (see 9.2.1.4) by the outbound value of the parameter group "Biometric type" shall be written to the field **`FactorsMask`** of *V*.

9.2.11.4.8 The octet string represented by the outbound value of **`BFPProperty`** shall be written to a memory block of sufficient size. The address and length of that memory block shall be written to the fields **`Data`** and **`Length`** (respectively) of a variable of type **`BioAPI_DATA`**, whose address shall be written to the field **`BFPProperty`** of *V*.

9.2.11.5 Given a value (say, *V*) of the native type **`BioAPI_BFP_SCHEMA`**, the inbound value of this parameter group that canonically represents *V* shall be determined as specified in the eight following subclauses.

9.2.11.5.1 The inbound value of **`BFPUuid`** and **`BFPPropertyID`** shall be the canonical representation (see 7.6.3) of the UUID in the field of *V* with the same name.

9.2.11.5.2 The inbound value of **`BFPCategory`** and **`NumSupportedFormats`** shall be the canonical representation of the integer in the field of *V* with the same name.

9.2.11.5.3 The inbound value of **`BFPDescription`**, **`ProductVersion`**, and **`Vendor`** shall be the character string in the field of *V* with the same name, without the final NUL (0) character.

9.2.11.5.4 The inbound value of **`Path`** shall be the character string pointed to by the field of *V* with the same name, without the final NUL (0) character.

9.2.11.5.5 The inbound value of **`SpecVersion`** shall be the canonical representation of the integer in the field of *V* with the same name.

9.2.11.5.6 The inbound value of **`Format_`*X*`_FormatOwner`** and **`Format_`*X*`_FormatType`** (with *X*=1, 2, 3, or 4) shall be determined as follows. If the field **`BFPSupportedFormats`** of *V* is NULL or the field **`NumSupportedFormats`** of *V* is less than *X*, then the inbound value shall be an empty string. Otherwise, the inbound value shall be the canonical representation of the integer in the fields **`FormatOwner`** and **`FormatType`** (respectively) of the element at position *X* of the array of type **`BioAPI_BIR_BIOMETRIC_DATA_FORMAT`** pointed to by the field **`BFPSupportedFormats`** of *V*.

9.2.11.5.7 The inbound value of the parameter group "Biometric type" shall be the canonical representation (see 9.2.1.5) of the value of type **`BioAPI_BIR_BIOMETRIC_TYPE`** in the field **`FactorsMask`** of *V*.

9.2.11.5.8 The inbound value of **`BFPProperty`** shall be the canonical representation of the octet string in the memory block whose address and length are in the fields **`Data`** and **`Length`** (respectively) of the variable of type **`BioAPI_DATA`** pointed to by the field **`BFPProperty`** of *V*.

9.2.12 Parameter group "Unit schema"

9.2.12.1 The parameter group "Unit schema" consists of the following parameters:

- **BSPUuid**
- **UnitManagerUuid**
- **UnitID**
- **UnitCategory**
- **UnitProperties**
- **VendorInformation**
- the members of the parameter group "Events" (see 9.2.4)
- **UnitPropertyID**

- **UnitProperty**
- **HardwareVersion**
- **FirmwareVersion**
- **SoftwareVersion**
- **HardwareSerialNumber**
- **AuthenticatedHardware**
- **MaxBSPDbSize**
- **MaxIdentify**

9.2.12.2 This parameter group supports the representation of the values of the native type `BioAPI_UNIT_SCHEMA`.

9.2.12.3 The members of this parameter group shall have inbound and outbound values as specified in the six following subclauses.

9.2.12.3.1 An outbound value of `BSPUuid`, `UnitManagerUuid`, UnitProperties, and `UnitPropertyID` shall be either a valid representation of a UUID (see 7.6), or an empty string. An inbound value shall be the canonical representation of a UUID (see 7.6.3).

9.2.12.3.2 outbound value of `UnitID`, `UnitCategory`, `MaxBSPDbSize`, and `MaxIdentify` shall be either a valid representation of an integer in the range 0 to 4294967295, or an empty string. An inbound value shall be the canonical representation of an integer in the same range.

9.2.12.3.3 An outbound or inbound value of `VendorInformation`, `HardwareVersion`, `FirmwareVersion`, `SoftwareVersion`, and `HardwareSerialNumber` shall be a character string whose UTF-8 encoding is no longer than 268 octets, and which does not contain any NUL (0) characters.

9.2.12.3.4 An outbound value of the parameter group "Events" shall be a valid representation of a value of the native type `BioAPI_EVENT_MASK`. An inbound value shall be the canonical representation of a value of that type.

9.2.12.3.5 An outbound value of `UnitProperty` shall be a valid representation of an octet string (see 7.7). An inbound value shall be the canonical representation of an octet string (see 7.7.2).

9.2.12.3.6 An outbound value of `AuthenticatedHardware` shall be either a valid representation of a boolean (see 7.5) or an empty string. An inbound value shall be the canonical representation of a boolean (see 7.5.2).

9.2.12.4 The value (say, *V*) of the native type `BioAPI_UNIT_`represented by an outbound value of this parameter group shall be determined as specified in the six following subclauses.

9.2.12.4.1 The UUID represented by the outbound value of `BSPUuid`, `UnitManagerUuid`, `UnitProperties`, and `UnitPropertyID` (or the UUID "`00000000-0000-0000-0000-000000000000`" if the outbound value is an empty string) shall be written to the field of *V* with the same name.

9.2.12.4.2 The integer represented by the outbound value of `UnitID`, `UnitCategory`, `MaxBSPDbSize`, and `MaxIdentify` (or zero if the outbound value is an empty string) shall be written to the field of *V* with the same name.

9.2.12.4.3 The value of type `BioAPI_EVENT_MASK` represented (see 9.2.4.4) by the outbound value of the parameter group "Events" shall be written to the field `SupportedEvents` of *V*.

9.2.12.4.4 The character string that is the outbound value of **VendorInformation**, **HardwareVersion**, **FirmwareVersion**, **SoftwareVersion**, and **HardwareSerialNumber** shall be written to the field of *V* with the same name, terminated by a NUL (0) character.

9.2.12.4.5 The octet string represented by the outbound value of **UnitProperty** shall be written to a memory block of sufficient size. The address and length of that memory block shall be written to the fields **Data** and **Length** (respectively) of a variable of type **BioAPI_DATA**, whose address shall be written to the field **UnitProperty** of *V*.

9.2.12.4.6 The boolean represented by the outbound value of **AuthenticatedHardware** (or "**false**" if the outbound value is an empty string) shall be written to the field of *V* with the same name.

9.2.12.5 Given a value (say, *V*) of the native type **BioAPI_UNIT_SCHEMA**, the inbound value of this parameter group that canonically represents *V* shall be determined as specified in the six following subclauses.

9.2.12.5.1 The inbound value of **BSPUuid**, **UnitManagerUuid**, **UnitProperties**, and **UnitPropertyID** shall be the canonical representation (see 7.6.3) of the UUID in the field of *V* with the same name.

9.2.12.5.2 The inbound value of **UnitID**, **UnitCategory**, **MaxBSPDbSize**, and **MaxIdentify** shall be the canonical representation of the integer in the field of *V* with the same name.

9.2.12.5.3 The inbound value of the parameter group "Events" shall be the canonical representation (see 9.2.1.5) of the value of type **BioAPI_EVENT_MASK** in the field **SupportedEvents** of *V*.

9.2.12.5.4 The inbound value of **VendorInformation**, **HardwareVersion**, **FirmwareVersion**, **SoftwareVersion**, and **HardwareSerialNumber** shall be the character string in the field of *V* with the same name, without the final NUL (0) character.

9.2.12.5.5 The inbound value of **UnitProperty** shall be the canonical representation of the octet string in the memory block whose address and length are in the fields **Data** and **Length** (respectively) of the variable of type **BioAPI_DATA** pointed to by the field **UnitProperty** of *V*.

9.2.12.5.6 The inbound value of **AuthenticatedHardware** shall be the canonical representation (see 7.5.2) of the boolean in the field of *V* with the same name.

9.2.13 Parameter group "BIR header"

9.2.13.1 The parameter group "BIR header" consists of the following parameters:

- **HeaderVersion**
- the members of the parameter group "Biometric data type" (see 9.2.5)
- **FormatOwner**
- **FormatType**
- **Quality**
- **Purpose**
- the members of the parameter group "Biometric type" (see 9.2.1)
- **ProductOwner**
- **ProductType**
- the members of the parameter group "Date and time" (see 9.2.8), with their names prefixed by "**Creation_**"
- the members of the parameter group "Biometric subtype" (see 9.2.6)

- the members of the parameter group "Date" (see 9.2.7), with their names prefixed by **"Expiration_"**
- **SBFormatOwner**
- **SBFormatType**
- **Index**

9.2.13.2 This parameter group supports the representation of the values of the native type `BioAPI_BIR_HEADER`.

9.2.13.3 The members of this parameter group shall have inbound and outbound values as specified in the nine following subclauses.

9.2.13.3.1 An outbound value of `HeaderVersion` and `Purpose` shall be either a valid representation of an integer in the range 0 to 255, or an empty string. An inbound value shall be the canonical representation of an integer in the same range.

9.2.13.3.2 An outbound value of the parameter group "Biometric data type" shall be a valid representation of a value of the native type `BioAPI_BIR_DATA_TYPE`. An inbound value shall be the canonical representation of a value of that type.

9.2.13.3.3 An outbound value of `FormatOwner`, `FormatType`, `ProductOwner`, `ProductType`, `SBFormatOwner`, and `SBFormatType` shall be either a valid representation of an integer in the range 0 to 65535, or an empty string. An inbound value shall be the canonical representation of an integer in the same range.

9.2.13.3.4 An outbound value of `Quality` shall be either a valid representation of an integer in the range -128 to 127, or an empty string. An inbound value shall be the canonical representation of an integer in the same range.

9.2.13.3.5 An outbound value of the parameter group "Biometric type" shall be a valid representation of a value of the native type `BioAPI_BIR_BIOMETRIC_TYPE`. An inbound value shall be the canonical representation of a value of that type.

9.2.13.3.6 An outbound value of the parameter group "Date and time" shall be a valid representation of a value of the native type `BioAPI_DTG`. An inbound value shall be the canonical representation of a value of that type.

9.2.13.3.7 An outbound value of the parameter group "Biometric subtype" shall be a valid representation of a value of the native type `BioAPI_BIR_SUBTYPE`. An inbound value shall be the canonical representation of a value of that type.

9.2.13.3.8 An outbound value of the parameter group "Date" shall be a valid representation of a value of the native type `BioAPI_DATE`. An inbound value shall be the canonical representation of a value of that type.

9.2.13.3.9 An outbound value of `Index` shall be either a valid representation of a UUID (see 7.6) or an empty string. An inbound value shall be the canonical representation of a UUID (see 7.6.3).

9.2.13.4 The value (say, *V*) of the native type `BioAPI_BIR_HEADER` represented by an outbound value of this parameter group shall be determined as specified in the ten following subclauses.

9.2.13.4.1 The integer represented by the outbound value of `HeaderVersion`, `Quality`, and `Purpose` (or zero if the outbound value is an empty string) shall be written to the field of *V* with the same name.

9.2.13.4.2 The value of type `BioAPI_BIR_DATA_TYPE` represented (see 9.2.5.4) by the outbound value of the parameter group "Biometric data type" shall be written to the field `Type` of *V*.

9.2.13.4.3 The integer represented by the outbound value of `FormatOwner` and `FormatType` (or zero if the outbound value is an empty string) shall be written to the fields `FormatOwner` and `FormatType` (respectively) of the field `Format` of *V*.

9.2.13.4.4 The value of type `BioAPI_BIR_BIOMETRIC_TYPE` represented (see 9.2.1.4) by the outbound value of the parameter group "Biometric type" shall be written to the field `FactorsMask` of *V*.

9.2.13.4.5 The integer represented by the outbound value of `ProductOwner` and `ProductType` (or zero if the outbound value is an empty string) shall be written to the fields `ProductOwner` and `ProductType` (respectively) of the field `ProductID` of *V*.

9.2.13.4.6 The value of type `BioAPI_DTG` represented (see 9.2.8.4) by the outbound value of the parameter group "Date and time" (with the prefix "`Creation_`") shall be written to the field `CreationDTG` of *V*.

9.2.13.4.7 The value of type `BioAPI_BIR_SUBTYPE` represented (see 9.2.6.4) by the outbound value of the parameter group "Biometric subtype" shall be written to the field `Subtype` of *V*.

9.2.13.4.8 The value of type `BioAPI_DATE` represented (see 9.2.7.4) by the outbound value of the parameter group "Date" (with the prefix "`Expiration_`") shall be written to the field `ExpirationDate` of *V*.

9.2.13.4.9 The integer represented by the outbound value of `SBFormatOwner` and `SBFormatType` (or zero if the outbound value is an empty string) shall be written to the fields `SecurityFormatOwner` and `SecurityFormatType` (respectively) of the field `SBFormat` of *V*.

9.2.13.4.10 The UUID represented by the outbound value of `Index` (or the UUID "`00000000-0000-0000-0000-000000000000`" if the outbound value is an empty string) shall be written to the field of *V* with the same name.

9.2.13.5 Given a value (say, *V*) of the native type `BioAPI_BIR_HEADER`, the inbound value of this parameter group that canonically represents *V* shall be determined as specified in the ten following subclauses.

9.2.13.5.1 The inbound value of `HeaderVersion`, `Quality`, and `Purpose` shall be the canonical representation of the integer in the field of *V* with the same name.

9.2.13.5.2 The inbound value of the parameter group "Biometric data type" shall be the canonical representation (see 9.2.5.5) of the value of type `BioAPI_BIR_DATA_TYPE` in the field `Type` of *V*.

9.2.13.5.3 The inbound value of `FormatOwner` and `FormatType` shall be the canonical representation of the integer in the fields `FormatOwner` and `FormatType` (respectively) of the field `Format` of *V*.

9.2.13.5.4 The inbound value of the parameter group "Biometric type" shall be the canonical representation (see 9.2.1.5) of the value of type `BioAPI_BIR_BIOMETRIC_TYPE` in the field `FactorsMask` of *V*.

9.2.13.5.5 The inbound value of `ProductOwner` and `ProductType` shall be the canonical representation of the integer in the fields `ProductOwner` and `ProductType` (respectively) of the field `ProductID` of *V*.

9.2.13.5.6 The inbound value of the parameter group "Date and time" (with the prefix "`Creation_`") shall be the canonical representation (see 9.2.8.5) of the value of type `BioAPI_DTG` in the field `CreationDTG` of *V*.

9.2.13.5.7 The inbound value of the parameter group "Biometric subtype" shall be the canonical representation (see 9.2.6.5) of the value of type `BioAPI_BIR_SUBTYPE` in the field `Subtype` of *V*.

9.2.13.5.8 The inbound value of the parameter group "Date" (with the prefix "`Expiration_`") shall be the canonical representation (see 9.2.7.5) of the value of type `BioAPI_DATE` in the field `ExpirationDate` of *V*.

9.2.13.5.9 The inbound value of `SBFormatOwner` and `SBFormatType` shall be the canonical representation of the integer in the fields `SecurityFormatOwner` and `SecurityFormatType` (respectively) of the field `SBFormat` of *V*.

9.2.13.5.10 The inbound value of `Index` shall be the canonical representation (see 7.6.3) of the UUID in the field of *V* with the same name.

9.2.14 Parameter group "BIR"

9.2.14.1 The parameter group "BIR" consists of the following parameters:

— the members of the parameter group "BIR header" (see 9.2.13)

— BiometricData

— SecurityBlock

9.2.14.2 This parameter group supports the representation of the values of the native type `BioAPI_BIR`.

9.2.14.3 The members of this parameter group shall have inbound and outbound values as specified in the two following subclauses.

9.2.14.3.1 An outbound value of the parameter group "BIR header" shall be a valid representation of a value of the native type `BioAPI_BIR_HEADER`. An inbound value shall be the canonical representation of a value of that type.

9.2.14.3.2 An outbound value of `BiometricData` and `SecurityBlock` shall be a valid representation of an octet string (see 7.7). An inbound value shall be the canonical representation of an octet string (see 7.7.2).

9.2.14.4 The value (say, *V*) of the native type `BioAPI_BIR` represented by an outbound value of this parameter group shall be determined as specified in the three following subclauses.

9.2.14.4.1 The value of type `BioAPI_BIR_HEADER` represented (see 9.2.13.4) by the outbound value of the parameter group "BIR header" shall be written to the field `Header` of *V*.

9.2.14.4.2 The octet string represented by the outbound value of `BiometricData` shall be written to a memory block of sufficient size. The address and length of that memory block shall be written to the fields `Data` and `Length` (respectively) of the field `BiometricData` of V.

9.2.14.4.3 The octet string represented by the outbound value of `SecurityBlock` shall be written to a memory block of sufficient size. The address and length of that memory block shall be written to the fields `Data` and `Length` (respectively) of the field `SecurityBlock` of *V*.

9.2.14.5 Given a value (say, *V*) of the native type `BioAPI_BIR`, the inbound value of this parameter group that canonically represents *V* shall be determined as specified in the three following subclauses.

9.2.14.5.1 The inbound value of the parameter group "BIR header" shall be the canonical representation (see 9.2.13.5) of the value of type `BioAPI_BIR_HEADER` in the field `Header` of *V*.

9.2.14.5.2 The inbound value of `BiometricData` shall be the canonical representation of the octet string in the memory block whose address and length are in the fields `Data` and `Length` (respectively) of the field `BiometricData` of *V*.

9.2.14.5.3 The inbound value of `SecurityBlock` shall be the canonical representation of the octet string in the memory block whose address and length are in the fields `Data` and `Length` (respectively) of the field `SecurityBlock` of *V*.

9.2.15 Parameter group "Input BIR"

9.2.15.1 The parameter group "Input BIR" consists of the following parameters:

- Form
- DbHandle
- KeyValue
- BIRHandle
- the members of the parameter group "BIR" (see 9.2.14).

9.2.15.2 This parameter group supports the representation of the values of the native type `BioAPI_INPUT_BIR`.

9.2.15.3 The members of this parameter group shall have inbound and outbound values as specified in the four following subclauses.

9.2.15.3.1 An outbound value of `Form` shall be either a valid representation of an integer (see 7.4) in the range 0 to 255, or an empty string. An inbound value shall be the canonical representation of an integer (see 7.4.3) in the same range.

9.2.15.3.2 An outbound value of `DbHandle` and `BIRHandle` shall be either a valid representation of an integer in the range -2147483648 to 2147483647, or an empty string. An inbound value shall be the canonical representation of an integer in the same range.

9.2.15.3.3 An outbound value of `KeyValue` shall be either a valid representation of a UUID (see 7.6), or an empty string. An inbound value shall be the canonical representation of a UUID (see 7.6.3).

9.2.15.3.4 An outbound value of the parameter group "BIR" shall be a valid representation of a value of the native type `BioAPI_BIR`. An inbound value shall be the canonical representation of a value of that type.

9.2.15.4 The value (say, *V*) of the native type `BioAPI_INPUT_BIR` represented by an outbound value of this parameter group shall be determined as specified in the five following subclauses.

9.2.15.4.1 The integer (say, *F*) represented by the outbound value of `Form` (or zero if the outbound value is an empty string) shall be written to the field of *V* with the same name.

9.2.15.4.2 If *F* is 1, then:

a) the integer represented by the outbound value of `DbHandle` (or zero if the outbound value is an empty string) shall be written to the field `DbHandle` of a variable of type `BioAPI_DBBIR_ID`;

b) the UUID represented by the outbound value of `KeyValue` (or the UUID "`00000000-0000-0000-0000-000000000000`" if the outbound value is an empty string) shall be written to the field `KeyValue` of that variable; and

c) the address of that variable shall be written to the field `BIRInDb` of the field `InputBIR` of *V*.

9.2.15.4.3 If *F* is 2 then:

a) the integer represented by the outbound value of `BIRHandle` (or zero if the outbound value is an empty string) shall be written to a variable of type `BioAPI_BIR_HANDLE`; and

b) the address of that variable shall be written to the field `BIRInBSP` of the field `InputBIR` of *V*.

9.2.15.4.4 If *F* is 3, then:

a) the value of type `BioAPI_BIR` represented (see 9.2.14.4) by the outbound value of the parameter group "BIR" shall be written to a variable of type `BioAPI_BIR`; and

b) the address of that variable shall be written to the field `BIR` of the field `InputBIR` of *V*.

9.2.15.4.5 If *F* is less than 1 or greater than 3, then the field `BIR` of the field `InputBIR` of *V* shall be set to NULL.

9.2.15.5 Given a value (say, *V*) of the native type `BioAPI_INPUT_BIR`, the inbound value of this parameter group that canonically represents *V* shall be determined as specified in the five following subclauses.

9.2.15.5.1 The inbound value of `Form` shall be the canonical representation of the integer (say, *F*) in the field of *V* with the same name.

9.2.15.5.2 If *F* is 1, then:

a) the inbound value of `DbHandle` shall be the canonical representation of the integer in the field `DbHandle` of the variable of type `BioAPI_DBBIR_ID` pointed to by the field `BIRInDb` of the field `InputBIR` of *V*;

b) the inbound value of `KeyValue` shall be the canonical representation (see 7.6.3) of the UUID in the field `KeyValue` of that variable; and

c) the inbound value of all other parameters shall be an empty string.

9.2.15.5.3 If *F* is 2, then:

a) the inbound value of `BIRHandle` shall be the canonical representation of the integer in the variable of type `BioAPI_BIR_HANDLE` pointed to by the field `BIRInBSP` of the field `InputBIR` of *V*; and

b) the inbound value of all other parameters shall be an empty string.

9.2.15.5.4 If *F* is 3, then:

a) the inbound value of the parameter group "BIR" shall be the canonical representation (see 9.2.14.5) of the value of type `BioAPI_BIR` in the field `BIR` of the field `InputBIR` of *V*; and

b) the inbound value of all other parameters shall be an empty string.

9.2.15.5.5 If *F* is less than 1 or greater than 3, then the inbound value of all parameters except `Form` shall be an empty string.

9.2.16 Parameter group "Identify population"

9.2.16.1 The parameter group "Identify population" consists of the following parameters:

- **Type**
- **BIRDataBase**
- **NumberOfMembers**
- the members of the parameter group "BIR" (see 9.2.14) with their names prefixed by "`BIR_1_`"
- the members of the parameter group "BIR" with their names prefixed by "`BIR_2_`"
- the members of the parameter group "BIR" with their names prefixed by "`BIR_3_`"
- the members of the parameter group "BIR" with their names prefixed by "`BIR_4_`"

9.2.16.2 This parameter group supports the representation of the values of the native type **`BioAPI_IDENTIFY_POPULATION`**.

9.2.16.3 The members of this parameter group shall have inbound and outbound values as specified in the four following subclauses.

9.2.16.3.1 An outbound value of **`Type`** shall be either a valid representation of an integer (see 7.4) in the range 0 to 255, or an empty string. An inbound value shall be the canonical representation of an integer (see 7.4.3) in the same range.

9.2.16.3.2 An outbound value of **`BIRDataBase`** shall be either a valid representation of an integer in the range -2147483648 to 2147483647, or an empty string. An inbound value shall be the canonical representation of an integer in the same range.

9.2.16.3.3 An outbound value of **`NumberOfMembers`** shall be either a valid representation of an integer in the range 0 to 4, or an empty string. An inbound value of **`NumberOfMembers`** shall be the canonical representation of an integer in the range 0 to 4294967295.

9.2.16.3.4 An outbound value of each parameter group "BIR" shall be a valid representation of a value of the native type **`BioAPI_BIR`**. An inbound value shall be either the canonical representation of a value of that type, or a set of empty strings.

9.2.16.4 The value (say, *V*) of the native type **`BioAPI_IDENTIFY_POPULATION`** represented by an outbound value of this parameter group shall be determined as specified in the four following subclauses.

9.2.16.4.1 The integer (say, *T*) represented by the outbound value of **`Type`** (or zero if the outbound value is an empty string) shall be written to the field of *V* with the same name.

9.2.16.4.2 If *T* is 1, then:

a) the integer represented by the outbound value of **`BIRDataBase`** (or zero if the outbound value is an empty string) shall be written to a variable of type **`BioAPI_DB_HANDLE`**; and

b) the address of that variable shall be written to the field **`BIRDataBase`** of the field **`BIRs`** of *V*.

9.2.16.4.3 If *T* is 2, then:

a) the integer represented by the outbound value of **`NumberOfMembers`** (or zero if the outbound value is an empty string) shall be written to the field **`NumberOfMembers`** of a variable of type **`BioAPI_BIR_ARRAY_POPULATION`**;

b) the value of type **`BioAPI_BIR`** represented by the outbound value of the parameter group "BIR" with the prefix "**`BIR_`***X*_" (with *X*=1, 2, 3, or 4) shall be written to a variable of type **`BioAPI_BIR`**;

c) the addresses of all the variables of type **`BioAPI_BIR`** assigned in b) shall be written, in order, to an array of pointers;

d) the address of that array of pointers shall be written to the field **`Members`** of the variable of type **`BioAPI_BIR_ARRAY_POPULATION`** referenced in a);

e) the address of that variable shall be written to the field **`BIRArray`** of the field **`BIRs`** of *V*.

9.2.16.4.4 If *T* is less than 1 or greater than 2, then the field **`BIRArray`** of the field **`BIRs`** of *V* shall be set to NULL.

9.2.16.5 Given a value (say, *V*) of the native type `BioAPI_IDENTIFY_POPULATION`, the inbound value of this parameter group that canonically represents *V* shall be determined as specified in the four following subclauses.

9.2.16.5.1 The inbound value of `Type` shall be the canonical representation of the integer (say, *T*) in the field of *V* with the same name.

9.2.16.5.2 If *T* is 1, then:

a) the inbound value of `BIRDataBase` shall be the canonical representation of the integer in the variable of type `BioAPI_DB_HANDLE` pointed to by the field `BIRDataBase` of the field `BIRs` of *V*; and

b) the inbound value of all other parameters shall be an empty string.

9.2.16.5.3 If *T* is 2, then:

a) the inbound value of `NumberOfMembers` shall be the canonical representation of the integer in the field `NumberOfMembers` of the variable of type `BioAPI_BIR_ARRAY_POPULATION` pointed to by the field `BIRArray` of the field `BIRs` of *V*;

b) the inbound value of the parameter group "BIR" with the prefix "`BIR_`*X*`_`" (with *X*=1, 2, 3, and 4) shall be determined as follows. If the field `Members` of the variable in a) is NULL or the field `NumberOfMembers` is less than *X*, then the inbound value shall be a set of empty strings. Otherwise, the inbound value shall be the canonical representation of the inbound value of type `BioAPI_BIR` pointed to by the element at position *X* of the array of pointers pointed to by the field `Members`; and

c) the inbound value of `BIRDataBase` shall be an empty string.

9.2.16.5.4 If *T* is less than 1 or greater than 2, then the inbound value of all parameters except `Type` shall be an empty string.

9.2.17 Parameter group "Candidate"

9.2.17.1 The parameter group "Candidate" consists of the following parameters:

- **Type**
- **BIRInDataBase**
- **BIRInArray**
- **FMRAchieved**

9.2.17.2 This parameter group supports the representation of the values of the native type `BioAPI_CANDIDATE`.

9.2.17.3 The members of this parameter group shall have inbound and outbound values as specified in the four following subclauses.

9.2.17.3.1 An outbound value of `Type` shall be either a valid representation of an integer (see 7.4) in the range 0 to 255, or an empty string. An inbound value shall be the canonical representation of an integer (see 7.4.3) in the same range.

9.2.17.3.2 An outbound value of `BIRInDataBase` shall be either a valid representation of a UUID (see 7.6), or an empty string. An inbound value shall be the canonical representation of a UUID (see 7.6.3).

9.2.17.3.3 An outbound value of `BIRInArray` shall be either a valid representation of an integer in the range 0 to 4294967295, or an empty string. An inbound value shall be the canonical representation of an integer in the same range.

9.2.17.3.4 An outbound value of `FMRAchieved` shall be either a valid representation of an integer in the range -2147483648 to 2147483647, or an empty string. An inbound value shall be the canonical representation of an integer in the same range.

9.2.17.4 The value (say, *V*) of the native type `BioAPI_CANDIDATE` represented by an outbound value of this parameter group shall be determined as specified in the five following subclauses.

9.2.17.4.1 The integer (say, *T*) represented by the outbound value of `Type` (or zero if the outbound value is an empty string) shall be written to the field `Type` of *V*.

9.2.17.4.2 If *T* is 1, then:

a) the UUID represented by the outbound value of `BIRInDataBase` (or the UUID "`00000000-0000-0000-0000-000000000000`" if the outbound value is an empty string) shall be written to a variable of type `BioAPI_UUID`; and

b) the address of that variable shall be written to the field `BIRInDataBase` of the field `BIR` of *V*.

9.2.17.4.3 If *T* is either 2 or 3, then:

a) the integer represented by the outbound value of `BIRInArray` (or zero if the outbound value is an empty string) shall be written to a variable of type `uint32_t`; and

b) the address of that variable shall be written to the field `BIRInArray` of the field `BIR` of *V*.

9.2.17.4.4 If *T* is less than 1 or greater than 3, then the field `BIRInArray` of the field `BIR` of *V* shall be set to NULL.

9.2.17.4.5 The integer represented by the outbound value of `FMRAchieved` (or zero if the outbound value is an empty string) shall be written to the field of *V* with the same name.

9.2.17.5 Given a value (say, *V*) of the native type `BioAPI_CANDIDATE`, the inbound value of this parameter group that canonically represents *V* shall be determined as specified in the four following subclauses.

9.2.17.5.1 The inbound value of `Type` shall be the canonical representation of the integer (say, *T*) in the field of *V* with the same name.

9.2.17.5.2 If *T* is 1, then the inbound value of `BIRInDataBase` shall be the canonical representation of the UUID in the variable of type `BioAPI_UUID` pointed to by the field `BIRInDataBase` of *V*. Otherwise, the inbound value of `BIRInDataBase` shall be an empty string.

9.2.17.5.3 If *T* is either 2 or 3, then the inbound value of `BIRInArray` shall be the canonical representation of the integer in the variable of type `uint32_t` pointed to by the field `BIRInArray` of *V*. Otherwise, the inbound value of `BIRInArray` shall be an empty string.

9.2.17.5.4 The inbound value of `FMRAchieved` shall be the canonical representation of the integer (say, *T*) in the field of *V* with the same name.

9.2.18 Parameter group "GUI state"

9.2.18.1 The parameter group "GUI state" consists of the following parameters:

- **SampleAvailable**
- **MessageProvided**
- **ProgressProvided**

9.2.18.2 This parameter group supports the representation of the values of the native type `BioAPI_GUI_STATE` (which is an integer type).

9.2.18.3 An outbound value of these parameters shall be either a valid representation of a boolean (see 7.5) or an empty string. An inbound value of these parameters shall be the canonical representation of a boolean (see 7.5.2).

9.2.18.4 The value (say, *V*) of the native type `BioAPI_GUI_STATE` represented by an outbound value of this parameter group shall be determined as follows:

1) the value *V* shall be initially set to zero;

2) for each member of the parameter group whose outbound value is "`true`", an integer shall be added to *V* according to the following table.

`SampleAvailable`	$1=2^0$
`MessageProvided`	$2=2^1$
`ProgressProvided`	$4=2^2$

9.2.18.5 Given an integer (say, *V*) that is a value of the native type `BioAPI_GUI_STATE`, the inbound value of this parameter group that canonically represents *V* shall be determined as follows:

1) the integer *V* shall be decomposed into a sum of powers of two, each occurring at most once;

2) for each power of two that occurs in the decomposition of *V*, the inbound value of the corresponding parameter (according to the table above) shall be "`true`";

3) the inbound value of all the remaining parameters (if any) shall be "`false`".

9.2.19 Parameter group "Access type"

9.2.19.1 The parameter group "Access type" consists of the following parameters:

- **ReadAccess**
- **WriteAccess**

9.2.19.2 This parameter group supports the representation of the values of the native type `BioAPI_DB_ACCESS_TYPE` (which is an integer type).

9.2.19.3 An outbound value of these parameters shall be either a valid representation of a boolean (see 7.5) or an empty string. An inbound value of these parameters shall be the canonical representation of a boolean (see 7.5.2).

9.2.19.4 The value (say, *V*) of the native type `BioAPI_DB_ACCESS_TYPE` represented by an outbound value of this parameter group shall be determined as follows:

1) the value *V* shall be initially set to zero;

2) for each member of the parameter group whose outbound value is "`true`", an integer shall be added to *V* according to the following table.

`ReadAccess`	$1=2^0$
`WriteAccess`	$2=2^1$

9.2.19.5 Given an integer (say, *V*) that is a value of the native type `BioAPI_DB_ACCESS_TYPE`, the inbound value of this parameter group that canonically represents *V* shall be determined as follows:

1) the integer *V* shall be decomposed into a sum of powers of two, each occurring at most once;

2) for each power of two that occurs in the decomposition of *V*, the inbound value of the corresponding parameter (according to the table above) shall be "`true`";

3) the inbound value of all the remaining parameters (if any) shall be "`false`".

9.3 BioAPI_Init

9.3.1 Function Invocation Scheme

This function belongs to the BioAPI interface, and has the following native prototype:

```
BioAPI_RETURN BioAPI
BioAPI_Init(
    BioAPI_VERSION Version);
```

and the following parameters:

Parameter	Function invocation (conformance testing model for frameworks)	Bound activity invocation (conformance testing model for applications)
Version	input, outbound	input, inbound
return	return, inbound	input, inbound

9.3.2 Constraints on the Parameters

9.3.2.1 An outbound value of `Version` shall be a valid representation of an integer (see 7.4) in the range 0 to 255. An inbound value shall be the canonical representation of an integer (see 7.4.3) in the same range.

9.3.2.2 An outbound value of `return` shall be a valid representation of an integer in the range 0 to 4294967295. An inbound value shall be the canonical representation of an integer in the same range.

9.3.3 Function Invocation Input

The integer represented by the outbound value of Version shall be assigned to the native parameter with the same name.

9.3.4 Function Invocation Output

There are no output parameters.

9.3.5 Bound Activity Invocation Input

9.3.5.1 The inbound value of `Version` shall be the canonical representation of the integer in the native parameter with the same name.

9.3.5.2 The inbound value of `return` shall be the canonical representation of the integer returned by the native function.

9.3.6 Bound Activity Invocation Output

There are no output parameters.

9.4 BioAPI_Terminate

9.4.1 Function Invocation Scheme

This function belongs to the BioAPI interface, and has the following native prototype:

BioAPI_RETURN BioAPI

BioAPI_Terminate(void);

and the following parameters:

Parameter	**Function invocation (conformance testing model for frameworks)**	**Bound activity invocation (conformance testing model for applications)**
`return`	return, inbound	input, inbound

9.4.2 Constraints on the parameters

An outbound value of `return` shall be a valid representation of an integer in the range 0 to 4294967295. An inbound value shall be the canonical representation of an integer in the same range.

9.4.3 Function Invocation Input

There are no input parameters.

9.4.4 Function Invocation Output

There are no output parameters.

9.4.5 Bound Activity Invocation Input

The inbound value of `return` shall be the canonical representation of the integer returned by the native function.

9.4.6 Bound Activity Invocation Output

There are no output parameters.

9.5 BioAPI_GetFrameworkInfo

9.5.1 Function Invocation Scheme

This function belongs to the BioAPI interface, and has the following native prototype:

BioAPI_RETURN BioAPI

BioAPI_GetFrameworkInfo(

BioAPI_FRAMEWORK_SCHEMA *FrameworkSchema);

and the following parameters:

Parameter	**Function invocation (conformance testing model for frameworks)**	**Bound activity invocation (conformance testing model for applications)**
no_FrameworkSchema	input, outbound	input, inbound
the members of the parameter group "Framework schema" (see 9.2.9)	output, inbound	input, inbound
return	return, inbound	input, inbound

9.5.2 Constraints on the Parameters

9.5.2.1 An outbound value of `no_FrameworkSchema` shall be either a valid representation of a boolean (see 7.5) or an empty string. An inbound value shall be the canonical representation of a boolean (see 7.5.2).

9.5.2.2 An outbound value of the parameter group "Framework schema" shall be a valid representation of a value of the native type `BioAPI_FRAMEWORK_SCHEMA`. An inbound value shall be either the canonical representation of a value of that type, or a set of empty strings.

9.5.2.3 An outbound value of `return` shall be a valid representation of an integer in the range 0 to 4294967295. An inbound value shall be the canonical representation of an integer in the same range.

9.5.3 Function Invocation Input

If the outbound value of `no_FrameworkSchema` is "`true`", then the native parameter `FrameworkSchema` shall be set to NULL, otherwise it shall be set to the address of a variable of type `BioAPI_FRAMEWORK_SCHEMA`.

9.5.4 Function Invocation Output

The inbound value of the parameter group "Framework schema" shall be determined as follows. If the native parameter `FrameworkSchema` is NULL, then the inbound value shall be a set of empty strings. Otherwise, the inbound value shall be the canonical representation (see 9.2.9.5) of the value of type `BioAPI_FRAMEWORK_SCHEMA` pointed to by the native parameter `FrameworkSchema`.

9.5.5 Bound Activity Invocation Input

The inbound value of `no_FrameworkSchema` shall be "`true`" if the native parameter `FrameworkSchema` is NULL, otherwise it shall be "`false`".

9.5.6 Bound Activity Invocation Output

There are no output parameters.

9.6 BioAPI_EnumBSPs

9.6.1 Function Invocation Scheme

This function belongs to the BioAPI interface, and has the following native prototype:

```
BioAPI_RETURN BioAPI
BioAPI_EnumBSPs(
    BioAPI_BSP_SCHEMA **BSPSchemaArray,
    uint32_t *NumberOfElements);
```

and the following parameters:

Parameter	Function invocation (conformance testing model for frameworks)	Bound activity invocation (conformance testing model for applications)
`no_BSPSchemaArray`	input, outbound	input, inbound
the members of the parameter group "BSP schema" (see 9.2.10) with their names prefixed by "`BSPSchema_1_`"	Output, inbound	input, inbound
the members of the parameter group "BSP schema" with their names prefixed by "`BSPSchema_2_`"	Output, inbound	input, inbound
the members of the parameter group "BSP schema" with their names prefixed by "`BSPSchema_3_`"	Output, inbound	input, inbound
the members of the parameter group "BSP schema" with their names prefixed by "`BSPSchema_4_`"	Output, inbound	input, inbound
`no_NumberOfElements`	input, outbound	input, inbound
`NumberOfElements`	output, inbound	input, inbound
`Return`	return, inbound	input, inbound

9.6.2 Constraints on the Parameters

9.6.2.1 An outbound value of `no_BSPSchemaArray` and `no_NumberOfElements` shall be either a valid representation of a boolean (see 7.5) or an empty string. An inbound value shall be the canonical representation of a boolean (see 7.5.2).

9.6.2.2 An outbound value of each parameter group "BSP schema" shall be a valid representation of a value of the native type `BioAPI_BSP_SCHEMA`. An inbound value shall be either the canonical representation of a value of that type, or a set of empty strings.

9.6.2.3 An outbound value of `NumberOfElements` shall be either a valid representation of an integer in the range 0 to 4294967295, or an empty string. An inbound value shall be either the canonical representation of an integer in the same range, or an empty string.

9.6.2.4 An outbound value of `return` shall be a valid representation of an integer in the range 0 to 4294967295. An inbound value shall be the canonical representation of an integer in the same range.

9.6.3 Function Invocation Input

9.6.3.1 If the outbound value of `no_BSPSchemaArray` is "`true`", then the native parameter `BSPSchemaArray` shall be set to NULL, otherwise it shall be set to the address of a variable of type `BioAPI_BSP_SCHEMA*` (a pointer).

9.6.3.2 If the outbound value of `no_NumberOfElements` is "`true`", then the native parameter `NumberOfElements` shall be set to NULL, otherwise it shall be set to the address of a variable of type `uint32_t`.

9.6.4 Function Invocation Output

9.6.4.1 The inbound value of the parameter group "BSP schema" with the prefix "`BSPSchema_`*X*`_`" (with *X*=1, 2, 3, or 4) shall be determined as follows. If the native parameter `BSPSchemaArray` is NULL or the variable pointed to by that native parameter is NULL or the native parameter `NumberOfElements` is NULL or the value of the integer variable pointed to by that native parameter is less than *X*, then the inbound value shall be a set of empty strings. Otherwise, the inbound value shall be the canonical representation (see 9.2.10.5) of the value of type `BioAPI_BSP_SCHEMA` at position *X* in the array pointed to by the variable pointed to by the native parameter `BSPSchemaArray`.

9.6.4.2 The inbound value of `NumberOfElements` shall be determined as follows. If the native parameter with the same name is NULL, then the inbound value shall be an empty string. Otherwise, the inbound value shall be the canonical representation of the integer in the variable of type `uint32_t` pointed to by the native parameter `NumberOfElements`.

9.6.5 Bound Activity Invocation Input

9.6.5.1 The inbound value of `no_BSPSchemaArray` shall be "`true`" if the native parameter `BSPSchemaArray` is NULL, otherwise it shall be "`false`".

9.6.5.2 The inbound value of `no_NumberOfElements` shall be "`true`" if the native parameter `NumberOfElements` is NULL, otherwise it shall be "`false`".

9.6.5.3 The inbound value of `return` shall be the canonical representation of the integer returned by the native function.

9.6.5.4 The inbound values of the other input parameters shall be determined as specified in 9.6.4.

9.6.6 Bound Activity Invocation Output

There are no output parameters.

9.7 BioAPI_BSPLoad

9.7.1 Function Invocation Scheme

This function belongs to the BioAPI interface, and has the following native prototype:

BioAPI_RETURN BioAPI

BioAPI_BSPLoad(

const BioAPI_UUID *BSPUuid,

BioAPI_EventHandler AppNotifyCallback,

void* AppNotifyCallbackCtx);

and the following parameters:

Parameter	**Function invocation (conformance testing model for frameworks)**	**Bound activity invocation (conformance testing model for applications)**
`BSPUuid`	input, outbound	input, inbound
`AppNotifyCallback`	input, outbound	input, inbound
`AppNotifyCallbackCtx`	input, outbound	input, inbound
`return`	return, inbound	input, inbound

9.7.2 Constraints on the Parameters

9.7.2.1 An outbound value of `BSPUuid` shall be a valid representation of a UUID (see 7.6). An inbound value shall be the canonical representation of a UUID (see 7.6.3).

9.7.2.2 An outbound value of `AppNotifyCallback` and `AppNotifyCallbackCtx` shall be either "`0`" or "`*`". An inbound value shall be the canonical representation of an integer (see 7.4.3) in the range 0 to 4294967295.

9.7.2.3 An outbound value of `return` shall be a valid representation of an integer in the range 0 to 4294967295. An inbound value shall be the canonical representation of an integer in the same range.

9.7.3 Function Invocation Input

9.7.3.1 The UUID represented by the outbound value of `BSPUuid` shall be written to a variable of type `BioAPI_UUID`, whose address shall be assigned to the native parameter `BSPUuid`.

9.7.3.2 If the outbound value of `AppNotifyCallback` is "`0`", a NULL pointer value shall be assigned to the native parameter `AppNotifyCallback`. If it is "`*`", a non-NULL pointer value, which is the address of a native function (within the testing component) implementing the standard BioAPI interface function `BioAPI_EventHandler`, shall be assigned to the native parameter `AppNotifyCallback`.

NOTE If the outbound value of `AppNotifyCallback` is not "`0`", any subsequent incoming calls to the standard BioAPI interface function `BioAPI_EventHandler` will result in an invocation of the activity bound to this function, if such a binding exists.

9.7.3.3 If the outbound value of **`AppNotifyCallbackCtx`** is "0", a NULL pointer value shall be assigned to the native parameter **`AppNotifyCallbackCtx`**. If it is "*", a non-NULL pointer value, which is the address of a variable of type **`void*`** set to NULL, shall be assigned to the native parameter **`AppNotifyCallbackCtx`**.

9.7.4 Function Invocation Output

There are no output parameters.

9.7.5 Bound Activity Invocation Input

9.7.5.1 The inbound value of **`BSPUuid`** shall be the canonical representation of the UUID in the variable of type **`BioAPI_UUID`** pointed to by the native parameter **`BSPUuid`**.

9.7.5.2 The inbound value of **`AppNotifyCallback`** and **`AppNotifyCallbackCtx`** shall be the canonical representation of the integer in the native parameter with the same name.

9.7.5.3 The inbound value of **`return`** shall be the canonical representation of the integer returned by the native function.

9.7.6 Bound Activity Invocation Output

There are no output parameters.

9.8 BioAPI_BSPUnload

9.8.1 Function Invocation Scheme

This function belongs to the BioAPI interface, and has the following native prototype:

```
BioAPI_RETURN BioAPI
BioAPI_BSPUnload(
    const BioAPI_UUID *BSPUuid,
    BioAPI_EventHandler AppNotifyCallback,
    void* AppNotifyCallbackCtx);
```

and the following parameters:

Parameter	Function invocation (conformance testing model for frameworks)	Bound activity invocation (conformance testing model for applications)
BSPUuid	input, outbound	input, inbound
AppNotifyCallback	input, outbound	input, inbound
AppNotifyCallbackCtx	input, outbound	input, inbound
return	return, inbound	input, inbound

9.8.2 Constraints on the Parameters

9.8.2.1 An outbound value of `BSPUuid` shall be a valid representation of a UUID (see 7.6). An inbound value shall be the canonical representation of a UUID (see 7.6.3).

9.8.2.2 An outbound value of `AppNotifyCallback` and `AppNotifyCallbackCtx` shall be either "`0`" or "`*`". An inbound value shall be the canonical representation of an integer (see 7.4.3) in the range 0 to 4294967295.

9.8.2.3 An outbound value of `return` shall be a valid representation of an integer in the range 0 to 4294967295. An inbound value shall be the canonical representation of an integer in the same range.

9.8.3 Function Invocation Input

9.8.3.1 The UUID represented by the outbound value of `BSPUuid` shall be written to a variable of type `BioAPI_UUID`, whose address shall be assigned to the native parameter `BSPUuid`.

9.8.3.2 If the outbound value of `AppNotifyCallback` is "`0`", a NULL pointer value shall be assigned to the native parameter `AppNotifyCallback`. If it is "`*`", a non-NULL pointer value, which is the address of a native function (within the testing component) implementing the standard BioAPI interface function `BioAPI_EventHandler`, shall be assigned to the native parameter `AppNotifyCallback`.

9.8.3.3 If the outbound value of `AppNotifyCallbackCtx` is "`0`", a NULL pointer value shall be assigned to the native parameter `AppNotifyCallbackCtx`. If it is "`*`", a non-NULL pointer value, which is the address of a variable of type `void*` set to NULL, shall be assigned to the native parameter `AppNotifyCallbackCtx`.

9.8.4 Function Invocation Output

There are no output parameters.

9.8.5 Bound Activity Invocation Input

9.8.5.1 The inbound value of `BSPUuid` shall be the canonical representation of the UUID in the variable of type `BioAPI_UUID` pointed to by the native parameter `BSPUuid`.

9.8.5.2 The inbound value of `AppNotifyCallback` and `AppNotifyCallbackCtx` shall be the canonical representation of the integer in the native parameter with the same name.

9.8.5.3 The inbound value of `return` shall be the canonical representation of the integer returned by the native function.

9.8.6 Bound Activity Invocation Output

There are no output parameters.

9.9 BioAPI_BSPAttach

9.9.1 Function Invocation Scheme

This function belongs to the BioAPI interface, and has the following native prototype:

BioAPI_RETURN BioAPI

BioAPI_BSPAttach(

const BioAPI_UUID *BSPUuid,

BioAPI_VERSION Version,

const BioAPI_UNIT_LIST_ELEMENT *UnitList,

uint32_t NumUnits,

BioAPI_HANDLE *NewBSPHandle);

and the following parameters:

Parameter	Function invocation (conformance testing model for frameworks)	Bound activity invocation (conformance testing model for applications)
BSPUuid	input, outbound	input, inbound
Version	input, outbound	input, inbound
Unit_1_UnitCategory	input, outbound	input, inbound
Unit_1_UnitID	input, outbound	input, inbound
Unit_2_UnitCategory	input, outbound	input, inbound
Unit_2_UnitID	input, outbound	input, inbound
Unit_3_UnitCategory	input, outbound	input, inbound
Unit_3_UnitID	input, outbound	input, inbound
Unit_4_UnitCategory	input, outbound	input, inbound
Unit_4_UnitID	input, outbound	input, inbound
NumUnits	input, outbound	input, inbound
no_NewBSPHandle	input, outbound	input, inbound
NewBSPHandle	output, inbound	input, inbound
return	return, inbound	input, inbound

9.9.2 Constraints on the Parameters

9.9.2.1 An outbound value of `BSPUuid` shall be a valid representation of a UUID (see 7.6). An inbound value shall be the canonical representation of a UUID (see 7.6.3).

9.9.2.2 An outbound value of `Version` shall be a valid representation of an integer (see 7.4) in the range 0 to 255. An inbound value shall be the canonical representation of an integer (see 7.4.3) in the same range.

9.9.2.3 An outbound value of `Unit_`*X*`_UnitCategory` and `Unit_`*X*`_UnitID` (with *X*=1, 2, 3, or 4) shall be either a valid representation of an integer in the range 0 to 4294967295, or an empty string. An inbound value shall be either the canonical representation of an integer in the same range, or an empty string.

9.9.2.4 An outbound value of `NumUnits` shall be a valid representation of an integer in the range 0 to 4. An inbound value shall be the canonical representation of an integer in the range 0 to 4294967295.

9.9.2.5 An outbound value of `no_NewBSPHandle` shall be either a valid representation of a boolean (see 7.5) or an empty string. An inbound value shall be the canonical representation of a boolean (see 7.5.2).

9.9.2.6 An outbound value of `NewBSPHandle` shall be either a valid representation of an integer in the range 0 to 4294967295, or an empty string. An inbound value shall be either the canonical representation of an integer in the same range, or an empty string.

9.9.2.7 An outbound value of `return` shall be a valid representation of an integer in the range 0 to 4294967295. An inbound value shall be the canonical representation of an integer in the same range.

9.9.3 Function Invocation Input

9.9.3.1 The UUID represented by the outbound value of `BSPUuid` shall be written to a variable of type `BioAPI_UUID`, whose address shall be assigned to the native parameter `BSPUuid`.

9.9.3.2 The integer represented by the outbound value of `Version` and `NumUnits` shall be assigned to the native parameter with the same name.

9.9.3.3 The integer represented by the outbound value of `Unit_`*X*`_UnitCategory` and `Unit_`*X*`_UnitID` (with *X*=1, 2, 3, or 4), or zero if that value is an empty string, shall be written to the fields `UnitCategory` and `UnitID` (respectively) of the element at position *X* of an array of four elements of type `BioAPI_UNIT_LIST_ELEMENT`. The address of that array shall be assigned to the native parameter `UnitList`.

9.9.3.4 If the outbound value of `no_NewBSPHandle` is "`true`", then the native parameter `NewBSPHandle` shall be set to NULL, otherwise it shall be set to the address of a variable of type `BioAPI_HANDLE`.

9.9.4 Function Invocation Output

The inbound value of `NewBSPHandle` shall be determined as follows. If the native parameter with the same name is NULL, then the inbound value shall be an empty string. Otherwise, the inbound value shall be the canonical representation of the integer in the variable of type `BioAPI_HANDLE` pointed to by the native parameter `NewBSPHandle`.

9.9.5 Bound Activity Invocation Input

9.9.5.1 The inbound value of `BSPUuid` shall be the canonical representation of the UUID in the variable of type `BioAPI_UUID` pointed to by the native parameter `BSPUuid`.

9.9.5.2 The inbound value of `Version` and `NumUnits` shall be the canonical representation of the integer in the native parameter with the same name.

9.9.5.3 The inbound value of `Unit_`*X*`_UnitCategory` and `Unit_`*X*`_UnitID` (with *X*=1, 2, 3, or 4) shall be determined as follows. If the native parameter `NumUnits` is less than *X*, then the inbound value shall be an empty string. Otherwise, the inbound value shall be the canonical representation of the integer in the fields `UnitCategory` and `UnitID` (respectively) of the element at position *X* of the array of elements of type `BioAPI_UNIT_LIST_ELEMENT` pointed to by the native parameter `UnitList`.

9.9.5.4 The inbound value of `no_NewBSPHandle` shall be "`true`" if the native parameter `NewBSPHandle` is NULL, otherwise it shall be "`false`".

9.9.5.5 The inbound value of `return` shall be the canonical representation of the integer returned by the native function.

9.9.5.6 The inbound values of the other input parameters shall be determined as specified in 9.9.4.

9.9.6 Bound Activity Invocation Output

There are no output parameters.

9.10 BioAPI_BSPDetach

9.10.1 Function Invocation Scheme

This function belongs to the BioAPI interface, and has the following native prototype:

```
BioAPI_RETURN BioAPI
BioAPI_BSPDetach(
    BioAPI_HANDLE BSPHandle);
```

and the following parameters:

Parameter	Function invocation (conformance testing model for frameworks)	Bound activity invocation (conformance testing model for applications)
BSPHandle	input, outbound	input, inbound
Return	return, inbound	input, inbound

9.10.2 Constraints on the Parameters

9.10.2.1 An outbound value of `BSPHandle` shall be a valid representation of an integer (see 7.4) in the range 0 to 4294967295. An inbound value shall be the canonical representation of an integer (see 7.4.3) in the same range.

9.10.2.2 An outbound value of `return` shall be a valid representation of an integer in the range 0 to 4294967295. An inbound value shall be the canonical representation of an integer in the same range.

9.10.3 Function Invocation Input

The integer represented by the outbound value of `BSPHandle` shall be assigned to the native parameter with the same name.

9.10.4 Function Invocation Output

There are no output parameters.

9.10.5 Bound Activity Invocation Input

9.10.5.1 The inbound value of `BSPHandle` shall be the canonical representation of the integer in the native parameter with the same name.

9.10.5.2 The inbound value of `return` shall be the canonical representation of the integer returned by the native function.

9.10.6 Bound Activity Invocation Output

There are no output parameters.

9.11 BioAPI_QueryUnits

9.11.1 Function Invocation Scheme

This function belongs to the BioAPI interface, and has the following native prototype:

```
BioAPI_RETURN BioAPI
BioAPI_QueryUnits(
    const BioAPI_UUID *BSPUuid,
    BioAPI_UNIT_SCHEMA **UnitSchemaArray,
    uint32_t *NumberOfElements);
```

and the following parameters:

Parameter	Function invocation (conformance testing model for frameworks)	Bound activity invocation (conformance testing model for applications)
`BSPUuid`	input, outbound	input, inbound
`no_UnitSchemaArray`	input, outbound	input, inbound
the members of the parameter group "Unit schema" (see 9.2.12) with their names prefixed by "`UnitSchema_1_`"	output, inbound	input, inbound
the members of the parameter group "Unit schema" with their names prefixed by "`UnitSchema_2_`"	output, inbound	input, inbound

the members of the parameter group "Unit schema" with their names prefixed by "UnitSchema_3_"	output, inbound	input, inbound
the members of the parameter group "Unit schema" with their names prefixed by "UnitSchema_4_"	output, inbound	input, inbound
no_NumberOfElements	input, outbound	input, inbound
NumberOfElements	output, inbound	input, inbound
Return	return, inbound	input, inbound

9.11.2 Constraints on the Parameters

9.11.2.1 An outbound value of **BSPUuid** shall be a valid representation of a UUID (see 7.6). An inbound value shall be the canonical representation of a UUID (see 7.6.3).

9.11.2.2 An outbound value of **no_UnitSchemaArray** and **no_NumberOfElements** shall be either a valid representation of a boolean (see 7.5) or an empty string. An inbound value shall be the canonical representation of a boolean (see 7.5.2).

9.11.2.3 An outbound value of each parameter group "Unit schema" shall be a valid representation of a value of the native type **BioAPI_UNIT_SCHEMA**. An inbound value shall be either the canonical representation of a value of that type, or a set of empty strings.

9.11.2.4 An outbound value of **NumberOfElements** shall be either a valid representation of an integer in the range 0 to 4294967295, or an empty string. An inbound value shall be either the canonical representation of an integer in the same range, or an empty string.

9.11.2.5 An outbound value of **return** shall be a valid representation of an integer in the range 0 to 4294967295. An inbound value shall be the canonical representation of an integer in the same range.

9.11.3 Function Invocation Input

9.11.3.1 The UUID represented by the outbound value of **BSPUuid** shall be written to a variable of type **BioAPI_UUID**, whose address shall be assigned to the native parameter **BSPUuid**.

9.11.3.2 If the outbound value of **no_UnitSchemaArray** is "**true**", then the native parameter **UnitSchemaArray** shall be set to NULL, otherwise it shall be set to the address of a variable of type **BioAPI_UNIT_SCHEMA*** (a pointer).

9.11.3.3 If the outbound value of **no_NumberOfElements** is "**true**", then the native parameter **NumberOfElements** shall be set to NULL, otherwise it shall be set to the address of a variable of type **uint32_t**.

9.11.4 Function Invocation Output

9.11.4.1 The inbound value of the parameter group "Unit schema" with the prefix "`UnitSchema_`*X*`_`" (with *X*=1, 2, 3, or 4) shall be determined as follows. If the native parameter `UnitSchemaArray` is NULL or the variable pointed to by that native parameter is NULL or the native parameter `NumberOfElements` is NULL or the value of the integer variable pointed to by that native parameter is less than *X*, then the inbound value shall be a set of empty strings. Otherwise, the inbound value be shall the canonical representation (see 9.2.14.5) of the value of type `BioAPI_UNIT_SCHEMA` at position *X* in the array pointed to by the variable pointed to by the native parameter `UnitSchemaArray`.

9.11.4.2 The inbound value of `NumberOfElements` shall be determined as follows. If the native parameter with the same name is NULL, then the inbound value shall be an empty string. Otherwise, the inbound value shall be the canonical representation of the integer in the variable of type `uint32_t` pointed to by the native parameter `NumberOfElements`.

9.11.5 Bound Activity Invocation Input

9.11.5.1 The inbound value of `BSPUuid` shall be the canonical representation of the UUID in the variable of type `BioAPI_UUID` pointed to by the native parameter `BSPUuid`.

9.11.5.2 The inbound value of `no_UnitSchemaArray` shall be "`true`" if the native parameter `UnitSchemaArray` is NULL, otherwise it shall be "`false`".

9.11.5.3 The inbound value of `no_NumberOfElements` shall be "`true`" if the native parameter `NumberOfElements` is NULL, otherwise it shall be "`false`".

9.11.5.4 The inbound value of `return` shall be the canonical representation of the integer returned by the native function.

9.11.5.5 The inbound values of the other input parameters shall be determined as specified in 9.11.4.

9.11.6 Bound Activity Invocation Output

There are no output parameters.

9.12 BioAPI_EnumBFPs

9.12.1 Function Invocation Scheme

This function belongs to the BioAPI interface, and has the following native prototype:

```
BioAPI_RETURN BioAPI
BioAPI_EnumBFPs(
    BioAPI_BFP_SCHEMA **BFPSchemaArray,
    uint32_t *NumberOfElements);
```

and the following parameters:

Parameter	Function invocation (conformance testing model for frameworks)	Bound activity invocation (conformance testing model for applications)
`no_BFPSchemaArray`	input, outbound	input, inbound
the members of the parameter group "BFP schema" (see 9.2.11) with their names prefixed by "`BFPSchema_1_`"	output, inbound	input, inbound
the members of the parameter group "BFP schema" with their names prefixed by "`BFPSchema_2_`"	output, inbound	input, inbound
the members of the parameter group "BFP schema" with their names prefixed by "`BFPSchema_3_`"	output, inbound	input, inbound
the members of the parameter group "BFP schema" with their names prefixed by "`BFPSchema_4_`"	output, inbound	input, inbound
`no_NumberOfElements`	input, outbound	input, inbound
`NumberOfElements`	output, inbound	input, inbound
`Return`	return, inbound	input, inbound

9.12.2 Constraints on the Parameters

9.12.2.1 An outbound value of `no_BFPSchemaArray` and `no_NumberOfElements` shall be either a valid representation of a boolean (see 7.5) or an empty string. An inbound value shall be the canonical representation of a boolean (see 7.5.2).

9.12.2.2 An outbound value of each parameter group "BFP schema" shall be a valid representation of a value of the native type `BioAPI_BFP_SCHEMA`. An inbound value shall be either the canonical representation of a value of that type, or a set of empty strings.

9.12.2.3 An outbound value of `NumberOfElements` shall be either a valid representation of an integer in the range 0 to 4294967295, or an empty string. An inbound value shall be either the canonical representation of an integer in the same range, or an empty string.

9.12.2.4 An outbound value of `return` shall be a valid representation of an integer in the range 0 to 4294967295. An inbound value shall be the canonical representation of an integer in the same range.

9.12.3 Function Invocation Input

9.12.3.1 If the outbound value of `no_BFPSchemaArray` is "`true`", then the native parameter `BFPSchemaArray` shall be set to NULL, otherwise it shall be set to the address of a variable of type `BioAPI_BFP_SCHEMA*` (a pointer).

9.12.3.2 If the outbound value of `no_NumberOfElements` is "`true`", then the native parameter `NumberOfElements` shall be set to NULL, otherwise it shall be set to the address of a variable of type `uint32_t`.

9.12.4 Function Invocation Output

9.12.4.1 The inbound value of the parameter group "BFP schema" with the prefix "`BFPSchema_`*X*`_`" (with *X*=1, 2, 3, or 4) shall be determined as follows. If the native parameter `BFPSchemaArray` is NULL or the variable pointed to by that native parameter is NULL or the native parameter `NumberOfElements` is NULL or the value of the integer variable pointed to that native parameter is less than *X*, then the inbound value shall be a set of empty strings. Otherwise, the inbound value shall be the canonical representation (see 9.2.11.5) of the value of type `BioAPI_BFP_SCHEMA` at position *X* in the array pointed to by the variable pointed to by the native parameter `BFPSchemaArray`.

9.12.4.2 The inbound value of `NumberOfElements` shall be determined as follows. If the native parameter with the same name is NULL, then the inbound value shall be an empty string. Otherwise, the inbound value shall be the canonical representation of the integer in the variable of type `uint32_t` pointed to by the native parameter `NumberOfElements`.

9.12.5 Bound Activity Invocation Input

9.12.5.1 The inbound value of `no_BFPSchemaArray` shall be "`true`" if the native parameter `BFPSchemaArray` is NULL, otherwise it shall be "`false`".

9.12.5.2 The inbound value of `no_NumberOfElements` shall be "`true`" if the native parameter `NumberOfElements` is NULL, otherwise it shall be "`false`".

9.12.5.3 The inbound value of `return` shall be the canonical representation of the integer returned by the native function.

9.12.5.4 The inbound values of the other input parameters shall be determined as specified in 9.12.4.

9.12.6 Bound Activity Invocation Output

There are no output parameters.

9.13 BioAPI_QueryBFPs

9.13.1 Function Invocation Scheme

This function belongs to the BioAPI interface, and has the following native prototype:

BioAPI_RETURN BioAPI

BioAPI_QueryBFPs(

const BioAPI_UUID *BSPUuid,

BioAPI_BFP_LIST_ELEMENT **BFPList,

uint32_t *NumberOfElements);

and the following parameters:

Parameter	**Function invocation (conformance testing model for frameworks)**	**Bound activity invocation (conformance testing model for applications)**
`BSPHandle`	input, outbound	input, inbound
`no_BFPList`	input, outbound	input, inbound
`BFP_1_BFPCategory`	output, inbound	input, inbound
`BFP_1_BFPUuid`	output, inbound	input, inbound
`BFP_2_BFPCategory`	output, inbound	input, inbound
`BFP_2_BFPUuid`	output, inbound	input, inbound
`BFP_3_BFPCategory`	output, inbound	input, inbound
`BFP_3_BFPUuid`	output, inbound	input, inbound
`BFP_4_BFPCategory`	output, inbound	input, inbound
`BFP_4_BFPUuid`	output, inbound	input, inbound
`no_NumberOfElements`	input, outbound	input, inbound
`NumberOfElements`	output, inbound	input, inbound
`return`	return, inbound	input, inbound

9.13.2 Constraints on the Parameters

9.13.2.1 An outbound value of `BSPHandle` shall be a valid representation of an integer (see 7.4) in the range 0 to 4294967295. An inbound value shall be the canonical representation of an integer (see 7.4.3) in the same range.

9.13.2.2 An outbound value of `no_BFPList` and `no_NumberOfElements` shall be either a valid representation of a boolean (see 7.5) or an empty string. An inbound value shall be the canonical representation of a boolean (see 7.5.2).

9.13.2.3 An outbound value of `NumberOfElements` shall be either a valid representation of an integer in the range 0 to 4294967295, or an empty string. An inbound value shall be either the canonical representation of an integer in the same range, or an empty string.

9.13.2.4 An outbound value of `return` shall be a valid representation of an integer in the range 0 to 4294967295. An inbound value shall be the canonical representation of an integer in the same range.

9.13.3 Function Invocation Input

9.13.3.1 The integer represented by the outbound value of `BSPHandle` shall be assigned to the native parameter with the same name.

9.13.3.2 If the outbound value of `no_BFPList` is "`true`", then the native parameter `BFPList` shall be set to NULL, otherwise it shall be set to the address of a variable of type `BioAPI_BFP_LIST_ELEMENT*` (a pointer).

9.13.3.3 If the outbound value of `no_NumberOfElements` is "`true`", then the native parameter `NumberOfElements` shall be set to NULL, otherwise it shall be set to the address of a variable of type `uint32_t`.

9.13.4 Function Invocation Output

9.13.4.1 The inbound value of `BFP_`*X*`_BFPCategory` and `BFP_`*X*`_BFPUuid` (with *X*=1, 2, 3, or 4) shall be determined as follows. If the native parameter `BFPList` is NULL or the variable pointed to by that native parameter is NULL or the native parameter `NumberOfElements` is NULL or the value of the integer variable pointed to by that native parameter is less than *X*, then the inbound value shall be an empty string. Otherwise, the inbound value shall be the canonical representation of the integer in the fields `BFPCategory` and `BFPUuid` (respectively) of the element at position *X* of the array of elements of type `BioAPI_BFP_LIST_ELEMENT` pointed to by the variable pointed to by the native parameter `BFPList`.

9.13.4.2 The inbound value of `NumberOfElements` shall be determined as follows. If the native parameter with the same name is NULL, then the inbound value shall be an empty string. Otherwise, the inbound value shall be the canonical representation of the integer in the variable of type `uint32_t` pointed to by the native parameter `NumberOfElements`.

9.13.5 Bound Activity Invocation Input

9.13.5.1 The inbound value of `BSPHandle` shall be the canonical representation of the integer in the native parameter with the same name.

9.13.5.2 The inbound value of `no_BFPList` shall be "`true`" if the native parameter `BFPList` is NULL, otherwise it shall be "`false`".

9.13.5.3 The inbound value of `no_NumberOfElements` shall be "`true`" if the native parameter `NumberOfElements` is NULL, otherwise it shall be "`false`".

9.13.5.4 The inbound value of `return` shall be the canonical representation of the integer returned by the native function.

9.13.5.5 The inbound values of the other input parameters shall be determined as specified in 9.13.4.

9.13.6 Bound Activity Invocation Output

There are no output parameters.

9.14 BioAPI_ControlUnit

9.14.1 Function Invocation Scheme

This function belongs to the BioAPI interface, and has the following native prototype:

```
BioAPI_RETURN BioAPI
BioAPI_ControlUnit(
    BioAPI_HANDLE BSPHandle,
    BioAPI_UNIT_ID UnitID,
    uint32_t ControlCode,
    const BioAPI_DATA *InputData,
    BioAPI_DATA *OutputData);
```

and the following parameters:

Parameter	**Function invocation (conformance testing model for frameworks)**	**Bound activity invocation (conformance testing model for applications)**
`BSPHandle`	input, outbound	input, inbound
`UnitID`	input, outbound	input, inbound
`ControlCode`	input, outbound	input, inbound
`InputData`	input, outbound	input, inbound
`no_OutputData`	input, outbound	input, inbound
`OutputData`	output, inbound	input, inbound
`return`	return, inbound	input, inbound

9.14.2 Constraints on the Parameters

9.14.2.1 An outbound value of `BSPHandle`, `UnitID`, and `ControlCode` shall be a valid representation of an integer (see 7.4) in the range 0 to 4294967295. An inbound value shall be the canonical representation of an integer (see 7.4.3) in the same range.

9.14.2.2 An outbound value of `InputData` and `OutputData` shall be a valid representation of an octet string (see 7.7). An inbound value shall be the canonical representation of an octet string (see 7.7.2).

9.14.2.3 An outbound value of `no_OutputData` shall be either a valid representation of a boolean (see 7.5), or an empty string. An inbound value shall be the canonical representation of a boolean (see 7.5.2).

9.14.2.4 An outbound value of `return` shall be a valid representation of an integer in the range 0 to 4294967295. An inbound value shall be the canonical representation of an integer in the same range.

9.14.3 Function Invocation Input

9.14.3.1 The integer represented by the outbound value of `BSPHandle`, `UnitID`, and `ControlCode` shall be assigned to the native parameter with the same name.

9.14.3.2 If the outbound value of **InputData** is an empty string, then the native parameter with the same name shall be set to NULL. Otherwise, the octet string represented by the outbound value shall be written to a memory block of sufficient size. The address and length of that memory block shall be written to the fields **Data** and **Length** (respectively) of a variable of type **BioAPI_DATA**, whose address shall be assigned to the native parameter **InputData.**

9.14.3.3 If the outbound value of `no_OutputData` is "`true`", then the native parameter `OutputData` shall be set to NULL, otherwise it shall be set to the address of a variable of type `BioAPI_DATA`.

9.14.4 Function Invocation Output

The inbound value of `OutputData` shall be determined as follows. If the native parameter with the same name is NULL, then the inbound value shall be an empty string. Otherwise, the inbound value shall be the canonical representation of the octet string in the memory block whose address and length are in the fields `Data` and `Length` (respectively) of the variable of type `BioAPI_DATA` pointed to by the native parameter `OutputData`.

9.14.5 Bound Activity Invocation Input

9.14.5.1 The inbound value of `BSPHandle`, `UnitID`, and `ControlCode` shall be the canonical representation of the integer in the native parameter with the same name.

9.14.5.2 The inbound value of `InputData` shall be determined as follows. If the native parameter with the same name is NULL, then the inbound value shall be an empty string. Otherwise, the inbound value shall be the canonical representation (see 7.7.2) of the octet string in the memory block whose address and length are in the fields `Data` and `Length` (respectively) of the variable of type `BioAPI_DATA` pointed to by the native parameter `InputData`

9.14.5.3 The inbound value of `no_OutputData` shall be "`true`" if the native parameter `OutputData` is NULL, otherwise it shall be "`false`".

9.14.5.4 The inbound value of `return` shall be the canonical representation of the integer returned by the native function.

9.14.5.5 The inbound values of the other input parameters shall be determined as specified in 8.14.4.

9.14.6 Bound Activity Invocation Output

There are no output parameters.

9.15 BioAPI_FreeBIRHandle

9.15.1 Function Invocation Scheme

This function belongs to the BioAPI interface, and has the following native prototype:

BioAPI_RETURN BioAPI

BioAPI_FreeBIRHandle(

BioAPI_HANDLE BSPHandle,

BioAPI_BIR_HANDLE Handle);

and the following parameters:

Parameter	**Function invocation (conformance testing model for frameworks)**	**Bound activity invocation (conformance testing model for applications)**
`BSPHandle`	input, outbound	input, inbound
`Handle`	input, outbound	input, inbound
`return`	return, inbound	input, inbound

9.15.2 Constraints on the Parameters

9.15.2.1 An outbound value of `BSPHandle` shall be a valid representation of an integer (see 7.4) in the range 0 to 4294967295. An inbound value shall be the canonical representation of an integer (see 7.4.3) in the same range.

9.15.2.2 An outbound value of `Handle` shall be a valid representation of an integer in the range -2147483648 to 2147483647. An inbound value shall be the canonical representation of an integer in the same range.

9.15.2.3 An outbound value of `return` shall be a valid representation of an integer in the range 0 to 4294967295. An inbound value shall be the canonical representation of an integer in the same range.

9.15.3 Function Invocation Input

The integer represented by the outbound value of `BSPHandle` and `Handle` shall be assigned to the native parameter with the same name.

9.15.4 Function Invocation Output

There are no output parameters.

9.15.5 Bound Activity Invocation Input

9.15.5.1 The inbound value of `BSPHandle` and `Handle` shall be the canonical representation of the integer in the native parameter with the same name.

9.15.5.2 The inbound value of `return` shall be the canonical representation of the integer returned by the native function.

9.15.6 Bound Activity Invocation Output

There are no output parameters.

9.16 BioAPI_GetBIRFromHandle

9.16.1 Function Invocation Scheme

This function belongs to the BioAPI interface, and has the following native prototype:

```
BioAPI_RETURN BioAPI
BioAPI_GetBIRFromHandle(
    BioAPI_HANDLE BSPHandle,
    BioAPI_BIR_HANDLE Handle,
    BioAPI_BIR *BIR);
```

and the following parameters:

Parameter	**Function invocation (conformance testing model for frameworks)**	**Bound activity invocation (conformance testing model for applications)**
`BSPHandle`	input, outbound	input, inbound
`Handle`	input, outbound	input, inbound
`no_BIR`	input, outbound	input, inbound
the members of the parameter group "BIR" (see 9.2.14)	output, inbound	input, inbound
`return`	return, inbound	input, inbound

9.16.2 Constraints on the Parameters

9.16.2.1 An outbound value of **`BSPHandle`** shall be a valid representation of an integer (see 7.4) in the range 0 to 4294967295. An inbound value shall be the canonical representation of an integer (see 7.4.3) in the same range.

9.16.2.2 An outbound value of **`Handle`** shall be a valid representation of an integer in the range -2147483648 to 2147483647. An inbound value shall be the canonical representation of an integer in the same range.

9.16.2.3 An outbound value of **`no_BIR`** shall be either a valid representation of a boolean (see 7.5) or an empty string. An inbound value shall be the canonical representation of a boolean (see 7.5.2).

9.16.2.4 An outbound value of the parameter group "BIR" shall be a valid representation of a value of the native type **`BioAPI_BIR`**. An inbound value shall be either the canonical representation of a value of that type, or a set of empty strings.

9.16.2.5 An outbound value of **`return`** shall be a valid representation of an integer in the range 0 to 4294967295. An inbound value shall be the canonical representation of an integer in the same range.

9.16.3 Function Invocation Input

9.16.3.1 The integer represented by the outbound value of **`BSPHandle`** and **`Handle`** shall be assigned to the native parameter with the same name.

9.16.3.2 If the outbound value of **`no_BIR`** is "**`true`**", then the native parameter **`BIR`** shall be set to NULL, otherwise it shall be set to the address of a variable of type **`BioAPI_BIR`**.

9.16.4 Function Invocation Output

The inbound value of the parameter group "BIR" shall be determined as follows. If the native parameter **`BIR`** is NULL, then the inbound value shall be a set of empty strings. Otherwise, the inbound value shall be the canonical representation (see 9.2.14.5) of the value of type **`BioAPI_BIR`** pointed to by the native parameter **`BIR`**.

9.16.5 Bound Activity Invocation Input

9.16.5.1 The inbound value of **`BSPHandle`** and **`Handle`** shall be the canonical representation of the integer in the native parameter with the same name.

9.16.5.2 The inbound value of **`no_BIR`** shall be "**`true`**" if the native parameter **`BIR`** is NULL, otherwise it shall be "**`false`**".

9.16.5.3 The inbound value of **`return`** shall be the canonical representation of the integer returned by the native function.

9.16.5.4 The inbound values of the other input parameters shall be determined as specified in 9.16.4.

9.16.6 Bound Activity Invocation Output

There are no output parameters.

9.17 BioAPI_GetHeaderFromHandle

9.17.1 Function Invocation Scheme

This function belongs to the BioAPI interface, and has the following native prototype:

BioAPI_RETURN BioAPI
BioAPI_GetHeaderFromHandle(
  BioAPI_HANDLE BSPHandle,
  BioAPI_BIR_HANDLE Handle,
  BioAPI_BIR_HEADER *Header);

and the following parameters:

Parameter	Function invocation (conformance testing model for frameworks)	Bound activity invocation (conformance testing model for applications)
`BSPHandle`	input, outbound	input, inbound
`Handle`	input, outbound	input, inbound
`no_Header`	input, outbound	input, inbound
the members of the parameter group "BIR header" (see 9.2.13)	output, inbound	input, inbound
`return`	return, inbound	input, inbound

9.17.2 Constraints on the Parameters

9.17.2.1 An outbound value of `BSPHandle` shall be a valid representation of an integer (see 7.4) in the range 0 to 4294967295. An inbound value shall be the canonical representation of an integer (see 7.4.3) in the same range.

9.17.2.2 An outbound value of `Handle` shall be a valid representation of an integer in the range -2147483648 to 2147483647. An inbound value shall be the canonical representation of an integer in the same range.

9.17.2.3 An outbound value of `no_Header` shall be either a valid representation of a boolean (see 7.5) or an empty string. An inbound value shall be the canonical representation of a boolean (see 7.5.2).

9.17.2.4 An outbound value of the parameter group "BIR header" shall be a valid representation of a value of the native type `BioAPI_BIR_HEADER`. An inbound value shall be either the canonical representation of a value of that type, or a set of empty strings.

9.17.2.5 An outbound value of `return` shall be a valid representation of an integer in the range 0 to 4294967295. An inbound value shall be the canonical representation of an integer in the same range.

9.17.3 Function Invocation Input

9.17.3.1 The integer represented by the outbound value of `BSPHandle` and `Handle` shall be assigned to the native parameter with the same name.

9.17.3.2 If the outbound value of `no_Header` is "`true`", then the native parameter `Header` shall be set to NULL, otherwise it shall be set to the address of a variable of type `BioAPI_BIR_HEADER`.

9.17.4 Function Invocation Output

The inbound value of the parameter group "BIR header" shall be determined as follows. If the native parameter `Header` is NULL, then the inbound value shall be a set of empty strings. Otherwise, the inbound value shall be the canonical representation (see 9.2.13.5) of the value of type `BioAPI_BIR_HEADER` pointed to by the native parameter `Header`.

9.17.5 Bound Activity Invocation Input

9.17.5.1 The inbound value of `BSPHandle` and `Handle` shall be the canonical representation of the integer in the native parameter with the same name.

9.17.5.2 The inbound value of `no_Header` shall be "`true`" if the native parameter `Header` is NULL, otherwise it shall be "`false`".

9.17.5.3 The inbound value of `return` shall be the canonical representation of the integer returned by the native function.

9.17.5.4 The inbound values of the other input parameters shall be determined as specified in 9.17.4.

9.17.6 Bound Activity Invocation Output

There are no output parameters.

9.18 BioAPI_EnableEvents

9.18.1 Function Invocation Scheme

This function belongs to the BioAPI interface, and has the following native prototype:

BioAPI_RETURN BioAPI

BioAPI_EnableEvents(

BioAPI_HANDLE BSPHandle,

BioAPI_EVENT_MASK Events);

and the following parameters:

Parameter	**Function invocation (conformance testing model for frameworks)**	**Bound activity invocation (conformance testing model for applications)**
`BSPHandle`	input, outbound	input, inbound
the members of the parameter group "Events" (see 9.2.4)	input, outbound	input, inbound
`return`	return, inbound	input, inbound

9.18.2 Constraints on the Parameters

9.18.2.1 An outbound value of `BSPHandle` shall be a valid representation of an integer (see 7.4) in the range 0 to 4294967295. An inbound value shall be the canonical representation of an integer (see 7.4.3) in the same range.

9.18.2.2 An outbound value of the parameter group "Events" shall be a valid representation of a value of the native type `BioAPI_EVENT_MASK`. An inbound value shall be the canonical representation of a value of that type.

9.18.2.3 An outbound value of `return` shall be a valid representation of an integer in the range 0 to 4294967295. An inbound value shall be the canonical representation of an integer in the same range.

9.18.3 Function Invocation Input

9.18.3.1 The integer represented by the outbound value of `BSPHandle` shall be assigned to the native parameter with the same name.

9.18.3.2 The value of type `BioAPI_EVENT_MASK` represented (see 9.2.4.4) by the outbound value of the parameter group "Events" shall be assigned to the native parameter `Events`.

9.18.4 Function Invocation Output

There are no output parameters.

9.18.5 Bound Activity Invocation Input

9.18.5.1 The inbound value of `BSPHandle` shall be the canonical representation of the integer in the native parameter with the same name.

9.18.5.2 The inbound value of the parameter group "Events" shall be the canonical representation (see 9.2.4.5) of the value of type `BioAPI_EVENT_MASK` in the native parameter `Events`.

9.18.5.3 The inbound value of `return` shall be the canonical representation of the integer returned by the native function.

9.18.6 Bound Activity Invocation Output

There are no output parameters.

9.19 BioAPI_SetGUICallbacks

9.19.1 Function Invocation Scheme

This function belongs to the BioAPI interface, and has the following native prototype:

```
BioAPI_RETURN BioAPI
BioAPI_SetGUICallbacks(
    BioAPI_HANDLE BSPHandle,
    BioAPI_GUI_STREAMING_CALLBACK GuiStreamingCallback,
    void *GuiStreamingCallbackCtx,
    BioAPI_GUI_STATE_CALLBACK GuiStateCallback,
    void *GuiStateCallbackCtx);
```

and the following parameters:

Parameter	Function invocation (conformance testing model for frameworks)	Bound activity invocation (conformance testing model for applications)
`BSPHandle`	input, outbound	input, inbound
`GuiStreamingCallback`	input, outbound	input, inbound
`GuiStreamingCallbackCtx`	input, outbound	input, inbound
`GuiStateCallback`	input, outbound	input, inbound
`GuiStateCallbackCtx`	input, outbound	input, inbound
`return`	return, inbound	input, inbound

9.19.2 Constraints on the Parameters

9.19.2.1 An outbound value of `BSPHandle` shall be a valid representation of an integer (see 7.4) in the range 0 to 4294967295. An inbound value shall be the canonical representation of an integer (see 7.4.3) in the same range.

9.19.2.2 An outbound value of `GuiStreamingCallback`, `GuiStreamingCallbackCtx`, `GuiStateCallback`, and `GuiStateCallbackCtx` shall be either "`0`" or "`*`". An inbound value shall be the canonical representation of an integer in the range 0 to 4294967295.

9.19.2.3 An outbound value of `return` shall be a valid representation of an integer in the range 0 to 4294967295. An inbound value shall be the canonical representation of an integer in the same range.

9.19.3 Function Invocation Input

9.19.3.1 The integer represented by the outbound value of `BSPHandle` shall be assigned to the native parameter with the same name.

9.19.3.2 If the outbound value of `GuiStreamingCallback` is "`0`", a NULL pointer value shall be assigned to the native parameter `GuiStreamingCallback`. If it is "`*`", a non-NULL pointer value, which is the address of a native function (within the testing component) implementing the standard BioAPI interface function `BioSPI_GUI_STREAMING_CALLBACK`, shall be assigned to the native parameter `GuiStreamingCallback`.

NOTE If the outbound value of `GuiStreamingCallback` is not "`0`", any subsequent incoming calls to the standard BioAPI interface function `BioSPI_GUI_STREAMING_CALLBACK` will result in an invocation of the activity bound to this function, if such a binding exists.

9.19.3.3 If the outbound value of `GuiStreamingCallbackCtx` is "`0`", a NULL pointer value shall be assigned to the native parameter `GuiStreamingCallbackCtx`. If it is "`*`", a non-NULL pointer value, which is the address of a variable of type `void*` set to NULL, shall be assigned to the native parameter `GuiStreamingCallbackCtx`.

9.19.3.4 If the outbound value of `GuiStateCallback` is "`0`", a NULL pointer value shall be assigned to the native parameter `GuiStateCallback`. If it is "`*`", a non-NULL pointer value, which is the address of a native function (within the testing component) implementing the standard BioAPI interface function `BioSPI_GUI_STATE_CALLBACK`, shall be assigned to the native parameter `GuiStateCallback`.

NOTE If the outbound value of `GuiStateCallback` is not "`0`", any subsequent incoming calls to the standard BioAPI interface function `BioSPI_GUI_STATE_CALLBACK` will result in an invocation of the activity bound to this function, if such a binding exists.

9.19.3.5 If the outbound value of `GuiStateCallbackCtx` is "`0`", a NULL pointer value shall be assigned to the native parameter `GuiStateCallbackCtx`. If it is "`*`", a non-NULL pointer value, which is the address of a variable of type `void*` set to NULL, shall be assigned to the native parameter `GuiStateCallbackCtx`.

9.19.4 Function Invocation Output

There are no output parameters.

9.19.5 Bound Activity Invocation Input

9.19.5.1 The inbound value of `BSPHandle` shall be the canonical representation of the integer in the native parameter with the same name.

9.19.5.2 The inbound value of `GuiStreamingCallback`, `GuiStreamingCallbackCtx`, `GuiStateCallback`, and `GuiStateCallbackCtx` shall be the canonical representation of the integer in the native parameter with the same name.

9.19.5.3 The inbound value of `return` shall be the canonical representation of the integer returned by the native function.

9.19.6 Bound Activity Invocation Output

There are no output parameters.

9.20 BioAPI_Capture

9.20.1 Function Invocation Scheme

This function belongs to the BioAPI interface, and has the following native prototype:

```
BioAPI_RETURN BioAPI
BioAPI_Capture(
    BioAPI_HANDLE BSPHandle,
    BioAPI_BIR_PURPOSE Purpose,
    BioAPI_BIR_SUBTYPE Subtype,
    const BioAPI_BIR_BIOMETRIC_DATA_FORMAT *OutputFormat,
    BioAPI_BIR_HANDLE *CapturedBIR,
    int32_t Timeout,
    BioAPI_BIR_HANDLE *AuditData);
```

and the following parameters:

Parameter	Function invocation (conformance testing model for frameworks)	Bound activity invocation (conformance testing model for applications)
`BSPHandle`	input, outbound	input, inbound
`Purpose`	input, outbound	input, inbound
`Subtype`	input, outbound	input, inbound
`OutputFormatOwner`	input, outbound	input, inbound
`OutputFormatType`	input, outbound	input, inbound
`no_CapturedBIR`	input, outbound	input, inbound

`Timeout`	input, outbound	input, inbound
`no_AuditData`	input, outbound	input, inbound
`CapturedBIR`	output, inbound	input, inbound
`AuditData`	output, inbound	input, inbound
`return`	return, inbound	input, inbound

9.20.2 Constraints on the Parameters

9.20.2.1 An outbound value of `BSPHandle` shall be a valid representation of an integer (see 7.4) in the range 0 to 4294967295. An inbound value shall be the canonical representation of an integer (see 7.4.3) in the same range.

9.20.2.2 An outbound value of `Purpose` and `Subtype` shall be a valid representation of an integer in the range 0 to 255. An inbound value shall be the canonical representation of an integer in the same range.

9.20.2.3 An outbound value of `OutputFormatOwner` and `OutputFormatType` shall be a valid representation of an integer in the range 0 to 65535. An inbound value shall be the canonical representation of an integer in the same range.

9.20.2.4 An outbound value of `no_CapturedBIR` and `no_AuditData` shall be either a valid representation of a boolean (see 7.5) or an empty string. An inbound value shall be the canonical representation of a boolean (see 7.5.2).

9.20.2.5 An outbound value of `Timeout` shall be a valid representation of an integer in the range -2147483648 to 2147483647. An inbound value shall be the canonical representation of an integer in the same range.

9.20.2.6 An outbound value of `CapturedBIR` and `AuditData` shall be a valid representation of an integer in the range -2147483648 to 2147483647, or an empty string. An inbound value shall be the canonical representation of an integer in the same range, or an empty string.

9.20.2.7 An outbound value of `return` shall be a valid representation of an integer in the range 0 to 4294967295. An inbound value shall be the canonical representation of an integer in the same range.

9.20.3 Function Invocation Input

9.20.3.1 The integer represented by the outbound value of `BSPHandle`, `Purpose`, `Subtype`, and `Timeout` shall be assigned to the native parameter with the same name.

9.20.3.2 The integer represented by the outbound value of `OutputFormatOwner` and `OutputFormatType` shall be written to the fields `FormatOwner` and `FormatType` (respectively) of a variable of type `BioAPI_BIR_BIOMETRIC_DATA_FORMAT`, whose address shall be assigned to the native parameter `OutputFormat`.

9.20.3.3 If the outbound value of `no_CapturedBIR` is "`true`", then the native parameter `CapturedBIR` shall be set to NULL, otherwise it shall be set to the address of a variable of type `BioAPI_BIR_HANDLE`.

9.20.3.4 If the outbound value of `no_AuditData` is "`true`", then the native parameter `AuditData` shall be set to NULL, otherwise it shall be set to the address of a variable of type `BioAPI_BIR_HANDLE`.

9.20.4 Function Invocation Output

9.20.4.1 The inbound value of `CapturedBIR` shall be determined as follows. If the native parameter with the same name is NULL, then the inbound value shall be an empty string. Otherwise, the inbound value shall be the canonical representation of the integer in the variable of type `BioAPI_BIR_HANDLE` pointed to by the native parameter `CapturedBIR`.

9.20.4.2 The inbound value of `AuditData` shall be determined as follows. If the native parameter with the same name is NULL, then the inbound value shall be an empty string. Otherwise, the inbound value shall be the canonical representation of the integer in the variable of type `BioAPI_BIR_HANDLE` pointed to by the native parameter `AuditData`.

9.20.5 Bound Activity Invocation Input

9.20.5.1 The inbound value of `BSPHandle`, `Purpose`, `Subtype`, and `Timeout` shall be the canonical representation of the integer in the native parameter with the same name.

9.20.5.2 The inbound value of `OutputFormatOwner` and `OutputFormatType` shall be the canonical representation of the integer in the fields `FormatOwner` and `FormatType` (respectively) of the variable of type `BioAPI_BIR_BIOMETRIC_DATA_FORMAT` pointed to by the native parameter `OutputFormat`.

9.20.5.3 The inbound value of `no_CapturedBIR` shall be "`true`" if the native parameter `CapturedBIR` is NULL, otherwise it shall be "`false`".

9.20.5.4 The inbound value of `no_AuditData` shall be "`true`" if the native parameter `AuditData` is NULL, otherwise it shall be "`false`".

9.20.5.5 The inbound value of `return` shall be the canonical representation of the integer returned by the native function.

9.20.5.6 The inbound values of the other input parameters shall be determined as specified in 9.20.4.

9.20.6 Bound Activity Invocation Output

There are no output parameters.

9.21 BioAPI_CreateTemplate

9.21.1 Function Invocation Scheme

This function belongs to the BioAPI interface, and has the following native prototype:

```
BioAPI_RETURN BioAPI
BioAPI_CreateTemplate(
    BioAPI_HANDLE BSPHandle,
    const BioAPI_INPUT_BIR *CapturedBIR,
    const BioAPI_INPUT_BIR *ReferenceTemplate,
    const BioAPI_BIR_BIOMETRIC_DATA_FORMAT *OutputFormat,
    BioAPI_BIR_HANDLE *NewTemplate,
```

const BioAPI_DATA *Payload,

BioAPI_UUID *TemplateUUID);

and the following parameters:

Parameter	**Function invocation (conformance testing model for frameworks)**	**Bound activity invocation (conformance testing model for applications)**
`BSPHandle`	input, outbound	input, inbound
the members of the parameter group "Input BIR" (see 9.2.15) with their names prefixed by "`CapturedBIR_`"	input, outbound	input, inbound
the members of the parameter group "Input BIR" with their names prefixed by "`ReferenceTemplate_`"	input, outbound	input, inbound
`OutputFormatOwner`	input, outbound	input, inbound
`OutputFormatType`	input, outbound	input, inbound
`no_NewTemplate`	input, outbound	input, inbound
`Payload`	input, outbound	input, inbound
`no_TemplateUUID`	input, outbound	input, inbound
`NewTemplate`	output, inbound	input, inbound
`TemplateUUID`	output, inbound	input, inbound
`Return`	return, inbound	input, inbound

9.21.2 Constraints on the Parameters

9.21.2.1 An outbound value of `BSPHandle` shall be a valid representation of an integer (see 7.4) in the range 0 to 4294967295. An inbound value shall be the canonical representation of an integer (see 7.4.3) in the same range.

9.21.2.2 An outbound value of each parameter group "Input BIR" shall be a valid representation of a value of the native type `BioAPI_INPUT_BIR`. An inbound value shall be the canonical representation of a value of that type.

9.21.2.3 An outbound value of `OutputFormatOwner` and `OutputFormatType` shall be a valid representation of an integer in the range 0 to 65535. An inbound value shall be the canonical representation of an integer in the same range.

9.21.2.4 An outbound value of `no_NewTemplate` and `no_TemplateUUID` shall be either a valid representation of a boolean (see 7.5) or an empty string. An inbound value shall be the canonical representation of a boolean (see 7.5.2).

9.21.2.5 An outbound value of `Payload` shall be a valid representation of an octet string (see 7.7). An inbound value shall be the canonical representation of an octet string (see 7.7.2).

9.21.2.6 An outbound value of `NewTemplate` shall be either a valid representation of an integer in the range -2147483648 to 2147483647, or an empty string. An inbound value shall be either the canonical representation of an integer in the same range, or an empty string.

9.21.2.7 An outbound value of `TemplateUUID` shall be either a valid representation of a UUID (see 7.6), or an empty string. An inbound value shall be either the canonical representation of a UUID (see 7.6.3), or an empty string.

9.21.2.8 An outbound value of `return` shall be a valid representation of an integer in the range 0 to 4294967295. An inbound value shall be the canonical representation of an integer in the same range.

9.21.3 Function Invocation Input

9.21.3.1 The integer represented by the outbound value of `BSPHandle` shall be assigned to the native parameter with the same name.

9.21.3.2 The value of type `BioAPI_INPUT_BIR` represented (see 9.2.15.4) by the outbound value of the parameter group "Input BIR" with the prefix "`CapturedBIR_`" shall be written to a variable of type `BioAPI_INPUT_BIR`, whose address shall be assigned to the native parameter `CapturedBIR`.

9.21.3.3 The value of type `BioAPI_INPUT_BIR` represented (see 9.2.15.4) by the outbound value of the parameter group "Input BIR" with the prefix "`ReferenceTemplate_`" shall be written to a variable of type `BioAPI_INPUT_BIR`, whose address shall be assigned to the native parameter `ReferenceTemplate`.

9.21.3.4 The integer represented by the outbound value of `OutputFormatOwner` and `OutputFormatType` shall be written to the fields `FormatOwner` and `FormatType` (respectively) of a variable of type `BioAPI_BIR_BIOMETRIC_DATA_FORMAT`, whose address shall be assigned to the native parameter `OutputFormat`.

9.21.3.5 If the outbound value of `no_NewTemplate` is "`true`", then the native parameter `NewTemplate` shall be set to NULL. Otherwise, it shall be set to the address of a variable of type `BioAPI_BIR_HANDLE`.

9.21.3.6 If the outbound value of `Payload` is an empty string, then the native parameter `Payload` shall be set to NULL. Otherwise, the octet string represented by the outbound value shall be written to a memory block of sufficient size. The address and length of that memory block shall be written to the fields `Data` and `Length` (respectively) of a variable of type `BioAPI_DATA`, whose address shall be assigned to the native parameter `Payload`.

9.21.3.7 If the outbound value of `no_TemplateUUID` is "`true`", then the native parameter `TemplateUUID` shall be set to NULL, otherwise it shall be set to the address of a variable of type `BioAPI_UUID`.

9.21.4 Function Invocation Output

9.21.4.1 The inbound value of **`NewTemplate`** shall be determined as follows. If the native parameter with the same name is NULL, then the inbound value shall be an empty string. Otherwise, the inbound value shall be the canonical representation of the integer in the variable of type **`BioAPI_BIR_HANDLE`** pointed to by the native parameter **`NewTemplate`**.

9.21.4.2 The inbound value of **`TemplateUUID`** shall be determined as follows. If the native parameter with the same name is NULL, then the inbound value shall be an empty string. Otherwise, the inbound value shall be the canonical representation of the UUID in the variable of type **`BioAPI_UUID`** pointed to by the native parameter **`TemplateUUID`**.

9.21.5 Bound Activity Invocation Input

9.21.5.1 The inbound value of **`BSPHandle`** shall be the canonical representation of the integer in the native parameter with the same name.

9.21.5.2 The inbound value of the parameter group "Input BIR" with the prefix "**`CapturedBIR_`**" shall be the canonical representation (see 9.2.15.5) of the value of type **`BioAPI_INPUT_BIR`** in the variable pointed to by the native parameter **`CapturedBIR`**.

9.21.5.3 The inbound value of the parameter group "Input BIR" with the prefix "**`ReferenceTemplate_`**" shall be the canonical representation (see 9.2.15.5) of the value of type **`BioAPI_INPUT_BIR`** in the variable pointed to by the native parameter **`ReferenceTemplate`**.

9.21.5.4 The inbound value of **`OutputFormatOwner`** and **`OutputFormatType`** shall be the canonical representation of the integer in the fields **`FormatOwner`** and **`FormatType`** (respectively) of the variable of type **`BioAPI_BIR_BIOMETRIC_DATA_FORMAT`** pointed to by the native parameter **`OutputFormat`**.

9.21.5.5 The inbound value of **`no_NewTemplate`** shall be "**`true`**" if the native parameter **`NewTemplate`** is NULL, otherwise it shall be "**`false`**".

9.21.5.6 The inbound value of **`Payload`** shall be determined as follows. If the native parameter with the same name is NULL, then the inbound value shall be an empty string. Otherwise, the inbound value shall be the canonical representation (see 7.7.2) of the octet string in the memory block whose address and length are in the fields **`Data`** and **`Length`** (respectively) of the variable of type **`BioAPI_DATA`** pointed to by the native parameter **`Payload`**.

9.21.5.7 The inbound value of **`no_TemplateUUID`** shall be "**`true`**" if the native parameter **`TemplateUUID`** is NULL, otherwise it shall be "**`false`**".

9.21.5.8 The inbound value of **`return`** shall be the canonical representation of the integer returned by the native function.

9.21.5.9 The inbound values of the other input parameters shall be determined as specified in 9.21.4.

9.21.6 Bound Activity Invocation Output

There are no output parameters.

9.22 BioAPI_Process

9.22.1 Function Invocation Scheme

This function belongs to the BioAPI interface, and has the following native prototype:

BioAPI_RETURN BioAPI

BioAPI_Process(

BioAPI_HANDLE BSPHandle,

const BioAPI_INPUT_BIR *CapturedBIR,

const BioAPI_BIR_BIOMETRIC_DATA_FORMAT *OutputFormat,

BioAPI_BIR_HANDLE *ProcessedBIR);

and the following parameters:

Parameter	**Function invocation (conformance testing model for frameworks)**	**Bound activity invocation (conformance testing model for applications)**
`BSPHandle`	input, outbound	input, inbound
the members of the parameter group "Input BIR" (see 9.2.15) with their names prefixed by "`CapturedBIR_`"	input, outbound	input, inbound
`OutputFormatOwner`	input, outbound	input, inbound
`OutputFormatType`	input, outbound	input, inbound
`no_ProcessedBIR`	input, outbound	input, inbound
`ProcessedBIR`	output, inbound	input, inbound
`return`	return, inbound	input, inbound

9.22.2 Constraints on the Parameters

9.22.2.1 An outbound value of `BSPHandle` shall be a valid representation of an integer (see 7.4) in the range 0 to 4294967295. An inbound value shall be the canonical representation of an integer (see 7.4.3) in the same range.

9.22.2.2 An outbound value of the parameter group "Input BIR" shall be a valid representation of a value of the native type `BioAPI_INPUT_BIR`. An inbound value shall be the canonical representation of a value of that type.

9.22.2.3 An outbound value of **`OutputFormatOwner`** and **`OutputFormatType`** shall be a valid representation of an integer in the range 0 to 65535. An inbound value shall be the canonical representation of an integer in the same range.

9.22.2.4 An outbound value of **`no_ProcessedBIR`** shall be either a valid representation of a boolean (see 7.5) or an empty string. An inbound value shall be the canonical representation of a boolean (see 7.5.2).

9.22.2.5 An outbound value of **`ProcessedBIR`** shall be either a valid representation of an integer in the range -2147483648 to 2147483647, or an empty string. An inbound value shall be either the canonical representation of an integer in the same range, or an empty string.

9.22.2.6 An outbound value of **`return`** shall be a valid representation of an integer in the range 0 to 4294967295. An inbound value shall be the canonical representation of an integer in the same range.

9.22.3 Function Invocation Input

9.22.3.1 The integer represented by the outbound value of **`BSPHandle`** shall be assigned to the native parameter with the same name.

9.22.3.2 The value of type **`BioAPI_INPUT_BIR`** represented (see 9.2.15.4) by the outbound value of the parameter group "Input BIR" shall be written to a variable of type **`BioAPI_INPUT_BIR`**, whose address shall be assigned to the native parameter **`CapturedBIR`**.

9.22.3.3 The integer represented by the outbound value of **`OutputFormatOwner`** and **`OutputFormatType`** shall be written to the fields **`FormatOwner`** and **`FormatType`** (respectively) of a variable of type **`BioAPI_BIR_BIOMETRIC_DATA_FORMAT`**, whose address shall be assigned to the native parameter **`OutputFormat`**.

9.22.3.4 If the outbound value of **`no_ProcessedBIR`** is "**`true`**", then the native parameter **`ProcessedBIR`** shall be set to NULL, otherwise it shall be set to the address of a variable of type **`BioAPI_BIR_HANDLE`**.

9.22.4 Function Invocation Output

The inbound value of **`ProcessedBIR`** shall be determined as follows. If the native parameter with the same name is NULL, then the inbound value shall be an empty string. Otherwise, the inbound value shall be the canonical representation of the integer in the variable of type **`BioAPI_BIR_HANDLE`** pointed to by the native parameter **`ProcessedBIR`**.

9.22.5 Bound Activity Invocation Input

9.22.5.1 The inbound value of **`BSPHandle`** shall be the canonical representation of the integer in the native parameter with the same name.

9.22.5.2 The inbound value of the parameter group "Input BIR" shall be the canonical representation (see 9.2.15.5) of the value of type **`BioAPI_INPUT_BIR`** in the variable pointed to by the native parameter **`CapturedBIR`**.

9.22.5.3 The inbound value of **`OutputFormatOwner`** and **`OutputFormatType`** shall be the canonical representation of the integer in the fields **`FormatOwner`** and **`FormatType`** (respectively) of the variable of type **`BioAPI_BIR_BIOMETRIC_DATA_FORMAT`** pointed to by the native parameter **`OutputFormat`**.

9.22.5.4 The inbound value of **`no_ProcessedBIR`** shall be "**`true`**" if the native parameter **`ProcessedBIR`** is NULL, otherwise it shall be "**`false`**".

9.22.5.5 The inbound value of **`return`** shall be the canonical representation of the integer returned by the native function.

9.22.5.6 The inbound values of the other input parameters shall be determined as specified in 9.22.4.

9.22.6 Bound Activity Invocation Output

There are no output parameters.

9.23 BioAPI_ProcessWithAuxBIR

9.23.1 Function Invocation Scheme

This function belongs to the BioAPI interface, and has the following native prototype:

BioAPI_RETURN BioAPI

BioAPI_ProcessWithAuxBIR(

BioAPI_HANDLE BSPHandle,

const BioAPI_INPUT_BIR *CapturedBIR,

const BioAPI_INPUT_BIR *AuxiliaryData,

const BioAPI_BIR_BIOMETRIC_DATA_FORMAT *OutputFormat,

BioAPI_HANDLE *ProcessedBIR);

and the following parameters:

Parameter	**Function invocation (conformance testing model for frameworks)**	**Bound activity invocation (conformance testing model for applications)**
`BSPHandle`	input, outbound	input, inbound
the members of the parameter group "Input BIR" (see 9.2.15) with their names prefixed by "`CapturedBIR_`"	input, outbound	input, inbound
the members of the parameter group "Input BIR" with their names prefixed by "`AuxiliaryData_`"	input, outbound	input, inbound
`OutputFormatOwner`	input, outbound	input, inbound
`OutputFormatType`	input, outbound	input, inbound
`no_ProcessedBIR`	input, outbound	input, inbound
`ProcessedBIR`	output, inbound	input, inbound
`return`	return, inbound	input, inbound

9.23.2 Constraints on the Parameters

9.23.2.1 An outbound value of **BSPHandle** shall be a valid representation of an integer (see 7.4) in the range 0 to 4294967295. An inbound value shall be the canonical representation of an integer (see 7.4.3) in the same range.

9.23.2.2 An outbound value of each parameter group "Input BIR" shall be a valid representation of a value of the native type **BioAPI_INPUT_BIR**. An inbound value shall be the canonical representation of a value of that type.

9.23.2.3 An outbound value of **OutputFormatOwner** and **OutputFormatType** shall be a valid representation of an integer in the range 0 to 65535. An inbound value shall be the canonical representation of an integer in the same range.

9.23.2.4 An outbound value of **no_ProcessedBIR** shall be either a valid representation of a boolean (see 7.5) or an empty string. An inbound value shall be the canonical representation of a boolean (see 7.5.2).

9.23.2.5 An outbound value of **ProcessedBIR** shall be either a valid representation of an integer in the range -2147483648 to 2147483647, or an empty string. An inbound value shall be either the canonical representation of an integer in the same range, or an empty string.

9.23.2.6 An outbound value of **return** shall be a valid representation of an integer in the range 0 to 4294967295. An inbound value shall be the canonical representation of an integer in the same range.

9.23.3 Function Invocation Input

9.23.3.1 The integer represented by the outbound value of **BSPHandle** shall be assigned to the native parameter with the same name.

9.23.3.2 The value of type **BioAPI_INPUT_BIR** represented (see 9.2.15.4) by the outbound value of the parameter group "Input BIR" with the prefix "**CapturedBIR_**" shall be written to a variable of type **BioAPI_INPUT_BIR**, whose address shall be assigned to the native parameter **CapturedBIR**.

9.23.3.3 The value of type **BioAPI_INPUT_BIR** represented (see 9.2.15.4) by the outbound value of the parameter group "Input BIR" with the prefix "**AuxiliaryData_**" shall be written to a variable of type **BioAPI_INPUT_BIR**, whose address shall be assigned to the native parameter **AuxiliaryData**.

9.23.3.4 The integer represented by the outbound value of **OutputFormatOwner** and **OutputFormatType** shall be written to the fields **FormatOwner** and **FormatType** (respectively) of a variable of type **BioAPI_BIR_BIOMETRIC_DATA_FORMAT**, whose address shall be assigned to the native parameter **OutputFormat**.

9.23.3.5 If the outbound value of **no_ProcessedBIR** is "**true**", then the native parameter **ProcessedBIR** shall be set to NULL, otherwise it shall be set to the address of a variable of type **BioAPI_BIR_HANDLE**.

9.23.4 Function Invocation Output

The inbound value of **ProcessedBIR** shall be determined as follows. If the native parameter with the same name is NULL, then the inbound value shall be an empty string. Otherwise, the inbound value shall be the canonical representation of the integer in the variable of type **BioAPI_BIR_HANDLE** pointed to by the native parameter **ProcessedBIR**.

9.23.5 Bound Activity Invocation Input

9.23.5.1 The inbound value of `BSPHandle` shall be the canonical representation of the integer in the native parameter with the same name.

9.23.5.2 The inbound value of the parameter group "Input BIR" with the prefix "`CapturedBIR_`" shall be the canonical representation (see 9.2.15.5) of the value of type `BioAPI_INPUT_BIR` in the variable pointed to by the native parameter `CapturedBIR`.

9.23.5.3 The inbound value of the parameter group "Input BIR" with the prefix "`AuxiliaryData_`" shall be the canonical representation (see 9.2.15.5) of the value of type `BioAPI_INPUT_BIR` in the variable pointed to by the native parameter `AuxiliaryData`.

9.23.5.4 The inbound value of `OutputFormatOwner` and `OutputFormatType` shall be the canonical representation of the integer in the fields `FormatOwner` and `FormatType` (respectively) of the variable of type `BioAPI_BIR_BIOMETRIC_DATA_FORMAT` pointed to by the native parameter `OutputFormat`.

9.23.5.5 The inbound value of `no_ProcessedBIR` shall be "`true`" if the native parameter `ProcessedBIR` is NULL, otherwise it shall be "`false`".

9.23.5.6 The inbound value of `return` shall be the canonical representation of the integer returned by the native function.

9.23.5.7 The inbound values of the other input parameters shall be determined as specified in 9.23.4.

9.23.6 Bound Activity Invocation Output

There are no output parameters.

9.24 BioAPI_VerifyMatch

9.24.1 Function Invocation Scheme

This function belongs to the BioAPI interface, and has the following native prototype:

```
BioAPI_RETURN BioAPI
BioAPI_VerifyMatch(
    BioAPI_HANDLE BSPHandle,
    BioAPI_FMR MaxFMRRequested,
    const BioAPI_INPUT_BIR *ProcessedBIR,
    const BioAPI_INPUT_BIR *ReferenceTemplate,
    BioAPI_BIR_HANDLE *AdaptedBIR,
    BioAPI_BOOL *Result,
    BioAPI_FMR *FMRAchieved,
    BioAPI_DATA *Payload);
```

and the following parameters:

Parameter	Function invocation (conformance testing model for frameworks)	Bound activity invocation (conformance testing model for applications)
`BSPHandle`	input, outbound	input, inbound
`MaxFMRRequested`	input, outbound	input, inbound
the members of the parameter group "Input BIR" (see 9.2.15) with their names prefixed by "`ProcessedBIR_`"	input, outbound	input, inbound
the members of the parameter group "Input BIR" with their names prefixed by "`ReferenceTemplate_`"	input, outbound	input, inbound
`no_AdaptedBIR`	input, outbound	input, inbound
`no_Result`	input, outbound	input, inbound
`no_FMRAchieved`	input, outbound	input, inbound
`no_Payload`	input, outbound	input, inbound
`AdaptedBIR`	output, inbound	input, inbound
`Result`	output, inbound	input, inbound
`FMRAchieved`	output, inbound	input, inbound
`Payload`	output, inbound	input, inbound
`Return`	return, inbound	input, inbound

9.24.2 Constraints on the Parameters

9.24.2.1 An outbound value of `BSPHandle` shall be a valid representation of an integer (see 7.4) in the range 0 to 4294967295. An inbound value shall be the canonical representation of an integer (see 7.4.3) in the same range.

9.24.2.2 An outbound value of **`MaxFMRRequested`** shall be a valid representation of an integer in the range -2147483648 to 2147483647. An inbound value shall be the canonical representation of an integer in the same range.

9.24.2.3 An outbound value of each parameter group "Input BIR" shall be a valid representation of a value of the native type **`BioAPI_INPUT_BIR`**. An inbound value shall be the canonical representation of a value of that type.

9.24.2.4 either a valid representation of a boolean (see 7.5) or an empty string. An inbound value shall be the canonical representation of a boolean (see 7.5.2).

9.24.2.5 An outbound value of **`AdaptedBIR`** and **`FMRAchieved`** shall be either a valid representation of an integer in the range -2147483648 to 2147483647, or an empty string. An inbound value shall be the canonical representation of an integer in the same range, or an empty string.

9.24.2.6 An outbound value of **`Result`** shall be either a valid representation of a boolean, or an empty string. An inbound value shall be the canonical representation of a boolean, or an empty string.

9.24.2.7 An outbound value of **`Payload`** shall be a valid representation of an octet string. An inbound value shall be the canonical representation of an octet string.

9.24.2.8 An outbound value of **`return`** shall be a valid representation of an integer in the range 0 to 4294967295. An inbound value shall be the canonical representation of an integer in the same range.

9.24.3 Function Invocation Input

9.24.3.1 The integer represented by the outbound value of **`BSPHandle`** and **`MaxFMRRequested`** shall be assigned to the native parameter with the same name.

9.24.3.2 The value of type **`BioAPI_INPUT_BIR`** represented (see 9.2.15.4) by the outbound value of the parameter group "Input BIR" with the prefix "**`ProcessedBIR_`**" shall be written to a variable of type **`BioAPI_INPUT_BIR`**, whose address shall be assigned to the native parameter **`ProcessedBIR`**.

9.24.3.3 The value of type **`BioAPI_INPUT_BIR`** represented (see 9.2.15.4) by the outbound value of the parameter group "Input BIR" with the prefix "**`ReferenceTemplate_`**" shall be written to a variable of type **`BioAPI_INPUT_BIR`**, whose address shall be assigned to the native parameter **`ReferenceTemplate`**.

9.24.3.4 If the outbound value of **`no_AdaptedBIR`** is "**`true`**", then the native parameter **`AdaptedBIR`** shall be set to NULL, otherwise it shall be set to the address of a variable of type **`BioAPI_BIR_HANDLE`**.

9.24.3.5 If the outbound value of **`no_Result`** is "**`true`**", then the native parameter **`Result`** shall be set to NULL, otherwise it shall be set to the address of a variable of type **`BioAPI_BOOL`**.

9.24.3.6 If the outbound value of **`no_FMRAchieved`** is "**`true`**", then the native parameter **`FMRAchieved`** shall be set to NULL, otherwise it shall be set to the address of a variable of type **`BioAPI_FMR`**.

9.24.3.7 If the outbound value of **`no_Payload`** is "**`true`**", then the native parameter **`Payload`** shall be set to NULL, otherwise it shall be set to the address of a variable of type **`BioAPI_DATA`**.

9.24.4 Function Invocation Output

9.24.4.1 The inbound value of **`AdaptedBIR`** shall be determined as follows. If the native parameter with the same name is NULL, then the inbound value shall be an empty string. Otherwise, the inbound value shall be the canonical representation of the integer in the variable of type **`BioAPI_BIR_HANDLE`** pointed to by the native parameter **`AdaptedBIR`**.

9.24.4.2 The inbound value of `FMRAchieved` shall be determined as follows. If the native parameter with the same name is NULL, then the inbound value shall be an empty string. Otherwise, the inbound value shall be the canonical representation of the integer in variable of type `BioAPI_FMR` pointed to by the native parameter `FMRAchieved`.

9.24.4.3 The inbound value of `Result` shall be determined as follows. If the native parameter with the same name is NULL, then the inbound value shall be an empty string. Otherwise, the inbound value shall be the canonical representation of the boolean in the variable of type `BioAPI_BOOL` pointed to by the native parameter `Result`.

9.24.4.4 The inbound value of `Payload` shall be determined as follows. If the native parameter with the same name is NULL, then the inbound value shall be an empty string. Otherwise, the inbound value shall be the canonical representation of the octet string in the memory block whose address and length are in the fields `Data` and `Length` (respectively) of the variable of type `BioAPI_DATA` pointed to by the native parameter `Payload`.

9.24.5 Bound Activity Invocation Input

9.24.5.1 The inbound value of `BSPHandle` and `MaxFMRRequested` shall be the canonical representation of the integer in the native parameter with the same name.

9.24.5.2 The inbound value of the parameter group "Input BIR" with the prefix "`ProcessedBIR_`" shall be the canonical representation (see 9.2.15.5) of the value of type `BioAPI_INPUT_BIR` in the variable pointed to by the native parameter `ProcessedBIR`.

9.24.5.3 The inbound value of the parameter group "Input BIR" with the prefix "`ReferenceTemplate_`" shall be the canonical representation (see 9.2.15.5) of the value of type `BioAPI_INPUT_BIR` in the variable pointed to by the native parameter `ReferenceTemplate`.

9.24.5.4 The inbound value of `no_AdaptedBIR` shall be "`true`" if the native parameter `AdaptedBIR` is NULL, otherwise it shall be "`false`".

9.24.5.5 The inbound value of `no_Result` shall be "`true`" if the native parameter `Result` is NULL, otherwise it shall be "`false`".

9.24.5.6 The inbound value of `no_FMRAchieved` shall be "`true`" if the native parameter `FMRAchieved` is NULL, otherwise it shall be "`false`".

9.24.5.7 The inbound value of `no_Payload` shall be "`true`" if the native parameter `Payload` is NULL, otherwise it shall be "`false`".

9.24.5.8 The inbound value of `return` shall be the canonical representation of the integer returned by the native function.

9.24.5.9 The inbound values of the other input parameters shall be determined as specified in 9.24.4.

9.24.6 Bound Activity Invocation Output

There are no output parameters.

9.25 BioAPI_IdentifyMatch

9.25.1 Function Invocation Scheme

This function belongs to the BioAPI interface, and has the following native prototype:

```
BioAPI_RETURN BioAPI
BioAPI_IdentifyMatch(
    BioAPI_HANDLE BSPHandle,
    BioAPI_FMR MaxFMRRequested,
    const BioAPI_INPUT_BIR *ProcessedBIR,
    const BioAPI_IDENTIFY_POPULATION *Population,
    uint32_t TotalNumberOfTemplates,
    BioAPI_BOOL Binning,
    uint32_t MaxNumberOfResults,
    uint32_t *NumberOfResults,
    BioAPI_CANDIDATE **Candidates,
    int32_t Timeout);
```

and the following parameters:

Parameter	**Function invocation (conformance testing model for frameworks)**	**Bound activity invocation (conformance testing model for applications)**
`BSPHandle`	input, outbound	input, inbound
`MaxFMRRequested`	input, outbound	input, inbound
the members of the parameter group "Input BIR" (see 9.2.15) with their names prefixed by "`ProcessedBIR_`"	input, outbound	input, inbound
the members of the parameter group "Identify population" (see 9.2.16)	input, outbound	input, inbound
`TotalNumberOfTemplates`	input, outbound	input, inbound
`Binning`	input, outbound	input, inbound
`MaxNumberOfResults`	input, outbound	input, inbound

no_NumberOfResults	input, outbound	input, inbound
no_Candidates	input, outbound	input, inbound
Timeout	input, outbound	input, inbound
NumberOfResults	output, inbound	input, inbound
the members of the parameter group "Candidate" (see 9.2.17) with their names prefixed by "**Candidate_1_**"	output, inbound	input, inbound
the members of the parameter group "Candidate" with their names prefixed by "**Candidate_2_**"	output, inbound	input, inbound
the members of the parameter group "Candidate" with their names prefixed by "**Candidate_3_**"	output, inbound	input, inbound
the members of the parameter group "Candidate" with their names prefixed by "**Candidate_4_**"	output, inbound	input, inbound
Return	return, inbound	input, inbound

9.25.2 Constraints on the Parameters

9.25.2.1 An outbound value of **BSPHandle**, **TotalNumberOfTemplates**, and **MaxNumberOfResults** shall be a valid representation of an integer (see 7.4) in the range 0 to 4294967295. An inbound value shall be the canonical representation of an integer (see 7.4.3) in the same range.

9.25.2.2 An outbound value of **MaxFMRRequested** shall be a valid representation of an integer in the range -2147483648 to 2147483647. An inbound value shall be the canonical representation of an integer in the same range.

9.25.2.3 An outbound value of the parameter group "Input BIR" shall be a valid representation of a value of the native type **BioAPI_INPUT_BIR**. An inbound value shall be the canonical representation of a value of that type.

9.25.2.4 An outbound value of the parameter group "Identify population" shall be a valid representation of a value of the native type **BioAPI_IDENTIFY_POPULATION**. An inbound value shall be the canonical representation of a value of that type.

9.25.2.5 An outbound value of **Binning** shall be a valid representation of a boolean. An inbound value shall be the canonical representation of a boolean.

9.25.2.6 An outbound value of **no_NumberOfResults** and **no_Candidates** shall be either a valid representation of a boolean or an empty string. An inbound value shall be the canonical representation of a boolean.

9.25.2.7 An outbound value of **Timeout** shall be a valid representation of an integer in the range -2147483648 to 2147483647. An inbound value shall be the canonical representation of an integer in the same range.

9.25.2.8 An outbound value of **NumberOfResults** shall be either a valid representation of an integer in the range 0 to 4294967295, or an empty string. An inbound value shall be either the canonical representation of an integer in the same range, or an empty string.

9.25.2.9 An outbound value of each parameter group "Candidate" shall be a valid representation of a value of the native type **BioAPI_CANDIDATE**. An inbound value shall be either the canonical representation of a value of that type, or a set of empty strings.

9.25.2.10 An outbound value of **return** shall be a valid representation of an integer in the range 0 to 4294967295. An inbound value shall be the canonical representation of an integer in the same range.

9.25.3 Function Invocation Input

9.25.3.1 The integer represented by the outbound value of **BSPHandle**, **MaxFMRRequested**, **TotalNumberOfTemplates**, **MaxNumberOfResults**, and **Timeout** shall be assigned to the native parameter with the same name.

9.25.3.2 The boolean represented by the outbound value of **Binning** shall be assigned to the native parameter with the same name.

9.25.3.3 The value of type **BioAPI_INPUT_BIR** represented (see 9.2.15.4) by the outbound value of the parameter group "Input BIR" shall be written to a variable of type **BioAPI_INPUT_BIR**, whose address shall be assigned to the native parameter **ProcessedBIR**.

9.25.3.4 The value of type **BioAPI_IDENTIFY_POPULATION** represented (see 9.2.16.4) by the outbound value of the parameter group "Identify population" shall be written to a variable of type **BioAPI_IDENTIFY_POPULATION**, whose address shall be assigned to the native parameter **Population**.

9.25.3.5 If the outbound value of **no_NumberOfResults** is "**true**", then the native parameter **NumberOfResults** shall be set to NULL, otherwise it shall be set to the address of a variable of type **uint32_t**.

9.25.3.6 If the outbound value of **no_Candidates** is "**true**", then the native parameter **Candidates** shall be set to NULL, otherwise it shall be set to the address of a variable of type **BioAPI_CANDIDATE*** (a pointer).

9.25.4 Function Invocation Output

9.25.4.1 The inbound value of **NumberOfResults** shall be determined as follows. If the native parameter with the same name is NULL, then the inbound value shall be an empty string. Otherwise, the inbound value shall be the canonical representation of the integer in the variable of type **uint32_t** pointed to by the native parameter **NumberOfResults**.

9.25.4.2 The inbound value of the parameter group "Candidate" with the prefix "**Candidate_*X*_**" (with *X*=1, 2, 3, or 4) shall be determined as follows. If the native parameter **Candidates** is NULL or the variable pointed to by that native parameter is NULL or the native parameter **NumberOfResults** is NULL or the value of the integer variable pointed to that native parameter is less than *X*, then the inbound value shall be a set of empty strings. Otherwise, the inbound value shall be the canonical representation (see 9.2.14.5) of the value of type `BioAPI_CANDIDATE` at position *X* in the array pointed to by the variable pointed to by the native parameter `Candidates`.

9.25.5 Bound Activity Invocation Input

9.25.5.1 The inbound value of `BSPHandle`, `MaxFMRRequested`, `TotalNumberOfTemplates`, `MaxNumberOfResults`, and `Timeout` shall be the canonical representation of the integer in the native parameter with the same name.

9.25.5.2 The inbound value of `Binning` shall be the canonical representation of the boolean in the native parameter with the same name.

9.25.5.3 The inbound value of the parameter group "Input BIR" shall be the canonical representation (see 9.2.15.5) of the value of type `BioAPI_INPUT_BIR` in the variable pointed to by the native parameter `ProcessedBIR`.

9.25.5.4 The inbound value of the parameter group "Identify population" shall be the canonical representation (see 9.2.16.5) of the value of type `BioAPI_IDENTIFY_POPULATION` in the variable pointed to by the native parameter `Population`.

9.25.5.5 The inbound value of `no_NumberOfResults` shall be "`true`" if the native parameter `NumberOfResults` is NULL, otherwise it shall be "`false`".

9.25.5.6 The inbound value of `no_Candidates` shall be "`true`" if the native parameter `Candidates` is NULL, otherwise it shall be "`false`".

9.25.5.7 The inbound value of `return` shall be the canonical representation of the integer returned by the native function.

9.25.5.8 The inbound values of the other input parameters shall be determined as specified in 9.25.4.

9.25.6 Bound Activity Invocation Output

There are no output parameters.

9.26 BioAPI_Enroll

9.26.1 Function Invocation Scheme

This function belongs to the BioAPI interface, and has the following native prototype:

```
BioAPI_RETURN BioAPI
BioAPI_Enroll(
    BioAPI_HANDLE BSPHandle,
    BioAPI_BIR_PURPOSE Purpose,
    BioAPI_BIR_SUBTYPE Subtype,
    const BioAPI_BIR_BIOMETRIC_DATA_FORMAT *OutputFormat,
    const BioAPI_INPUT_BIR *ReferenceTemplate,
    BioAPI_BIR_HANDLE *NewTemplate,
    const BioAPI_DATA *Payload,
```

int32_t Timeout,

BioAPI_BIR_HANDLE *AuditData,

BioAPI_UUID *TemplateUUID);

and the following parameters:

Parameter	**Function invocation (conformance testing model for frameworks)**	**Bound activity invocation (conformance testing model for applications)**
`BSPHandle`	input, outbound	input, inbound
`Purpose`	input, outbound	input, inbound
`Subtype`	input, outbound	input, inbound
`OutputFormatOwner`	input, outbound	input, inbound
`OutputFormatType`	input, outbound	input, inbound
the members of the parameter group "Input BIR" (see 9.2.15) with their names prefixed by "`ReferenceTemplate_`"	input, outbound	input, inbound
`no_NewTemplate`	input, outbound	input, inbound
`Payload`	input, outbound	input, inbound
`Timeout`	input, outbound	input, inbound
`no_AuditData`	input, outbound	input, inbound
`no_TemplateUUID`	input, outbound	input, inbound
`NewTemplate`	output, inbound	input, inbound
`AuditData`	output, inbound	input, inbound
`TemplateUUID`	output, inbound	input, inbound
`Return`	return, inbound	input, inbound

9.26.2 Constraints on the Parameters

9.26.2.1 An outbound value of `BSPHandle` shall be a valid representation of an integer (see 7.4) in the range 0 to 4294967295. An inbound value shall be the canonical representation of an integer (see 7.4.3) in the same range.

9.26.2.2 An outbound value of `Purpose` and `Subtype` shall be a valid representation of an integer in the range 0 to 255. An inbound value shall be the canonical representation of an integer in the same range.

9.26.2.3 An outbound value of `OutputFormatOwner` and `OutputFormatType` shall be a valid representation of an integer in the range 0 to 65535. An inbound value shall be the canonical representation of an integer in the same range.

9.26.2.4 An outbound value of the parameter group "Input BIR" shall be a valid representation of a value of the native type `BioAPI_INPUT_BIR`. An inbound value shall be the canonical representation of a value of that type.

9.26.2.5 An outbound value of `no_NewTemplate`, `no_AuditData`, and `no_TemplateUUID` shall be either a valid representation of a boolean (see 7.5) or an empty string. An inbound value shall be the canonical representation of a boolean (see 7.5.2).

9.26.2.6 An outbound value of `Payload` shall be a valid representation of an octet string (see 7.7). An inbound value shall be the canonical representation of an octet string (see 7.7.2).

9.26.2.7 An outbound value of `Timeout` shall be a valid representation of an integer in the range -2147483648 to 2147483647. An inbound value shall be the canonical representation of an integer in the same range.

9.26.2.8 An outbound value of `NewTemplate` and `AuditData` shall be either a valid representation of an integer in the range -2147483648 to 2147483647, or an empty string. An inbound value shall be either the canonical representation of an integer in the same range, or an empty string.

9.26.2.9 An outbound value of `TemplateUUID` shall be either a valid representation of a UUID (see 7.6), or an empty string. An inbound value shall be either the canonical representation of a UUID (see 7.6.3), or an empty string.

9.26.2.10 An outbound value of `return` shall be a valid representation of an integer in the range 0 to 4294967295. An inbound value shall be the canonical representation of an integer in the same range.

9.26.3 Function Invocation Input

9.26.3.1 The integer represented by the outbound value of `BSPHandle`, `Purpose`, `Subtype`, and `Timeout` shall be assigned to the native parameter with the same name.

9.26.3.2 The integer represented by the outbound value of `OutputFormatOwner` and `OutputFormatType` shall be written to the fields `FormatOwner` and `FormatType` (respectively) of a variable of type `BioAPI_BIR_BIOMETRIC_DATA_FORMAT`, whose address shall be assigned to the native parameter `OutputFormat`.

9.26.3.3 The value of type `BioAPI_INPUT_BIR` represented (see 9.2.15.4) by the outbound value of the parameter group "Input BIR" shall be written to a variable of type `BioAPI_INPUT_BIR`, whose address shall be assigned to the native parameter `ReferenceTemplate`.

9.26.3.4 If the outbound value of `no_NewTemplate` is "`true`", then the native parameter `NewTemplate` shall be set to NULL, otherwise it shall be set to the address of a variable of type `BioAPI_BIR_HANDLE`.

9.26.3.5 If the outbound value of **Payload** is an empty string, then the native parameter with the same name shall be set to NULL. Otherwise, the octet string represented by the outbound value shall be written to a memory block of sufficient size. The address and length of that memory block shall be written to the fields **Data** and **Length** (respectively) of a variable of type **BioAPI_DATA**, whose address shall be assigned to the native parameter **Payload**.

9.26.3.6 If the outbound value of **no_AuditData** is "**true**", then the native parameter **AuditData** shall be set to NULL, otherwise it shall be set to the address of a variable of type **BioAPI_BIR_HANDLE**.

9.26.3.7 If the outbound value of **no_TemplateUUID** is "**true**", then the native parameter **TemplateUUID** shall be set to NULL, otherwise it shall be set to the address of a variable of type **BioAPI_UUID**.

9.26.4 Function Invocation Output

9.26.4.1 The inbound value of **NewTemplate** shall be determined as follows. If the native parameter with the same name is NULL, then the inbound value shall be an empty string. Otherwise, the inbound value shall be the canonical representation of the integer in the variable of type **BioAPI_BIR_HANDLE** pointed to by the native parameter **NewTemplate**.

9.26.4.2 The inbound value of **AuditData** shall be determined as follows. If the native parameter with the same name is NULL, then the inbound value shall be an empty string. Otherwise, the inbound value shall be the canonical representation of the integer in the variable of type **BioAPI_BIR_HANDLE** pointed to by the native parameter **AuditData**.

9.26.4.3 The inbound value of **TemplateUUID** shall be determined as follows. If the native parameter with the same name is NULL, then the inbound value shall be an empty string. Otherwise, the inbound value value shall be the canonical representation of the UUID in the variable of type **BioAPI_UUID** pointed to by the native parameter **TemplateUUID**.

9.26.5 Bound Activity Invocation Input

9.26.5.1 The inbound value of **BSPHandle**, **Purpose**, **Subtype**, and **Timeout** shall be the canonical representation of the integer in the native parameter with the same name.

9.26.5.2 The inbound value of **OutputFormatOwner** and **OutputFormatType** shall be the canonical representation of the integer in the fields **FormatOwner** and **FormatType** (respectively) of the variable of type **BioAPI_BIR_BIOMETRIC_DATA_FORMAT** pointed to by the native parameter **OutputFormat**.

9.26.5.3 The inbound value of the parameter group "Input BIR" with the prefix "**ReferenceTemplate_**" shall be the canonical representation (see 9.2.15.5) of the value of type **BioAPI_INPUT_BIR** in the variable pointed to by the native parameter **ReferenceTemplate**.

9.26.5.4 The inbound value of **no_NewTemplate** shall be "**true**" if the native parameter **NewTemplate** is NULL, otherwise it shall be "**false**".

9.26.5.5 The inbound value of **Payload** shall be determined as follows. If the native parameter with the same name is NULL, then the inbound value shall be an empty string. Otherwise, the inbound value shall be the canonical representation (see 7.7.2) of the octet string in the memory block whose address and length are in the fields **Data** and **Length** (respectively) of the variable of type **BioAPI_DATA** pointed to by the native parameter **Payload**.

9.26.5.6 The inbound value of **no_AuditData** shall be "**true**" if the native parameter **AuditData** is NULL, otherwise it shall be "**false**".

9.26.5.7 The inbound value of **no_TemplateUUID** shall be "**true**" if the native parameter **TemplateUUID** is NULL, otherwise it shall be "**false**".

9.26.5.8 The inbound value of `return` shall be the canonical representation of the integer returned by the native function.

9.26.5.9 The inbound values of the other input parameters shall be determined as specified in 9.26.4.

9.26.6 Bound Activity Invocation Output

There are no output parameters.

9.27 BioAPI_Verify

9.27.1 Function Invocation Scheme

This function belongs to the BioAPI interface, and has the following native prototype:

```
BioAPI_RETURN BioAPI
BioAPI_Verify(
    BioAPI_HANDLE BSPHandle,
    BioAPI_FMR MaxFMRRequested,
    const BioAPI_INPUT_BIR *ReferenceTemplate,
    BioAPI_BIR_SUBTYPE Subtype,
    BioAPI_BIR_HANDLE *AdaptedBIR,
    BioAPI_BOOL *Result,
    BioAPI_FMR *FMRAchieved,
    BioAPI_DATA *Payload,
    int32_t Timeout,
    BioAPI_BIR_HANDLE *AuditData);
```

and the following parameters:

Parameter	**Function invocation (conformance testing model for frameworks)**	**Bound activity invocation (conformance testing model for applications)**
`BSPHandle`	input, outbound	input, inbound
`MaxFMRRequested`	input, outbound	input, inbound
the members of the parameter group "Input BIR" (see 9.2.15) with their names prefixed by "`ReferenceTemplate_`"	input, outbound	input, inbound
`Subtype`	input, outbound	input, inbound

`no_AdaptedBIR`	input, outbound	input, inbound
`no_Result`	input, outbound	input, inbound
`no_FMRAchieved`	input, outbound	input, inbound
`no_Payload`	input, outbound	input, inbound
`Timeout`	input, outbound	input, inbound
`no_AuditData`	input, outbound	input, inbound
`AdaptedBIR`	output, inbound	input, inbound
`Result`	output, inbound	input, inbound
`FMRAchieved`	output, inbound	input, inbound
`Payload`	output, inbound	input, inbound
`AuditData`	output, inbound	input, inbound
`Return`	return, inbound	input, inbound

9.27.2 Constraints on the Parameters

9.27.2.1 An outbound value of `BSPHandle` shall be a valid representation of an integer (see 7.4) in the range 0 to 4294967295. An inbound value shall be the canonical representation of an integer (see 7.4.3) in the same range.

9.27.2.2 An outbound value of `MaxFMRRequested` shall be a valid representation of an integer in the range -2147483648 to 2147483647. An inbound value shall be the canonical representation of an integer in the same range.

9.27.2.3 An outbound value of the parameter group "Input BIR" shall be a valid representation of a value of the native type `BioAPI_INPUT BIR`. An inbound value shall be the canonical representation of a value of that type.

9.27.2.4 An outbound value of `no_AdaptedBIR`, `no_Result`, `no_FMRAchieved`, `no_Payload`, and `no_AuditData` shall be either a valid representation of a boolean (see 7.5) or an empty string. An inbound value shall be the canonical representation of a boolean (see 7.5.2).

9.27.2.5 An outbound value of `Subtype` shall be a valid representation of an integer in the range 0 to 255. An inbound value shall be the canonical representation of an integer in the same range.

9.27.2.6 An outbound value of `Timeout` shall be a valid representation of an integer in the range -2147483648 to 2147483647. An inbound value shall be the canonical representation of an integer in the same range.

9.27.2.7 An outbound value of **`AdaptedBIR`**, **`FMRAchieved`**, and **`AuditData`** shall be either a valid representation of an integer in the range -2147483648 to 2147483647, or an empty string. An inbound value shall be the canonical representation of an integer in the same range, or an empty string.

9.27.2.8 An outbound value of **`Result`** shall be either a valid representation of a boolean, or an empty string. An inbound value shall be the canonical representation of a boolean, or an empty string.

9.27.2.9 An outbound value of **`Payload`** shall be a valid representation of an octet string. An inbound value shall be the canonical representation of an octet string.

9.27.2.10 An outbound value of **`return`** shall be a valid representation of an integer in the range 0 to 4294967295. An inbound value shall be the canonical representation of an integer in the same range.

9.27.3 Function Invocation Input

9.27.3.1 The integer represented by the outbound value of **`BSPHandle`**, **`MaxFMRRequested`**, **`Subtype`**, and **`Timeout`** shall be assigned to the native parameter with the same name.

9.27.3.2 The value of type **`BioAPI_INPUT_BIR`** represented (see 9.2.15.4) by the outbound value of the parameter group "Input BIR" shall be written to a variable of type **`BioAPI_INPUT_BIR`**, whose address shall be assigned to the native parameter **`ReferenceTemplate`**.

9.27.3.3 If the outbound value of **`no_AdaptedBIR`** is "**`true`**", then the native parameter **`AdaptedBIR`** shall be set to NULL, otherwise it shall be set to the address of a variable of type **`BioAPI_BIR_HANDLE`**.

9.27.3.4 If the outbound value of **`no_Result`** is "**`true`**", then the native parameter **`Result`** shall be set to NULL, otherwise it shall be set to the address of a variable of type **`BioAPI_BOOL`**.

9.27.3.5 If the outbound value of **`no_FMRAchieved`** is "**`true`**", then the native parameter **`FMRAchieved`** shall be set to NULL, otherwise it shall be set to the address of a variable of type **`BioAPI_FMR`**.

9.27.3.6 If the outbound value of **`no_Payload`** is "**`true`**", then the native parameter **`Payload`** shall be set to NULL, otherwise it shall be set to the address of a variable of type **`BioAPI_DATA`**.

9.27.3.7 If the outbound value of **`no_AuditData`** is "**`true`**", then the native parameter **`AuditData`** shall be set to NULL, otherwise it shall be set to the address of a variable of type **`BioAPI_BIR_HANDLE`**.

9.27.4 Function Invocation Output

9.27.4.1 The inbound value of **`AdaptedBIR`** shall be determined as follows. If the native parameter with the same name is NULL, then the inbound value shall be an empty string. Otherwise, the inbound value shall be the canonical representation of the integer in the variable of type **`BioAPI_BIR_HANDLE`** pointed to by the native parameter **`AdaptedBIR`**.

9.27.4.2 The inbound value of **`FMRAchieved`** shall be determined as follows. If the native parameter with the same name is NULL, then the inbound value shall be an empty string. Otherwise, the inbound value shall be the canonical representation of the integer in the variable of type **`BioAPI_FMR`** pointed to by the native parameter **`FMRAchieved`**.

9.27.4.3 The inbound value of **`Result`** shall be determined as follows. If the native parameter with the same name is NULL, then the inbound value shall be an empty string. Otherwise, the inbound value shall be the canonical representation of the boolean in the variable of type **`BioAPI_BOOL`** pointed to by the native parameter **`Result`**.

9.27.4.4 The inbound value of `Payload` shall be determined as follows. If the native parameter with the same name is NULL, then the inbound value shall be an empty string. Otherwise, the inbound value shall be the canonical representation of the octet string in the memory block whose address and length are in the fields `Data` and `Length` (respectively) of the variable of type `BioAPI_DATA` pointed to by the native parameter `Payload`.

9.27.4.5 The inbound value of `AuditData` shall be determined as follows. If the native parameter with the same name is NULL, then the inbound value shall be an empty string. Otherwise, the inbound value shall be the canonical representation of the integer in the variable of type `BioAPI_BIR_HANDLE` pointed to by the native parameter `AuditData`.

9.27.5 Bound Activity Invocation Input

9.27.5.1 The inbound value of `BSPHandle`, `MaxFMRRequested`, `Subtype`, and `Timeout` shall be the canonical representation of the integer in the native parameter with the same name.

9.27.5.2 The inbound value of the parameter group "Input BIR" with the prefix "`ReferenceTemplate_`" shall be the canonical representation (see 9.2.15.5) of the value of type `BioAPI_INPUT_BIR` in the variable pointed to by the native parameter `ReferenceTemplate`.

9.27.5.3 The inbound value of `no_AdaptedBIR` shall be "`true`" if the native parameter `AdaptedBIR` is NULL, otherwise it shall be "`false`".

9.27.5.4 The inbound value of `no_Result` shall be "`true`" if the native parameter `Result` is NULL, otherwise it shall be "`false`".

9.27.5.5 The inbound value of `no_FMRAchieved` shall be "`true`" if the native parameter `FMRAchieved` is NULL, otherwise it shall be "`false`".

9.27.5.6 The inbound value of `no_Payload` shall be "`true`" if the native parameter `Payload` is NULL, otherwise it shall be "`false`".

9.27.5.7 The inbound value of `no_AuditData` shall be "`true`" if the native parameter `AuditData` is NULL, otherwise it shall be "`false`".

9.27.5.8 The inbound value of `return` shall be the canonical representation of the integer returned by the native function.

9.27.5.9 The inbound values of the other input parameters shall be determined as specified in 9.27.4.

9.27.6 Bound Activity Invocation Output

There are no output parameters.

9.28 BioAPI_Identify

9.28.1 Function Invocation Scheme

This function belongs to the BioAPI interface, and has the following native prototype:

BioAPI_RETURN BioAPI
BioAPI_Identify(

BioAPI_HANDLE BSPHandle,

BioAPI_FMR MaxFMRRequested,

BioAPI_BIR_SUBTYPE Subtype,

const BioAPI_IDENTIFY_POPULATION *Population,

uint32_t TotalNumberOfTemplates,

BioAPI_BOOL Binning,

uint32_t MaxNumberOfResults,

uint32_t *NumberOfResults,

BioAPI_CANDIDATE **Candidates,

int32_t Timeout,

BioAPI_BIR_HANDLE *AuditData);

and the following parameters:

Parameter	**Function invocation (conformance testing model for frameworks)**	**Bound activity invocation (conformance testing model for applications)**
`BSPHandle`	input, outbound	input, inbound
`MaxFMRRequested`	input, outbound	input, inbound
`Subtype`	input, outbound	input, inbound
the members of the parameter group "Identify population" (see 9.2.16)	input, outbound	input, inbound
`TotalNumberOfTemplates`	input, outbound	input, inbound
`Binning`	input, outbound	input, inbound
`MaxNumberOfResults`	input, outbound	input, inbound
`no_NumberOfResults`	input, outbound	input, inbound

`no_Candidates`	input, outbound	input, inbound
`Timeout`	input, outbound	input, inbound
`no_AuditData`	input, outbound	input, inbound
`NumberOfResults`	output, inbound	input, inbound
the members of the parameter group "Candidate" (see 9.2.17) with their names prefixed by "`Candidate_1_`"	output, inbound	input, inbound
the members of the parameter group "Candidate" with their names prefixed by "`Candidate_2_`"	output, inbound	input, inbound
the members of the parameter group "Candidate" with their names prefixed by "`Candidate_3_`"	output, inbound	input, inbound
the members of the parameter group "Candidate" with their names prefixed by "`Candidate_4_`"	output, inbound	input, inbound
`AuditData`	output, inbound	input, inbound
`Return`	return, inbound	input, inbound

9.28.2 Constraints on the Parameters

9.28.2.1 An outbound value of `BSPHandle`, `TotalNumberOfTemplates`, and `MaxNumberOfResults` shall be a valid representation of an integer (see 7.4) in the range 0 to 4294967295. An inbound value shall be the canonical representation of an integer (see 7.4.3) in the same range.

9.28.2.2 An outbound value of `MaxFMRRequested` shall be a valid representation of an integer in the range -2147483648 to 2147483647. An inbound value shall be the canonical representation of an integer in the same range.

9.28.2.3 An outbound value of `Subtype` shall be a valid representation of an integer in the range 0 to 255. An inbound value shall be the canonical representation of an integer in the same range.

9.28.2.4 An outbound value of the parameter group "Identify population" shall be a valid representation of a value of the native type `BioAPI_IDENTIFY_POPULATION`. An inbound value shall be the canonical representation of a value of that type.

9.28.2.5 An outbound value of `Binning` shall be a valid representation of a boolean. An inbound value shall be the canonical representation of a boolean.

9.28.2.6 An outbound value of `no_NumberOfResults`, `no_Candidates`, and `no_AuditData` shall be either a valid representation of a boolean (see 7.5) or an empty string. An inbound value shall be the canonical representation of a boolean (see 7.5.2).

9.28.2.7 An outbound value of `Timeout` shall be a valid representation of an integer in the range -2147483648 to 2147483647. An inbound value shall be the canonical representation of an integer in the same range.

9.28.2.8 An outbound value of `NumberOfResults` shall be either a valid representation of an integer in the range 0 to 4294967295, or an empty string. An inbound value shall be either the canonical representation of an integer in the same range, or an empty string.

9.28.2.9 An outbound value of `AuditData` shall be either a valid representation of an integer in the range -2147483648 to 2147483647, or an empty string. An inbound value shall be either the canonical representation of an integer in the same range, or an empty string.

9.28.2.10 An outbound value of each parameter group "Candidate" shall be a valid representation of a value of the native type `BioAPI_CANDIDATE`. An inbound value shall be either the canonical representation of a value of that type, or a set of empty strings.

9.28.2.11 An outbound value of `return` shall be a valid representation of an integer in the range 0 to 4294967295. An inbound value shall be the canonical representation of an integer in the same range.

9.28.3 Function Invocation Input

9.28.3.1 The integer represented by the outbound value of `BSPHandle`, `MaxFMRRequested`, `Subtype`, `TotalNumberOfTemplates`, `MaxNumberOfResults`, and `Timeout` shall be assigned to the native parameter with the same name.

9.28.3.2 The value of type `BioAPI_IDENTIFY_POPULATION` represented (see 9.2.16.4) by the outbound value of the parameter group "Identify population" shall be written to a variable of type `BioAPI_IDENTIFY_POPULATION`, whose address shall be assigned to the native parameter `Population`.

9.28.3.3 The boolean represented by the outbound value of `Binning` shall be assigned to the native parameter with the same name.

9.28.3.4 If the outbound value of `no_NumberOfResults` is "`true`", then the native parameter `NumberOfResults` shall be set to NULL, otherwise it shall be set to the address of a variable of type `uint32_t`.

9.28.3.5 If the outbound value of `no_Candidates` is "`true`", then the native parameter `Candidates` shall be set to NULL, otherwise it shall be set to the address of a variable of type `BioAPI_CANDIDATE*` (a pointer)..

9.28.3.6 If the outbound value of `no_AuditData` is "`true`", then the native parameter `AuditData` shall be set to NULL, otherwise it shall be set to the address of a variable of type `BioAPI_BIR_HANDLE`.

9.28.4 Function Invocation Output

9.28.4.1 The inbound value of **NumberOfResults** shall be determined as follows. If the native parameter with the same name is NULL, then the inbound value shall be an empty string. Otherwise, the inbound value shall be the canonical representation of the integer in the variable of type **uint32_t** pointed to by the native parameter **NumberOfResults**.

9.28.4.2 The inbound value of the parameter group "Candidate" with the prefix "**Candidate_*X*_**" (with *X*=1, 2, 3, or 4) shall be determined as follows. If the native parameter **Candidates** is NULL or the variable pointed to by that native parameter is NULL or the native parameter **NumberOfResults** is NULL or the value of the integer variable pointed to by that native parameter is less than *X*, then the inbound value shall be a set of empty strings. Otherwise, the inbound value shall be the canonical representation (see 9.2.14.5) of the value of type **BioAPI_CANDIDATE** at position *X* in the array pointed to by the variable pointed to by the native parameter **Candidates**.

9.28.4.3 The inbound value of **AuditData** shall be determined as follows. If the native parameter with the same name is NULL, then the inbound value shall be an empty string. Otherwise, the inbound value shall be the canonical representation of the integer in the variable of type **BioAPI_BIR_HANDLE** pointed to by the native parameter **AuditData**.

9.28.5 Bound Activity Invocation Input

9.28.5.1 The inbound value of **BSPHandle**, **MaxFMRRequested**, **Subtype**, **TotalNumberOfTemplates**, **MaxNumberOfResults**, and **Timeout** shall be the canonical representation of the integer in the native parameter with the same name.

9.28.5.2 The inbound value of the parameter group "Identify population" shall be the canonical representation (see 9.2.16.5) of the value of type **BioAPI_IDENTIFY_POPULATION** in the variable pointed to by the native parameter **Population**.

9.28.5.3 The inbound value of **Binning** shall be the canonical representation of the boolean in the native parameter with the same name.

9.28.5.4 The inbound value of **no_NumberOfResults** shall be "**true**" if the native parameter **NumberOfResults** is NULL, otherwise it shall be "**false**".

9.28.5.5 The inbound value of **no_Candidates** shall be "**true**" if the native parameter **Candidates** is NULL, otherwise it shall be "**false**".

9.28.5.6 The inbound value of **no_AuditData** shall be "**true**" if the native parameter **AuditData** is NULL, otherwise it shall be "**false**".

9.28.5.7 The inbound value of **return** shall be the canonical representation of the integer returned by the native function.

9.28.5.8 The inbound values of the other input parameters shall be determined as specified in 9.28.4.

9.28.6 Bound Activity Invocation Output

There are no output parameters.

9.29 BioAPI_Import

9.29.1 Function Invocation Scheme

This function belongs to the BioAPI interface, and has the following native prototype:

BioAPI_RETURN BioAPI

BioAPI_Import(

BioAPI_HANDLE BSPHandle,

const BioAPI_DATA *InputData,

const BioAPI_BIR_BIOMETRIC_DATA_FORMAT *InputFormat,

const BioAPI_BIR_BIOMETRIC_DATA_FORMAT *OutputFormat,

BioAPI_BIR_PURPOSE Purpose,

BioAPI_BIR_HANDLE *ConstructedBIR);

and the following parameters:

Parameter	**Function invocation (conformance testing model for frameworks)**	**Bound activity invocation (conformance testing model for applications)**
`BSPHandle`	input, outbound	input, inbound
`InputData`	input, outbound	input, inbound
`InputFormatOwner`	input, outbound	input, inbound
`InputFormatType`	input, outbound	input, inbound
`OutputFormatOwner`	input, outbound	input, inbound
`OutputFormatType`	input, outbound	input, inbound
`Purpose`	input, outbound	input, inbound
`no_ConstructedBIR`	input, outbound	input, inbound
`ConstructedBIR`	output, inbound	input, inbound
`return`	return, inbound	input, inbound

9.29.2 Constraints on the Parameters

9.29.2.1 An outbound value of **`BSPHandle`** shall be a valid representation of an integer (see 7.4) in the range 0 to 4294967295. An inbound value shall be the canonical representation of an integer (see 7.4.3) in the same range.

9.29.2.2 An outbound value of **`InputData`** shall be a valid representation of an octet string (see 7.7). An inbound value shall be the canonical representation of an octet string (see 7.7.2).

9.29.2.3 An outbound value of **`InputFormatOwner`**, **`InputFormatType`**, **`OutputFormatOwner`**, and **`OutputFormatType`** shall be a valid representation of an integer in the range 0 to 65535. An inbound value shall be the canonical representation of an integer in the same range.

9.29.2.4 An outbound value of **`Purpose`** shall be a valid representation of an integer in the range 0 to 255. An inbound value shall be the canonical representation of an integer in the same range.

9.29.2.5 An outbound value of **`no_ConstructedBIR`** shall be either a valid representation of a boolean (see 7.5) or an empty string. An inbound value shall be the canonical representation of a boolean (see 7.5.2).

9.29.2.6 An outbound value of **`ConstructedBIR`** shall be either a valid representation of an integer in the range -2147483648 to 2147483647, or an empty string. An inbound value shall be either the canonical representation of an integer in the same range, or an empty string.

9.29.2.7 An outbound value of **`return`** shall be a valid representation of an integer in the range 0 to 4294967295. An inbound value shall be the canonical representation of an integer in the same range.

9.29.3 Function Invocation Input

9.29.3.1 The integer represented by the outbound value of **`BSPHandle`** and **`Purpose`** shall be assigned to the native parameter with the same name.

9.29.3.2 If the outbound value of **`InputData`** is an empty string, then the native parameter with the same name shall be set to NULL. Otherwise, the octet string represented by the outbound value shall be written to a memory block of sufficient size. The address and length of that memory block shall be written to the fields **`Data`** and **`Length`** (respectively) of a variable of type **`BioAPI_DATA`**, whose address shall be assigned to the native parameter **`InputData`**.

9.29.3.3 The integer represented by the outbound value of **`InputFormatOwner`** and **`InputFormatType`** shall be written to the fields **`FormatOwner`** and **`FormatType`** (respectively) of a variable of type **`BioAPI_BIR_BIOMETRIC_DATA_FORMAT`**, whose address shall be assigned to the native parameter **`InputFormat`**.

9.29.3.4 The integer represented by the outbound value of **`OutputFormatOwner`** and **`OutputFormatType`** shall be written to the fields **`FormatOwner`** and **`FormatType`** (respectively) of a variable of type **`BioAPI_BIR_BIOMETRIC_DATA_FORMAT`**, whose address shall be assigned to the native parameter **`OutputFormat`**.

9.29.3.5 If the outbound value of **`no_ConstructedBIR`** is "**`true`**", then the native parameter **`ConstructedBIR`** shall be set to NULL, otherwise it shall be set to the address of a variable of type **`BioAPI_BIR_HANDLE`**.

9.29.4 Function Invocation Output

The inbound value of `ConstructedBIR` shall be determined as follows. If the native parameter with the same name is NULL, then the inbound value shall be an empty string. Otherwise, the inbound value shall be the canonical representation of an integer in the range -2147483648 to 2147483647 (see 7.4.3). The integer shall be read from the variable of type `BioAPI_BIR_HANDLE` pointed to by the native parameter `ConstructedBIR`.

9.29.5 Bound Activity Invocation Input

9.29.5.1 The inbound value of `BSPHandle` and `Purpose` shall be the canonical representation of the integer in the native parameter with the same name.

9.29.5.2 The inbound value of `InputData` shall be determined as follows. If the native parameter with the same name is NULL, then the inbound value shall be an empty string. Otherwise, the inbound value shall be the canonical representation (see 7.7.2) of the octet string in the memory block whose address and length are in the fields `Data` and `Length` (respectively) of the variable of type `BioAPI_DATA` pointed to by the native parameter `InputData`.

9.29.5.3 The inbound value of `InputFormatOwner` and `InputFormatType` shall be the canonical representation of the integer in the fields `FormatOwner` and `FormatType` (respectively) of the variable of type `BioAPI_BIR_BIOMETRIC_DATA_FORMAT` pointed to by the native parameter `InputFormat`.

9.29.5.4 The inbound value of `OutputFormatOwner` and `OutputFormatType` shall be the canonical representation of the integer in the fields `FormatOwner` and `FormatType` (respectively) of the variable of type `BioAPI_BIR_BIOMETRIC_DATA_FORMAT` pointed to by the native parameter `OutputFormat`.

9.29.5.5 The inbound value of `no_ConstructedBIR` shall be "`true`" if the native parameter `ConstructedBIR` is NULL, otherwise it shall be "`false`".

9.29.5.6 The inbound value of `return` shall be the canonical representation of the integer returned by the native function.

9.29.5.7 The inbound values of the other input parameters shall be determined as specified in 9.29.4.

9.29.6 Bound Activity Invocation Output

There are no output parameters.

9.30 BioAPI_PresetIdentifyPopulation

9.30.1 Function Invocation Scheme

This function belongs to the BioAPI interface, and has the following native prototype:

```
BioAPI_RETURN BioAPI
BioAPI_PresetIdentifyPopulation(
    BioAPI_HANDLE BSPHandle,
    const BioAPI_IDENTIFY_POPULATION *Population);
```

and the following parameters:

Parameter	Function invocation (conformance testing model for frameworks)	Bound activity invocation (conformance testing model for applications)
`BSPHandle`	input, outbound	input, inbound
members of the parameter group "Identify population" (see 9.2.16)	input, outbound	input, inbound
`return`	return, inbound	input, inbound

9.30.2 Constraints on the Parameters

9.30.2.1 An outbound value of `BSPHandle` shall be a valid representation of an integer (see 7.4) in the range 0 to 4294967295. An inbound value shall be the canonical representation of an integer (see 7.4.3) in the same range.

9.30.2.2 An outbound value of the parameter group "Identify population" shall be a valid representation of a value of the native type `BioAPI_IDENTIFY_POPULATION`. An inbound value shall be the canonical representation of a value of that type.

9.30.2.3 An outbound value of `return` shall be a valid representation of an integer in the range 0 to 4294967295. An inbound value shall be the canonical representation of an integer in the same range.

9.30.3 Function Invocation Input

9.30.3.1 The integer represented by the outbound value of `BSPHandle` shall be assigned to the native parameter with the same name.

9.30.3.2 The value of type `BioAPI_IDENTIFY_POPULATION` represented (see 9.2.16.4) by the outbound value of the parameter group "Identify population" shall be written to a variable of type `BioAPI_IDENTIFY_POPULATION`, whose address shall be assigned to the native parameter `Population`.

9.30.4 Function Invocation Output

There are no output parameters.

9.30.5 Bound Activity Invocation Input

9.30.5.1 The inbound value of `BSPHandle` shall be the canonical representation of the integer in the native parameter with the same name.

9.30.5.2 The inbound value of the parameter group "Identify population" shall be the canonical representation (see 9.2.16.5) of the value of type `BioAPI_IDENTIFY_POPULATION` in the variable pointed to by the native parameter `Population`.

9.30.5.3 The inbound value of `return` shall be the canonical representation of the integer returned by the native function.

9.30.6 Bound Activity Invocation Output

There are no output parameters.

9.31 BioAPI_DbOpen

9.31.1 Function Invocation Scheme

This function belongs to the BioAPI interface, and has the following native prototype:

```
BioAPI_RETURN BioAPI
BioAPI_DbOpen(
    BioAPI_HANDLE BSPHandle,
    const BioAPI_UUID *DbUuid,
    BioAPI_DB_ACCESS_TYPE AccessRequest,
    BioAPI_DB_HANDLE *DbHandle,
    BioAPI_DB_MARKER_HANDLE *MarkerHandle);
```

and the following parameters:

Parameter	**Function invocation (conformance testing model for frameworks)**	**Bound activity invocation (conformance testing model for applications)**
`BSPHandle`	input, outbound	input, inbound
`DbUuid`	input, outbound	input, inbound
the members of the parameter group "Access type" (see 9.2.19)	input, outbound	input, inbound
`no_DbHandle`	input, outbound	input, inbound
`no_MarkerHandle`	input, outbound	input, inbound
`DbHandle`	output, inbound	input, inbound
`MarkerHandle`	output, inbound	input, inbound
`return`	return, inbound	input, inbound

9.31.2 Constraints on the Parameters

9.31.2.1 An outbound value of **BSPHandle** shall be a valid representation of an integer (see 7.4) in the range 0 to 4294967295. An inbound value shall be the canonical representation of an integer (see 7.4.3) in the same range.

9.31.2.2 An outbound value of **DbUuid** shall be a valid representation of a UUID (see 7.6). An inbound value shall be the canonical representation of a UUID (see 7.6.3).

9.31.2.3 An outbound value of the parameter group "Access type" shall be a valid representation of a value of the native type **BioAPI_DB_ACCESS_TYPE**. An inbound value shall be the canonical representation of a value of that type.

9.31.2.4 An outbound value of **no_DbHandle** and **no_MarkerHandle** shall be either a valid representation of a boolean (see 7.5) or an empty string. An inbound value shall be the canonical representation of a boolean (see 7.5.2).

9.31.2.5 An outbound value of **DbHandle** shall be either a valid representation of an integer in the range -2147483648 to 2147483647, or an empty string. An inbound value shall be either the canonical representation of an integer in the same range, or an empty string.

9.31.2.6 An outbound value of **MarkerHandle** shall be either a valid representation of an integer in the range 0 to 4294967295, or an empty string. An inbound value shall be either the canonical representation of an integer in the same range, or an empty string.

9.31.2.7 An outbound value of **return** shall be a valid representation of an integer in the range 0 to 4294967295. An inbound value shall be the canonical representation of an integer in the same range.

9.31.3 Function Invocation Input

9.31.3.1 The integer represented by the outbound value of **BSPHandle** shall be assigned to the native parameter with the same name.

9.31.3.2 The UUID represented by the outbound value of **DbUuid** shall be written to a variable of type **BioAPI_UUID**, whose address shall be assigned to the native parameter **DbUuid**.

9.31.3.3 The value of type **BioAPI_DB_ACCESS_TYPE** represented (see 9.2.19.4) by the outbound value of the parameter group "Access type" shall be assigned to the native parameter **AccessRequest**.

9.31.3.4 If the outbound value of **no_DbHandle** is "**true**", then the native parameter **DbHandle** shall be set to NULL, otherwise it shall be set to the address of a variable of type **BioAPI_DB_HANDLE**.

9.31.3.5 If the outbound value of **no_MarkerHandle** is "**true**", then the native parameter **MarkerHandle** shall be set to NULL, otherwise it shall be set to the address of a variable of type **BioAPI_DB_MARKER_HANDLE**.

9.31.4 Function Invocation Output

9.31.4.1 The inbound value of **DbHandle** shall be determined as follows. If the native parameter with the same name is NULL, then the inbound value shall be an empty string. Otherwise, the inbound value shall be the canonical representation of the integer in the variable of type **BioAPI_DB_HANDLE** pointed to by the native parameter **DbHandle**.

9.31.4.2 The inbound value of **MarkerHandle** shall be determined as follows. If the native parameter with the same name is NULL, then the inbound value shall be an empty string. Otherwise, the inbound value shall be the canonical representation of the integer in the variable of type **BioAPI_DB_MARKER_HANDLE** pointed to by the native parameter **MarkerHandle**.

9.31.5 Bound Activity Invocation Input

9.31.5.1 The inbound value of **`BSPHandle`** shall be the canonical representation of the integer in the native parameter with the same name.

9.31.5.2 The inbound value of **`DbUuid`** shall be the canonical representation of the UUID in the variable of type **`BioAPI_UUID`** pointed to by the native parameter **`DbUuid`**.

9.31.5.3 The inbound value of the parameter group "Access type" shall be the canonical representation (see 9.2.19.5) of the value of type **BioAPI_DB_ACCESS_TYPE** in the native parameter **AccessRequest**.

9.31.5.4 The inbound value of **`no_DbHandle`** shall be "**`true`**" if the native parameter **`DbHandle`** is NULL, otherwise it shall be "**`false`**".

9.31.5.5 The inbound value of **`no_MarkerHandle`** shall be "**`true`**" if the native parameter **`MarkerHandle`** is NULL, otherwise it shall be "**`false`**".

9.31.5.6 The inbound value of **`return`** shall be the canonical representation of the integer returned by the native function.

9.31.5.7 The inbound values of the other input parameters shall be determined as specified in 9.31.4.

9.31.6 Bound Activity Invocation Output

There are no output parameters.

9.32 BioAPI_DbClose

9.32.1 Function Invocation Scheme

This function belongs to the BioAPI interface, and has the following native prototype:

```
BioAPI_RETURN BioAPI
BioAPI_DbClose(
    BioAPI_HANDLE BSPHandle,
    BioAPI_DB_HANDLE DbHandle);
```

and the following parameters:

Parameter	**Function invocation (conformance testing model for frameworks)**	**Bound activity invocation (conformance testing model for applications)**
`BSPHandle`	input, outbound	input, inbound
`DbHandle`	input, outbound	input, inbound
`return`	return, inbound	input, inbound

9.32.2 Constraints on the Parameters

9.32.2.1 An outbound value of `BSPHandle` shall be a valid representation of an integer (see 7.4) in the range 0 to 4294967295. An inbound value shall be the canonical representation of an integer (see 7.4.3) in the same range.

9.32.2.2 An outbound value of `DbHandle` shall be a valid representation of an integer in the range -2147483648 to 2147483647. An inbound value shall be the canonical representation of an integer in the same range.

9.32.2.3 An outbound value of `return` shall be a valid representation of an integer in the range 0 to 4294967295. An inbound value shall be the canonical representation of an integer in the same range.

9.32.3 Function Invocation Input

The integer represented by the outbound value of `BSPHandle` and `DbHandle` shall be assigned to the native parameter with the same name.

9.32.4 Function Invocation Output

There are no output parameters.

9.32.5 Bound Activity Invocation Input

9.32.5.1 The inbound value of `BSPHandle` and `DbHandle` shall be the canonical representation of the integer in the native parameter with the same name.

9.32.5.2 The inbound value of `return` shall be the canonical representation of the integer returned by the native function.

9.32.6 Bound Activity Invocation Output

There are no output parameters.

9.33 BioAPI_DbCreate

9.33.1 Function Invocation Scheme

This function belongs to the BioAPI interface, and has the following native prototype:

```
BioAPI_RETURN BioAPI
BioAPI_DbCreate(
    BioAPI_HANDLE BSPHandle,
    const BioAPI_UUID *DbUuid,
    uint32_t NumberOfRecords,
    BioAPI_DB_ACCESS_TYPE AccessRequest,
    BioAPI_DB_HANDLE *DbHandle);
```

and the following parameters:

Parameter	**Function invocation (conformance testing model for frameworks)**	**Bound activity invocation (conformance testing model for applications)**
`BSPHandle`	input, outbound	input, inbound
`DbUuid`	input, outbound	input, inbound
`NumberOfRecords`	input, outbound	input, inbound
the members of the parameter group "Access type" (see 9.2.19)	input, outbound	input, inbound
`no_DbHandle`	input, outbound	input, inbound
`DbHandle`	output, inbound	input, inbound
`Return`	return, inbound	input, inbound

9.33.2 Constraints on the Parameters

9.33.2.1 An outbound value of `BSPHandle` and `NumberOfRecords` shall be a valid representation of an integer (see 7.4) in the range 0 to 4294967295. An inbound value shall be the canonical representation of an integer (see 7.4.3) in the same range.

9.33.2.2 An outbound value of `DbUuid` shall be a valid representation of a UUID (see 7.6). An inbound value shall be the canonical representation of a UUID (see 7.6.3).

9.33.2.3 An outbound value of the parameter group "Access type" shall be a valid representation of a value of the native type `BioAPI_DB_ACCESS_TYPE`. An inbound value shall be the canonical representation of a value of that type.

9.33.2.4 An outbound value of `no_DbHandle` shall be either a valid representation of a boolean (see 7.5) or an empty string. An inbound value shall be the canonical representation of a boolean (see 7.5.2).

9.33.2.5 An outbound value of `DbHandle` shall be either a valid representation of an integer in the range -2147483648 to 2147483647, or an empty string. An inbound value shall be either the canonical representation of an integer in the same range, or an empty string.

9.33.2.6 An outbound value of `return` shall be a valid representation of an integer in the range 0 to 4294967295. An inbound value shall be the canonical representation of an integer in the same range.

9.33.3 Function Invocation Input

9.33.3.1 The integer represented by the outbound value of `BSPHandle` and `NumberOfRecords` shall be assigned to the native parameter with the same name.

9.33.3.2 The UUID represented by the outbound value of `DbUuid` shall be written to a variable of type `BioAPI_UUID`, whose address shall be assigned to the native parameter `DbUuid`.

9.33.3.3 The value of type `BioAPI_DB_ACCESS_TYPE` represented (see 9.2.19.4) by the outbound value of the parameter group "Access type" shall be assigned to the native parameter `AccessRequest`.

9.33.3.4 If the outbound value of `no_DbHandle` is "`true`", then the native parameter `DbHandle` shall be set to NULL, otherwise it shall be set to the address of a variable of type `BioAPI_DB_HANDLE`.

9.33.4 Function Invocation Output

The inbound value of `DbHandle` shall be determined as follows. If the native parameter with the same name is NULL, then the inbound value shall be an empty string. Otherwise, the inbound value shall be the canonical representation of the integer in the variable of type `BioAPI_DB_HANDLE` pointed to by the native parameter `DbHandle`.

9.33.5 Bound Activity Invocation Input

9.33.5.1 The inbound value of `BSPHandle` and `NumberOfRecords` shall be the canonical representation of the integer in the native parameter with the same name.

9.33.5.2 The inbound value of `DbUuid` shall be the canonical representation of the UUID in the variable of type `BioAPI_UUID` pointed to by the native parameter `DbUuid`.

9.33.5.3 The inbound value of the parameter group "Access type" shall be the canonical representation (see 9.2.19.5) of the value of type **BioAPI_DB_ACCESS_TYPE** in the native parameter **AccessRequest**.

9.33.5.4 The inbound value of `no_DbHandle` shall be "`true`" if the native parameter `DbHandle` is NULL, otherwise it shall be "`false`".

9.33.5.5 The inbound value of `return` shall be the canonical representation of the integer returned by the native function.

9.33.5.6 The inbound values of the other input parameters shall be determined as specified in 9.33.4.

9.33.6 Bound Activity Invocation Output

There are no output parameters.

9.34 BioAPI_DbDelete

9.34.1 Function Invocation Scheme

This function belongs to the BioAPI interface, and has the following native prototype:

BioAPI_RETURN BioAPI

BioAPI_DbDelete(

BioAPI_HANDLE BSPHandle,

const BioAPI_UUID *DbUuid);

and the following parameters:

Parameter	Function invocation (conformance testing model for frameworks)	Bound activity invocation (conformance testing model for applications)
`BSPHandle`	input, outbound	input, inbound
`DbUuid`	input, outbound	input, inbound
`return`	return, inbound	input, inbound

9.34.2 Constraints on the Parameters

9.34.2.1 An outbound value of `BSPHandle` shall be a valid representation of an integer (see 7.4) in the range 0 to 4294967295. An inbound value shall be the canonical representation of an integer (see 7.4.3) in the same range.

9.34.2.2 An outbound value of `DbUuid` shall be a valid representation of a UUID (see 7.6). An inbound value shall be the canonical representation of a UUID (see 7.6.3).

9.34.2.3 An outbound value of `return` shall be a valid representation of an integer in the range 0 to 4294967295. An inbound value shall be the canonical representation of an integer in the same range.

9.34.3 Function Invocation Input

9.34.3.1 The integer represented by the outbound value of `BSPHandle` shall be assigned to the native parameter with the same name.

9.34.3.2 The UUID represented by the outbound value of `DbUuid` shall be written to a variable of type `BioAPI_UUID`, whose address shall be assigned to the native parameter `DbUuid`.

9.34.4 Function Invocation Output

There are no output parameters.

9.34.5 Bound Activity Invocation Input

9.34.5.1 The inbound value of `BSPHandle` shall be the canonical representation of the integer in the native parameter with the same name.

9.34.5.2 The inbound value of `DbUuid` shall be the canonical representation of the UUID in the variable of type `BioAPI_UUID` pointed to by the native parameter `DbUuid`.

9.34.5.3 The inbound value of `return` shall be the canonical representation of the integer returned by the native function.

9.34.6 Bound Activity Invocation Output

There are no output parameters.

9.35 BioAPI_DbSetMarker

9.35.1 Function Invocation Scheme

This function belongs to the BioAPI interface, and has the following native prototype:

```
BioAPI_RETURN BioAPI
BioAPI_DbSetMarker(
    BioAPI_HANDLE BSPHandle,
    BioAPI_DB_HANDLE DbHandle,
    const BioAPI_UUID *KeyValue,
    BioAPI_DB_MARKER_HANDLE MarkerHandle);
```

and the following parameters:

Parameter	**Function invocation (conformance testing model for frameworks)**	**Bound activity invocation (conformance testing model for applications)**
`BSPHandle`	input, outbound	input, inbound
`DbHandle`	input, outbound	input, inbound
`KeyValue`	input, outbound	input, inbound
`MarkerHandle`	input, outbound	input, inbound
`return`	return, inbound	input, inbound

9.35.2 Constraints on the Parameters

9.35.2.1 An outbound value of `BSPHandle` and `MarkerHandle` shall be a valid representation of an integer (see 7.4) in the range 0 to 4294967295. An inbound value shall be the canonical representation of an integer (see 7.4.3) in the same range.

9.35.2.2 An outbound value of `DbHandle` shall be a valid representation of an integer in the range -2147483648 to 2147483647. An inbound value shall be the canonical representation of an integer in the same range.

9.35.2.3 An outbound value of `KeyValue` shall be a valid representation of a UUID (see 7.6). An inbound value shall be the canonical representation of a UUID (see 7.6.3).

9.35.2.4 An outbound value of `return` shall be a valid representation of an integer in the range 0 to 4294967295. An inbound value shall be the canonical representation of an integer in the same range.

9.35.3 Function Invocation Input

9.35.3.1 The integer represented by the outbound value of `BSPHandle`, `DbHandle`, and `MarkerHandle` shall be assigned to the native parameter with the same name.

9.35.3.2 The UUID represented by the outbound value of `KeyValue` shall be written to a variable of type `BioAPI_UUID`, whose address shall be assigned to the native parameter `KeyValue`.

9.35.4 Function Invocation Output

There are no output parameters.

9.35.5 Bound Activity Invocation Input

9.35.5.1 The inbound value of `BSPHandle`, `DbHandle`, and `MarkerHandle` shall be the canonical representation of the integer in the native parameter with the same name.

9.35.5.2 The inbound value of `KeyValue` shall be the canonical representation of the UUID in the variable of type `BioAPI_UUID` pointed to by the native parameter `KeyValue`.

9.35.5.3 The inbound value of `return` shall be the canonical representation of the integer returned by the native function.

9.35.5.4 The inbound values of the other input parameters shall be determined as specified in 9.35.4.

9.35.6 Bound Activity Invocation Output

There are no output parameters.

9.36 BioAPI_DbFreeMarker

9.36.1 Function Invocation Scheme

This function belongs to the BioAPI interface, and has the following native prototype:

```
BioAPI_RETURN BioAPI
BioAPI_DbFreeMarker(
    BioAPI_HANDLE BSPHandle,
    BioAPI_DB_MARKER_HANDLE MarkerHandle);
```

and the following parameters:

Parameter	Function invocation (conformance testing model for frameworks)	Bound activity invocation (conformance testing model for applications)
`BSPHandle`	input, outbound	input, inbound
`MarkerHandle`	input, outbound	input, inbound
`return`	return, inbound	input, inbound

9.36.2 Constraints on the Parameters

9.36.2.1 An outbound value of `BSPHandle` and `MarkerHandle` shall be a valid representation of an integer (see 7.4) in the range 0 to 4294967295. An inbound value shall be the canonical representation of an integer (see 7.4.3) in the same range.

9.36.2.2 An outbound value of `return` shall be a valid representation of an integer in the range 0 to 4294967295. An inbound value shall be the canonical representation of an integer in the same range.

9.36.3 Function Invocation Input

The integer represented by the outbound value of `BSPHandle` and `MarkerHandle` shall be assigned to the native parameter with the same name.

9.36.4 Function Invocation Output

There are no output parameters.

9.36.5 Bound Activity Invocation Input

9.36.5.1 The inbound value of `BSPHandle` and `MarkerHandle` shall be the canonical representation of the integer in the native parameter with the same name.

9.36.5.2 The inbound value of `return` shall be the canonical representation of the integer returned by the native function.

9.36.6 Bound Activity Invocation Output

There are no output parameters.

9.37 BioAPI_DbStoreBIR

9.37.1 Function Invocation Scheme

This function belongs to the BioAPI interface, and has the following native prototype:

```
BioAPI_RETURN BioAPI
BioAPI_DbStoreBIR(
    BioAPI_HANDLE BSPHandle,
    const BioAPI_INPUT_BIR *BIRToStore,
    BioAPI_DB_HANDLE DbHandle,
    BioAPI_UUID *BirUuid);
```

and the following parameters:

Parameter	**Function invocation (conformance testing model for frameworks)**	**Bound activity invocation (conformance testing model for applications)**
`BSPHandle`	input, outbound	input, inbound
the members of the parameter group "Input BIR" (see 9.2.15) with their names prefixed by "`BIRToStore_`"	input, outbound	input, inbound
`DBHandle`	input, outbound	input, inbound
`no_BirUuid`	input, outbound	input, inbound
`BirUuid`	output, inbound	input, inbound
`return`	return, inbound	input, inbound

9.37.2 Constraints on the Parameters

9.37.2.1 An outbound value of `BSPHandle` shall be a valid representation of an integer (see 7.4) in the range 0 to 4294967295. An inbound value shall be the canonical representation of an integer (see 7.4.3) in the same range.

9.37.2.2 An outbound value of the parameter group "Input BIR" shall be a valid representation of a value of the native type `BioAPI_INPUT_BIR`. An inbound value shall be the canonical representation of a value of that type.

9.37.2.3 An outbound value of `DbHandle` shall be a valid representation of an integer in the range -2147483648 to 2147483647. An inbound value shall be the canonical representation of an integer in the same range.

9.37.2.4 An outbound value of `no_BirUuid` shall be either a valid representation of a boolean (see 7.5) or an empty string. An inbound value shall be the canonical representation of a boolean (see 7.5.2).

9.37.2.5 An outbound value of `BirUuid` shall be either a valid representation of a UUID (see 7.6), or an empty string. An inbound value shall be either the canonical representation of a UUID (see 7.6.3), or an empty string.

9.37.2.6 An outbound value of `return` shall be a valid representation of an integer in the range 0 to 4294967295. An inbound value shall be the canonical representation of an integer in the same range.

9.37.3 Function Invocation Input

9.37.3.1 The integer represented by the outbound value of `BSPHandle` and `DbHandle` shall be assigned to the native parameter with the same name.

9.37.3.2 The value of type `BioAPI_INPUT_BIR` represented (see 9.2.15.4) by the outbound value of the parameter group "Input BIR" shall be written to a variable of type `BioAPI_INPUT_BIR`, whose address shall be assigned to the native parameter `BIRToStore`.

9.37.3.3 If the outbound value of `no_BirUuid` is "`true`", then the native parameter `BirUuid` shall be set to NULL, otherwise it shall be set to the address of a variable of type `BioAPI_UUID`.

9.37.4 Function Invocation Output

The inbound value of `BirUuid` shall be determined as follows. If the native parameter with the same name is NULL, then the inbound value shall be an empty string. Otherwise, the inbound value shall be the canonical representation of the UUID in the variable of type `BioAPI_UUID` pointed to by the native parameter `BirUuid`.

9.37.5 Bound Activity Invocation Input

9.37.5.1 The inbound value of `BSPHandle` and `DbHandle` shall be the canonical representation of the integer in the native parameter with the same name.

9.37.5.2 The inbound value of the parameter group "Input BIR" shall be the canonical representation (see 9.2.15.5) of the value of type `BioAPI_INPUT_BIR` in the variable pointed to by the native parameter `BIRToStore`.

9.37.5.3 The inbound value of `no_BirUuid` shall be "`true`" if the native parameter `BirUuid` is NULL, otherwise it shall be "`false`".

9.37.5.4 The inbound value of `return` shall be the canonical representation of the integer returned by the native function.

9.37.5.5 The inbound values of the other input parameters shall be determined as specified in 9.37.4.

9.37.6 Bound Activity Invocation Output

There are no output parameters.

9.38 BioAPI_DbGetBIR

9.38.1 Function Invocation Scheme

This function belongs to the BioAPI interface, and has the following native prototype:

BioAPI_RETURN BioAPI
BioAPI_DbGetBIR(
    BioAPI_HANDLE BSPHandle,
    BioAPI_DB_HANDLE DbHandle,
    const BioAPI_UUID *KeyValue,
    BioAPI_BIR_HANDLE *RetrievedBIR,
    BioAPI_DB_MARKER_HANDLE *MarkerHandle);

and the following parameters:

Parameter	**Function invocation (conformance testing model for frameworks)**	**Bound activity invocation (conformance testing model for applications)**
`BSPHandle`	input, outbound	input, inbound
`DbHandle`	input, outbound	input, inbound
`KeyValue`	input, outbound	input, inbound
`no_RetrievedBIR`	input, outbound	input, inbound
`no_MarkerHandle`	input, outbound	input, inbound
`RetrievedBIR`	output, inbound	input, inbound
`MarkerHandle`	output, inbound	input, inbound
`return`	return, inbound	input, inbound

9.38.2 Constraints on the Parameters

9.38.2.1 An outbound value of `BSPHandle` shall be a valid representation of an integer (see 7.4) in the range 0 to 4294967295. An inbound value shall be the canonical representation of an integer (see 7.4.3) in the same range.

9.38.2.2 An outbound value of **`DbHandle`** shall be a valid representation of an integer in the range -2147483648 to 2147483647. An inbound value shall be the canonical representation of an integer in the same range.

9.38.2.3 An outbound value of **`KeyValue`** shall be a valid representation of a UUID (see 7.6). An inbound value shall be the canonical representation of a UUID (see 7.6.3).

9.38.2.4 An outbound value of **`no_RetrievedBIR`** and **`no_MarkerHandle`** shall be either a valid representation of a boolean (see 7.5) or an empty string. An inbound value shall be the canonical representation of a boolean (see 7.5.2).

9.38.2.5 An outbound value of **`RetrievedBIR`** shall be either a valid representation of an integer in the range -2147483648 to 2147483647, or an empty string. An inbound value shall be the canonical representation of an integer in the same range, or an empty string.

9.38.2.6 An outbound value of **`MarkerHandle`** shall be either a valid representation of an integer in the range 0 to 4294967295, or an empty string. An inbound value shall be the canonical representation of an integer in the same range, or an empty string.

9.38.2.7 An outbound value of **`return`** shall be a valid representation of an integer in the range 0 to 4294967295. An inbound value shall be the canonical representation of an integer in the same range.

9.38.3 Function Invocation Input

9.38.3.1 The integer represented by the outbound value of **`BSPHandle`** and **`DbHandle`** shall be assigned to the native parameter with the same name.

9.38.3.2 The UUID represented by the outbound value of **`KeyValue`** shall be written to a variable of type **`BioAPI_UUID`**, whose address shall be assigned to the native parameter **`KeyValue`**.

9.38.3.3 If the outbound value of **`no_RetrievedBIR`** is "**`true`**", then the native parameter **`RetrievedBIR`** shall be set to NULL, otherwise it shall be set to the address of a variable of type **`BioAPI_BIR_HANDLE`**.

9.38.3.4 If the outbound value of **`no_MarkerHandle`** is "**`true`**", then the native parameter **`MarkerHandle`** shall be set to NULL, otherwise it shall be set to the address of a variable of type **`BioAPI_DB_MARKER_HANDLE`**.

9.38.4 Function Invocation Output

9.38.4.1 The inbound value of **`RetrievedBIR`** shall be determined as follows. If the native parameter with the same name is NULL, then the inbound value shall be an empty string. Otherwise, the inbound value shall be the canonical representation of the integer in the variable of type **`BioAPI_BIR_HANDLE`** pointed to by the native parameter **`RetrievedBIR`**.

9.38.4.2 The inbound value of **`MarkerHandle`** shall be determined as follows. If the native parameter with the same name is NULL, then the inbound value shall be an empty string. Otherwise, the inbound value shall be the canonical representation of the integer in the variable of type **`BioAPI_DB_MARKER_HANDLE`** pointed to by the native parameter **`MarkerHandle`**.

9.38.5 Bound Activity Invocation Input

9.38.5.1 The inbound value of **`BSPHandle`** and **`DbHandle`** shall be the canonical representation of the integer in the native parameter with the same name.

9.38.5.2 The inbound value of **`KeyValue`** shall be the canonical representation of the UUID in the variable of type **`BioAPI_UUID`** pointed to by the native parameter **`KeyValue`**.

9.38.5.3 The inbound value of `no_RetrievedBIR` shall be "`true`" if the native parameter `RetrievedBIR` is NULL, otherwise it shall be "`false`".

9.38.5.4 The inbound value of `no_MarkerHandle` shall be "`true`" if the native parameter `MarkerHandle` is NULL, otherwise it shall be "`false`".

9.38.5.5 The inbound value of `return` shall be the canonical representation of the integer returned by the native function.

9.38.5.6 The inbound values of the other input parameters shall be determined as specified in 9.38.4.

9.38.6 Bound Activity Invocation Output

There are no output parameters.

9.39 BioAPI_DbGetNextBIR

9.39.1 Function Invocation Scheme

This function belongs to the BioAPI interface, and has the following native prototype:

BioAPI_RETURN BioAPI

BioAPI_DbGetNextBIR(

BioAPI_HANDLE BSPHandle,

BioAPI_DB_HANDLE DbHandle,

BioAPI_DB_MARKER_HANDLE MarkerHandle,

BioAPI_BIR_HANDLE *RetrievedBIR,

BioAPI_UUID *BirUuid);

and the following parameters:

Parameter	**Function invocation (conformance testing model for frameworks)**	**Bound activity invocation (conformance testing model for applications)**
`BSPHandle`	input, outbound	input, inbound
`DbHandle`	input, outbound	input, inbound
`MarkerHandle`	input, outbound	input, inbound
`no_RetrievedBIR`	input, outbound	input, inbound
`no_BirUuid`	input, outbound	input, inbound
`RetrievedBIR`	output, inbound	input, inbound
`BirUuid`	output, inbound	input, inbound
`return`	return, inbound	input, inbound

9.39.2 Constraints on the Parameters

9.39.2.1 An outbound value of **`BSPHandle`** and **`MarkerHandle`** shall be a valid representation of an integer (see 7.4) in the range 0 to 4294967295. An inbound value shall be the canonical representation of an integer (see 7.4.3) in the same range.

9.39.2.2 An outbound value of **`DbHandle`** shall be a valid representation of an integer in the range -2147483648 to 2147483647. An inbound value shall be the canonical representation of an integer in the same range.

9.39.2.3 An outbound value of **`no_RetrievedBIR`** and **`no_BirUuid`** shall be either a valid representation of a boolean (see 7.5) or an empty string. An inbound value shall be the canonical representation of a boolean (see 7.5.2).

9.39.2.4 An outbound value of **`RetrievedBIR`** shall be either a valid representation of an integer in the range -2147483648 to 2147483647, or an empty string. An inbound value shall be the canonical representation of an integer in the same range, or an empty string.

9.39.2.5 An outbound value of **`BirUuid`** shall be either a valid representation of a UUID (see 7.6), or an empty string. An inbound value shall be either the canonical representation of a UUID (see 7.6.3), or an empty string.

9.39.2.6 An outbound value of **`return`** shall be a valid representation of an integer in the range 0 to 4294967295. An inbound value shall be the canonical representation of an integer in the same range.

9.39.3 Function Invocation Input

9.39.3.1 The integer represented by the outbound value of **`BSPHandle`**, **`DbHandle`**, and **`MarkerHandle`** shall be assigned to the native parameter with the same name.

9.39.3.2 If the outbound value of **`no_RetrievedBIR`** is "**`true`**", then the native parameter **`RetrievedBIR`** shall be set to NULL, otherwise it shall be set to the address of a variable of type **`BioAPI_BIR_HANDLE`**.

9.39.3.3 If the outbound value of **`no_BirUuid`** is "**`true`**", then the native parameter **`BirUuid`** shall be set to NULL, otherwise it shall be set to the address of a variable of type **`BioAPI_UUID`**.

9.39.4 Function Invocation Output

9.39.4.1 The inbound value of **`RetrievedBIR`** shall be determined as follows. If the native parameter with the same name is NULL, then the inbound value shall be an empty string. Otherwise, the inbound value shall be the canonical representation of the integer in the variable of type **`BioAPI_BIR_HANDLE`** pointed to by the native parameter **`RetrievedBIR`**.

9.39.4.2 The inbound value of **`BirUuid`** shall be determined as follows. If the native parameter with the same name is NULL, then the inbound value shall be an empty string. Otherwise, the inbound value shall be the canonical representation of the UUID in the variable of type **`BioAPI_UUID`** pointed to by the native parameter **`BirUuid`**.

9.39.5 Bound Activity Invocation Input

9.39.5.1 The inbound value of **`BSPHandle`**, **`DbHandle`**, and **`MarkerHandle`** shall be the canonical representation of the integer in the native parameter with the same name.

9.39.5.2 The inbound value of **`no_RetrievedBIR`** shall be "**`true`**" if the native parameter **`RetrievedBIR`** is NULL, otherwise it shall be "**`false`**".

9.39.5.3 The inbound value of `no_BirUuid` shall be "`true`" if the native parameter `BirUuid` is NULL, otherwise it shall be "`false`".

9.39.5.4 The inbound value of `return` shall be the canonical representation of the integer returned by the native function.

9.39.5.5 The inbound values of the other input parameters shall be determined as specified in 9.39.4.

9.39.6 Bound Activity Invocation Output

There are no output parameters.

9.40 BioAPI_DbDeleteBIR

9.40.1 Function Invocation Scheme

This function belongs to the BioAPI interface, and has the following native prototype:

```
BioAPI_RETURN BioAPI
BioAPI_DbDeleteBIR(
    BioAPI_HANDLE BSPHandle,
    BioAPI_DB_HANDLE DbHandle,
    const BioAPI_UUID *KeyValue);
```

and the following parameters:

Parameter	Function invocation (conformance testing model for frameworks)	Bound activity invocation (conformance testing model for applications)
`BSPHandle`	input, outbound	input, inbound
`DbHandle`	input, outbound	input, inbound
`KeyValue`	input, outbound	input, inbound
`return`	return, inbound	input, inbound

9.40.2 Constraints on the Parameters

9.40.2.1 An outbound value of `BSPHandle` shall be a valid representation of an integer (see 7.4) in the range 0 to 4294967295. An inbound value shall be the canonical representation of an integer (see 7.4.3) in the same range.

9.40.2.2 An outbound value of `DbHandle` shall be a valid representation of an integer in the range -2147483648 to 2147483647. An inbound value shall be the canonical representation of an integer in the same range.

9.40.2.3 An outbound value of `KeyValue` shall be a valid representation of a UUID (see 7.6). An inbound value shall be the canonical representation of a UUID (see 7.6.3).

9.40.2.4 An outbound value of `return` shall be a valid representation of an integer in the range 0 to 4294967295. An inbound value shall be the canonical representation of an integer in the same range.

9.40.3 Function Invocation Input

9.40.3.1 The integer represented by the outbound value of `BSPHandle` and `DbHandle` shall be assigned to the native parameter with the same name.

9.40.3.2 The UUID represented by the outbound value of `KeyValue` shall be written to a variable of type `BioAPI_UUID`, whose address shall be assigned to the native parameter `KeyValue`.

9.40.4 Function Invocation Output

There are no output parameters.

9.40.5 Bound Activity Invocation Input

9.40.5.1 The inbound value of `BSPHandle` and `DbHandle` shall be the canonical representation of the integer in the native parameter with the same name.

9.40.5.2 The inbound value of `KeyValue` shall be the canonical representation of the UUID in the variable of type `BioAPI_UUID` pointed to by the native parameter `KeyValue`.

9.40.5.3 The inbound value of `return` shall be the canonical representation of the integer returned by the native function.

9.40.6 Bound Activity Invocation Output

There are no output parameters.

9.41 BioAPI_SetPowerMode

9.41.1 Function Invocation Scheme

This function belongs to the BioAPI interface, and has the following native prototype:

```
BioAPI_RETURN BioAPI
BioAPI_SetPowerMode(
    BioAPI_HANDLE BSPHandle,
    BioAPI_UNIT_ID UnitID,
    BioAPI_POWER_MODE PowerMode);
```

and the following parameters:

Parameter	Function invocation (conformance testing model for frameworks)	Bound activity invocation (conformance testing model for applications)
`BSPHandle`	input, outbound	input, inbound
`UnitID`	input, outbound	input, inbound
`PowerMode`	input, outbound	input, inbound
`return`	return, inbound	input, inbound

9.41.2 Constraints on the Parameters

9.41.2.1 An outbound value of `BSPHandle`, `UnitID`, and `PowerMode` shall be a valid representation of an integer (see 7.4) in the range 0 to 4294967295. An inbound value shall be the canonical representation of an integer (see 7.4.3) in the same range.

9.41.2.2 An outbound value of `return` shall be a valid representation of an integer in the range 0 to 4294967295. An inbound value shall be the canonical representation of an integer in the same range.

9.41.3 Function Invocation Input

The integer represented by the outbound value of `BSPHandle_UnitID`, and `PowerMode` shall be assigned to the native parameter with the same name.

9.41.4 Function Invocation Output

There are no output parameters.

9.41.5 Bound Activity Invocation Input

9.41.5.1 The inbound value of `BSPHandle_UnitID`, and `PowerMode` shall be the canonical representation of the integer in the native parameter with the same name.

9.41.5.2 The inbound value of `return` shall be the canonical representation of the integer returned by the native function.

9.41.6 Bound Activity Invocation Output

There are no output parameters.

9.42 BioAPI_SetIndicatorStatus

9.42.1 Function Invocation Scheme

This function belongs to the BioAPI interface, and has the following native prototype:

```
BioAPI_RETURN BioAPI
BioAPI_SetIndicatorStatus(
    BioAPI_HANDLE BSPHandle,
    BioAPI_UNIT_ID UnitID,
    BioAPI_INDICATOR_STATUS IndicatorStatus);
```

and the following parameters:

Parameter	Function invocation (conformance testing model for frameworks)	Bound activity invocation (conformance testing model for applications)
`BSPHandle`	input, outbound	input, inbound
`UnitID`	input, outbound	input, inbound
`IndicatorStatus`	input, outbound	input, inbound
`return`	return, inbound	input, inbound

9.42.2 Constraints on the Parameters

9.42.2.1 An outbound value of `BSPHandle` and `UnitID` shall be a valid representation of an integer (see 7.4) in the range 0 to 4294967295. An inbound value shall be the canonical representation of an integer (see 7.4.3) in the same range.

9.42.2.2 An outbound value of `IndicatorStatus` shall be a valid representation of an integer in the range 0 to 255. An inbound value shall be the canonical representation of an integer in the same range.

9.42.2.3 An outbound value of `return` shall be a valid representation of an integer in the range 0 to 4294967295. An inbound value shall be the canonical representation of an integer in the same range.

9.42.3 Function Invocation Input

The integer represented by the outbound value of `BSPHandle`, `UnitID`, and `IndicatorStatus` shall be assigned to the native parameter with the same name.

9.42.4 Function Invocation Output

There are no output parameters.

9.42.5 Bound Activity Invocation Input

9.42.5.1 The inbound value of `BSPHandle`, `UnitID`, and `IndicatorStatus` shall be the canonical representation of the integer in the native parameter with the same name.

9.42.5.2 The inbound value of `return` shall be the canonical representation of the integer returned by the native function.

9.42.6 Bound Activity Invocation Output

There are no output parameters.

9.43 BioAPI_GetIndicatorStatus

9.43.1 Function Invocation Scheme

This function belongs to the BioAPI interface, and has the following native prototype:

```
BioAPI_RETURN BioAPI
BioAPI_GetIndicatorStatus(
    BioAPI_HANDLE BSPHandle,
    BioAPI_UNIT_ID UnitID,
    BioAPI_INDICATOR_STATUS *IndicatorStatus);
```

and the following parameters:

Parameter	Function invocation (conformance testing models for frameworks)	Bound activity invocation (conformance testing model for applications)
`BSPHandle`	input, outbound	input, inbound
`UnitID`	input, outbound	input, inbound
`no_IndicatorStatus`	input, outbound	input, inbound
`IndicatorStatus`	output, inbound	input, inbound
`return`	return, inbound	input, inbound

9.43.2 Constraints on the Parameters

9.43.2.1 An outbound value of `BSPHandle` and `UnitID` shall be a valid representation of an integer (see 7.4) in the range 0 to 4294967295. An inbound value shall be the canonical representation of an integer (see 7.4.3) in the same range.

9.43.2.2 An outbound value of `no_IndicatorStatus` shall be either a valid representation of a boolean (see 7.5) or an empty string. An inbound value shall be the canonical representation of a boolean (see 7.5.2).

9.43.2.3 An outbound value of `IndicatorStatus` shall be either a valid representation of an integer in the range 0 to 255, or an empty string. An inbound value shall be either the canonical representation of an integer in the same range, or an empty string.

9.43.2.4 An outbound value of `return` shall be a valid representation of an integer in the range 0 to 4294967295. An inbound value shall be the canonical representation of an integer in the same range.

9.43.3 Function Invocation Input

9.43.3.1 The integer represented by the outbound value of `BSPHandle` and `UnitID` shall be assigned to the native parameter with the same name.

9.43.3.2 If the outbound value of `no_IndicatorStatus` is "`true`", then the native parameter `IndicatorStatus` shall be set to NULL, otherwise it shall be set to the address of a variable of type `BioAPI_INDICATOR_STATUS`.

9.43.4 Function Invocation Output

The inbound value of `IndicatorStatus` shall be determined as follows. If the native parameter with the same name is NULL, then the inbound value shall be an empty string. Otherwise, the inbound value shall be the canonical representation of the integer in the variable of type `BioAPI_INDICATOR_STATUS` pointed to by the native parameter `IndicatorStatus`.

9.43.5 Bound Activity Invocation Input

9.43.5.1 The inbound value of `BSPHandle` and `UnitID` shall be the canonical representation of the integer in the native parameter with the same name.

9.43.5.2 The inbound value of `no_IndicatorStatus` shall be "`true`" if the native parameter `IndicatorStatus` is NULL, otherwise it shall be "`false`".

9.43.5.3 The inbound value of `return` shall be the canonical representation of the integer returned by the native function.

9.43.5.4 The inbound values of the other input parameters shall be determined as specified in 9.43.4.

9.43.6 Bound Activity Invocation Output

There are no output parameters.

9.44 BioAPI_CalibrateSensor

9.44.1 Function Invocation Scheme

This function belongs to the BioAPI interface, and has the following native prototype:

BioAPI_RETURN BioAPI

BioAPI_CalibrateSensor(

BioAPI_HANDLE BSPHandle,

int32_t Timeout);

and the following parameters:

Parameter	Function invocation (conformance testing model for frameworks)	Bound activity invocation (conformance testing model for applications)
`BSPHandle`	input, outbound	input, inbound
`Timeout`	input, outbound	input, inbound
`return`	return, inbound	input, inbound

9.44.2 Constraints on the Parameters

9.44.2.1 An outbound value of `BSPHandle` shall be a valid representation of an integer (see 7.4) in the range 0 to 4294967295. An inbound value shall be the canonical representation of an integer (see 7.4.3) in the same range.

9.44.2.2 An outbound value of `Timeout` shall be a valid representation of an integer (see 7.4) in the range -2147483648 to 2147483647. An inbound value shall be the canonical representation of an integer (see 7.4.3) in the same range.

9.44.2.3 An outbound value of `return` shall be a valid representation of an integer in the range 0 to 4294967295. An inbound value shall be the canonical representation of an integer in the same range.

9.44.3 Function Invocation Input

The integer represented by the outbound value of `BSPHandle` and `Timeout` shall be assigned to the native parameter with the same name.

9.44.4 Function Invocation Output

There are no output parameters.

9.44.5 Bound Activity Invocation Input

9.44.5.1 The inbound value of `BSPHandle` and `Timeout` shall be the canonical representation of the integer in the native parameter with the same name.

9.44.5.2 The inbound value of `return` shall be the canonical representation of the integer returned by the native function.

9.44.6 Bound Activity Invocation Output

There are no output parameters.

9.45 BioAPI_Cancel

9.45.1 Function Invocation Scheme

This function belongs to the BioAPI interface, and has the following native prototype:

BioAPI_RETURN BioAPI

BioAPI_Cancel(

BioAPI_HANDLE BSPHandle);

and the following parameters:

Parameter	Function invocation (conformance testing model for frameworks)	Bound activity invocation (conformance testing model for applications)
`BSPHandle`	input, outbound	input, inbound
`return`	return, inbound	input, inbound

9.45.2 Constraints on the Parameters

9.45.2.1 An outbound value of `BSPHandle` shall be a valid representation of an integer (see 7.4) in the range 0 to 4294967295. An inbound value shall be the canonical representation of an integer (see 7.4.3) in the same range.

9.45.2.2 An outbound value of `return` shall be a valid representation of an integer in the range 0 to 4294967295. An inbound value shall be the canonical representation of an integer in the same range.

9.45.3 Function Invocation Input

The integer represented by the outbound value of `BSPHandle` shall be assigned to the native parameter with the same name.

9.45.4 Function Invocation Output

There are no output parameters.

9.45.5 Bound Activity Invocation Input

9.45.5.1 The inbound value of `BSPHandle` shall be the canonical representation of the integer in the native parameter with the same name.

9.45.5.2 The inbound value of `return` shall be the canonical representation of the integer returned by the native function.

9.45.6 Bound Activity Invocation Output

There are no output parameters.

9.46 BioAPI_Free

9.46.1 Function Invocation Scheme

This function belongs to the BioAPI interface, and has the following native prototype:

```
BioAPI_RETURN BioAPI
BioAPI_Free(
    void* Ptr);
```

and the following parameters:

Parameter	Function invocation (conformance testing model for frameworks)	Bound activity invocation (conformance testing model for applications)
`Ptr`	input, outbound	input, inbound
`return`	return, inbound	input, inbound

9.46.2 Constraints on the Parameters

9.46.2.1 An outbound value of `Ptr` shall be a valid representation of an integer (see 7.4) in the range 0 to 4294967295. An inbound value shall be the canonical representation of an integer (see 7.4.3) in the same range.

9.46.2.2 An outbound value of `return` shall be a valid representation of an integer in the range 0 to 4294967295. An inbound value shall be the canonical representation of an integer in the same range.

9.46.3 Function Invocation Input

The integer represented by the outbound value of `Ptr` shall be assigned to the native parameter with the same name.

9.46.4 Function Invocation Output

There are no output parameters.

9.46.5 Bound Activity Invocation Input

9.46.5.1 The inbound value of `Ptr` shall be the canonical representation of the pointer (interpreted as an integer) in the native parameter with the same name.

9.46.5.2 The inbound value of `return` shall be the canonical representation of the integer returned by the native function.

9.46.6 Bound Activity Invocation Output

There are no output parameters.

9.47 BioAPI_Util_InstallBSP

9.47.1 Function Invocation Scheme

This function belongs to the BioAPI interface, and has the following native prototype:

```
BioAPI_RETURN BioAPI
BioAPI_Util_InstallBSP (
    BioAPI_INSTALL_ACTION Action,
    BioAPI_INSTALL_ERROR *Error,
    const BioAPI_BSP_SCHEMA *BSPSchema);
```

and the following parameters:

Parameter	Function invocation (conformance testing model for frameworks)	Bound activity invocation (conformance testing model for applications)
`Action`	input, outbound	input, inbound
`no_Error`	input, outbound	input, inbound
the members of the parameter group "BSP schema" (see 9.2.10)	input, outbound	input, inbound
`ErrorCode`	output, inbound	input, inbound
`ErrorString`	output, inbound	input, inbound
`return`	return, inbound	input, inbound

9.47.2 Constraints on the Parameters

9.47.2.1 An outbound value of **Action** shall be a valid representation of an integer (see 7.4) in the range 0 to 4294967295. An inbound value shall be the canonical representation of an integer (see 7.4.3) in the same range.

9.47.2.2 An outbound value of **no_Error** shall be either a valid representation of a boolean (see 7.5), or an empty string. An inbound value shall be the canonical representation of a boolean (see 7.5.2).

9.47.2.3 An outbound value of the parameter group "BSP schema" shall be a valid representation of a value of the native type **BioAPI_BSP_SCHEMA**. An inbound value shall be either the canonical representation of a value of that type, or a set of empty strings.

9.47.2.4 An outbound value of **ErrorCode** shall be either a valid representation of an integer in the range 0 to 4294967295, or an empty string. An inbound value shall be either the canonical representation of an integer in the same range, or an empty string.

9.47.2.5 An outbound or inbound value of **ErrorString** shall be a valid representation of a character string whose UTF-8 encoding is no longer than 268 octets, and which does not contain any NUL (0) characters.

9.47.2.6 An outbound value of **return** shall be a valid representation of an integer in the range 0 to 4294967295. An inbound value shall be the canonical representation of an integer in the same range.

9.47.3 Function Invocation Input

9.47.3.1 The integer represented by the outbound value of **Action** shall be assigned to the native parameter with the same name.

9.47.3.2 If the outbound value of **no_Error** is "**true**", then the native parameter **Error** shall be set to NULL, otherwise it shall be set to the address of a variable of type **BioAPI_INSTALL_ERROR**.

9.47.3.3 The value of type **BioAPI_BSP_SCHEMA** represented (see 9.2.10.4) by the outbound value of the parameter group "BSP schema" shall be written to a variable of type **BioAPI_BSP_SCHEMA**, whose address shall be assigned to the native parameter **BSPSchema**.

9.47.4 Function Invocation Output

The inbound value of **ErrorCode** and **ErrorString** shall be determined as follows. If the native parameter **Error** is NULL, then the inbound value shall be an empty string. Otherwise, the inbound value of **ErrorCode** shall be the canonical representation of the integer in the field **ErrorCode** of the variable of type **BioAPI_INSTALL_ERROR** pointed to by the native parameter **Error**, and the inbound value of **ErrorString** shall be the character string in the field **ErrorString** of that variable.

9.47.5 Bound Activity Invocation Input

9.47.5.1 The inbound value of **Action** shall be the canonical representation of the integer in the native parameter with the same name.

9.47.5.2 The inbound value of **no_Error** shall be "**true**" if the native parameter **Error** is NULL, otherwise it shall be "**false**".

9.47.5.3 The inbound value of the parameter group "BSP schema" shall be the canonical representation (see 9.2.10.5) of the value of type **BioAPI_BSP_SCHEMA** in the variable pointed to by the native parameter **BSPSchema**.

9.47.5.4 The inbound value of `return` shall be the canonical representation of the integer returned by the native function.

9.47.5.5 The inbound values of the other input parameters shall be determined as specified in 9.47.4.

9.47.6 Bound Activity Invocation Output

There are no output parameters.

9.48 BioAPI_Util_InstallBFP

9.48.1 Function Invocation Scheme

This function belongs to the BioAPI interface, and has the following native prototype:

```
BioAPI_RETURN BioAPI
BioAPI_Util_InstallBFP (
    BioAPI_INSTALL_ACTION Action,
    BioAPI_INSTALL_ERROR *Error,
    const BioAPI_BFP_SCHEMA *BFPSchema);
```

and the following parameters:

Parameter	**Function invocation (conformance testing model for frameworks)**	**Bound activity invocation (conformance testing model for applications)**
`Action`	input, outbound	input, inbound
`no_Error`	input, outbound	input, inbound
the members of the parameter group "BFP schema" (see 9.2.11)	input, outbound	input, inbound
`ErrorCode`	output, inbound	input, inbound
`ErrorString`	output, inbound	input, inbound
`return`	return, inbound	input, inbound

9.48.2 Constraints on the Parameters

9.48.2.1 An outbound value of **`Action`** shall be a valid representation of an integer (see 7.4) in the range 0 to 4294967295. An inbound value shall be the canonical representation of an integer (see 7.4.3) in the same range.

9.48.2.2 An outbound value of **`no_Error`** shall be either a valid representation of a boolean (see 7.5), or an empty string. An inbound value shall be the canonical representation of a boolean (see 7.5.2).

9.48.2.3 An outbound value of the parameter group "BFP schema" shall be a valid representation of a value of the native type **`BioAPI_BFP_SCHEMA`**. An inbound value shall be either the canonical representation of a value of that type, or a set of empty strings.

9.48.2.4 An outbound value of **`ErrorCode`** shall be either a valid representation of an integer in the range 0 to 4294967295, or an empty string. An inbound value shall be either the canonical representation of an integer in the same range, or an empty string.

9.48.2.5 An outbound or inbound value of **`ErrorString`** shall be a valid representation of a character string whose UTF-8 encoding is no longer than 268 octets, and which does not contain any NUL (0) characters.

9.48.2.6 An outbound value of **`return`** shall be a valid representation of an integer in the range 0 to 4294967295. An inbound value shall be the canonical representation of an integer in the same range.

9.48.3 Function Invocation Input

9.48.3.1 The integer represented by the outbound value of **`Action`** shall be assigned to the native parameter with the same name.

9.48.3.2 If the outbound value of **`no_Error`** is "**`true`**", then the native parameter **`Error`** shall be set to NULL, otherwise it shall be set to the address of a variable of type **`BioAPI_INSTALL_ERROR`**.

9.48.3.3 The value of type **`BioAPI_BFP_SCHEMA`** represented (see 9.2.10.4) by the outbound value of the parameter group "BFP schema" shall be written to a variable of type **`BioAPI_BFP_SCHEMA`**, whose address shall be assigned to the native parameter **`BFPSchema`**.

9.48.4 Function Invocation Output

The inbound value of **`ErrorCode`** and **`ErrorString`** shall be determined as follows. If the native parameter **`Error`** is NULL, then the inbound value shall be an empty string. Otherwise, the inbound value of **`ErrorCode`** shall be the canonical representation of the integer in the field **`ErrorCode`** of the variable of type **`BioAPI_INSTALL_ERROR`** pointed to by the native parameter **`Error`**, and the inbound value of **`ErrorString`** shall be the character string in the field **`ErrorString`** of that variable.

9.48.5 Bound Activity Invocation Input

9.48.5.1 The inbound value of **`Action`** shall be the canonical representation of the integer in the native parameter with the same name.

9.48.5.2 The inbound value of **`no_Error`** shall be "**`true`**" if the native parameter **`Error`** is NULL, otherwise it shall be "**`false`**".

9.48.5.3 The inbound value of the parameter group "BFP schema" shall be the canonical representation (see 9.2.11.5) of the value of type **`BioAPI_BFP_SCHEMA`** in the variable pointed to by the native parameter **`BFPSchema`**.

9.48.5.4 The inbound value of `return` shall be the canonical representation of the integer returned by the native function.

9.48.5.5 The inbound values of the other input parameters shall be determined as specified in 9.48.4.

9.48.6 Bound Activity Invocation Output

There are no output parameters.

9.49 BioSPI_BSPLoad

9.49.1 Function Invocation Scheme

This function belongs to the BioSPI interface, and has the following native prototype:

```
BioAPI_RETURN BioAPI
BioSPI_BSPLoad(
    const BioAPI_UUID *BSPUuid,
    BioSPI_EventHandler BioAPINotifyCallback,
    BioSPI_BFP_ENUMERATION_HANDLER BFPEnumerationHandler,
    BioSPI_MEMORY_FREE_HANDLER  MemoryFreeHandler);
```

and the following parameters:

Parameter	Function invocation (conformance testing model for BSPs)	Bound activity invocation (conformance testing model for frameworks)
`BSPUuid`	input, outbound	input, inbound
`BioAPINotifyCallback`	input, outbound	input, inbound
`BFPEnumerationHandler`	input, outbound	input, inbound
`MemoryFreeHandler`	input, outbound	input, inbound
`return`	return, inbound	output, outbound

9.49.2 Constraints on the Parameters

9.49.2.1 An outbound value of `BSPUuid` shall be a valid representation of a UUID (see 7.6). An inbound value shall be the canonical representation of a UUID (see 7.6.3).

9.49.2.2 An outbound value of **BioAPINotifyCallback**, **BFPEnumerationHandler**, and **MemoryFreeHandler** shall be either "0" or "*". An inbound value shall be the canonical representation of an integer (see 7.4.3) in the range 0 to 4294967295.

9.49.2.3 An outbound value of **return** shall be a valid representation of an integer in the range 0 to 4294967295. An inbound value shall be the canonical representation of an integer in the same range.

9.49.3 Function Invocation Input

9.49.3.1 The UUID represented by the outbound value of **BSPUuid** shall be written to a variable of type **BioAPI_UUID**, whose address shall be assigned to the native parameter **BSPUuid**.

9.49.3.2 If the outbound value of **BioAPINotifyCallback** is "0", a NULL pointer value shall be assigned to the native parameter **BioAPINotifyCallback**. If it is "*", a non-NULL pointer value, which is the address of a native function (within the testing component) implementing the standard BioAPI interface function **BioSPI_EventHandler**, shall be assigned to the native parameter **BioAPINotifyCallback**.

NOTE If the outbound value of **BioAPINotifyCallback** is not "0", any subsequent incoming calls to the standard BioAPI interface function **BioSPI_EventHandler** will result in an invocation of the activity bound to this function, if such a binding exists.

9.49.3.3 If the outbound value of **BFPEnumerationHandler** is "0", a NULL pointer value shall be assigned to the native parameter **BFPEnumerationHandler**. If it is "*", a non-NULL pointer value, which is the address of a native function (within the testing component) implementing the standard BioAPI interface function **BioSPI_BFP_ENUMERATION_HANDLER**, shall be assigned to the native parameter **BFPEnumerationHandler**.

NOTE If the outbound value of **BFPEnumerationHandler** is not "0", any subsequent incoming calls to the standard BioAPI interface function **BioSPI_BFP_ENUMERATION_HANDLER** will result in an invocation of the activity bound to this function, if such a binding exists.

9.49.3.4 If the outbound value of **MemoryFreeHandler** is "0", a NULL pointer value shall be assigned to the native parameter **MemoryFreeHandler**. If it is "*", a non-NULL pointer value, which is the address of a native function (within the testing component) implementing the standard BioAPI interface function **BioSPI_MEMORY_FREE_HANDLER**, shall be assigned to the native parameter **MemoryFreeHandler**.

NOTE If the outbound value of **MemoryFreeHandler** is not "0", any subsequent incoming calls to the standard BioAPI interface function **BioSPI_MEMORY_FREE_HANDLER** will result in an invocation of the activity bound to this function, if such a binding exists.

9.49.4 Function Invocation Output

There are no output parameters.

9.49.5 Bound Activity Invocation Input

9.49.5.1 The inbound value of **BSPUuid** shall be the canonical representation of the UUID in the variable of type **BioAPI_UUID** pointed to by the native parameter **BSPUuid**.

9.49.5.2 The inbound value of **BioAPINotifyCallback**, **BFPEnumerationHandler**, and **MemoryFreeHandler** shall be the canonical representation of the integer in the native parameter with the same name.

9.49.6 Bound Activity Invocation Output

The integer represented by the outbound value of **return** shall be returned by the native function.

9.49.7 Default Output

If there is no activity bound to this standard BioAPI interface function, the native function shall return zero.

9.50 BioSPI_BSPUnload

9.50.1 Function Invocation Scheme

This function belongs to the BioSPI interface, and has the following native prototype:

BioAPI_RETURN BioAPI

BioSPI_BSPUnload(

const BioAPI_UUID *BSPUuid);

and the following parameters:

Parameter	Function invocation (conformance testing model for BSPs)	Bound activity invocation (conformance testing model for frameworks)
`BSPUuid`	input, outbound	input, inbound
`return`	return, inbound	output, outbound

9.50.2 Constraints on the Parameters

9.50.2.1 An outbound value of `BSPUuid` shall be a valid representation of a UUID (see 7.6). An inbound value shall be the canonical representation of a UUID (see 7.6.3).

9.50.2.2 An outbound value of `return` shall be a valid representation of an integer in the range 0 to 4294967295. An inbound value shall be the canonical representation of an integer in the same range.

9.50.3 Function Invocation Input

The UUID represented by the outbound value of `BSPUuid` shall be written to a variable of type `BioAPI_UUID`, whose address shall be assigned to the native parameter `BSPUuid`.

9.50.4 Function Invocation Output

There are no output parameters.

9.50.5 Bound Activity Invocation Input

The inbound value of `BSPUuid` shall be the canonical representation of the UUID in the variable of type `BioAPI_UUID` pointed to by the native parameter `BSPUuid`.

9.50.6 Bound Activity Invocation Output

The integer represented by the outbound value of `return` shall be returned by the native function.

9.50.7 Default Output

If there is no activity bound to this standard BioAPI interface function, the native function shall return zero.

9.51 BioSPI_BSPAttach

9.51.1 Function Invocation Scheme

This function belongs to the BioSPI interface, and has the following native prototype:

```
BioAPI_RETURN BioAPI
BioSPI_BSPAttach(
    const BioAPI_UUID *BSPUuid,
    BioAPI_VERSION Version,
    const BioAPI_UNIT_LIST_ELEMENT *UnitList,
    uint32_t NumUnits,
    BioAPI_HANDLE BSPHandle);
```

and the following parameters:

Parameter	Function invocation (conformance testing model for BSPs)	Bound activity invocation (conformance testing model for frameworks)
`BSPUuid`	input, outbound	input, inbound
`Version`	input, outbound	input, inbound
`Unit_1_UnitCategory`	input, outbound	input, inbound
`Unit_1_UnitID`	input, outbound	input, inbound
`Unit_2_UnitCategory`	input, outbound	input, inbound
`Unit_2_UnitID`	input, outbound	input, inbound
`Unit_3_UnitCategory`	input, outbound	input, inbound
`Unit_3_UnitID`	input, outbound	input, inbound
`Unit_4_UnitCategory`	input, outbound	input, inbound
`Unit_4_UnitID`	input, outbound	input, inbound

`NumUnits`	input, outbound	input, inbound
`BSPHandle`	input, outbound	input, inbound
`return`	return, inbound	output, outbound

9.51.2 Constraints on the Parameters

9.51.2.1 An outbound value of **`BSPUuid`** shall be a valid representation of a UUID (see 7.6). An inbound value shall be the canonical representation of a UUID (see 7.6.3).

9.51.2.2 An outbound value of **`Version`** shall be a valid representation of an integer (see 7.4) in the range 0 to 255. An inbound value shall be the canonical representation of an integer (see 7.4.3) in the same range.

9.51.2.3 An outbound value of **`Unit_`***X***`_UnitCategory`** and **`Unit_`***X***`_UnitID`** (with *X*=1, 2, 3, or 4) shall be either a valid representation of an integer in the range 0 to 4294967295, or an empty string. An inbound value shall be either the canonical representation of an integer (see 7.4.3) in the same range, or an empty string.

9.51.2.4 An outbound value of **`NumUnits`** shall be a valid representation of an integer in the range 0 to 4. An inbound value shall be the canonical representation of an integer in the range 0 to 4294967295.

9.51.2.5 An outbound value of **`BSPHandle`** shall be a valid representation of an integer (see 7.4) in the range 0 to 4294967295. An inbound value shall be the canonical representation of an integer (see 7.4.3) in the same range.

9.51.2.6 An outbound value of **`return`** shall be a valid representation of an integer in the range 0 to 4294967295. An inbound value shall be the canonical representation of an integer in the same range.

9.51.3 Function Invocation Input

9.51.3.1 The UUID represented by the outbound value of **`BSPUuid`** shall be written to a variable of type **`BioAPI_UUID`**, whose address shall be assigned to the native parameter **`BSPUuid`**.

9.51.3.2 The integer represented by the outbound value of **`Version`**, **`NumUnits`**, and **`BSPHandle`** shall be assigned to the native parameter with the same name.

9.51.3.3 The integer represented by the outbound value of **`Unit_`***X***`_UnitCategory`** and **`Unit_`***X***`_UnitID`** (with *X*=1, 2, 3, or 4), or zero if that value is an empty string, shall be written to the fields **`UnitCategory`** and **`UnitID`** (respectively) of the element at position *X* of an array of four elements of type **`BioAPI_UNIT_LIST_ELEMENT`**. The address of that array shall be assigned to the native parameter **`UnitList`**.

9.51.4 Function Invocation Output

There are no output parameters.

9.51.5 Bound Activity Invocation Input

9.51.5.1 The inbound value of `BSPUuid` shall be the canonical representation of the UUID in the variable of type `BioAPI_UUID` pointed to by the native parameter `BSPUuid`.

9.51.5.2 The inbound value of `Version`, `NumUnits`, and `BSPHandle` shall be the canonical representation of the integer in the native parameter with the same name.

9.51.5.3 The inbound value of `Unit_`*X*`_UnitCategory` and `Unit_`*X*`_UnitID` (with *X*=1, 2, 3, or 4) shall be determined as follows. If the native parameter `NumUnits` is less than *X*, then the inbound value shall be an empty string. Otherwise, the inbound value shall be the canonical representation of the integer in the fields `UnitCategory` and `UnitID` (respectively) of the element at position *X* of the array of elements of type `BioAPI_UNIT_LIST_ELEMENT` pointed to by the native parameter `UnitList`.

9.51.6 Bound Activity Invocation Output

The integer represented by the outbound value of `return` shall be returned by the native function.

9.51.7 Default Output

If there is no activity bound to this standard BioAPI interface function, the native function shall return zero.

9.52 BioSPI_BSPDetach

9.52.1 Function Invocation Scheme

This function belongs to the BioSPI interface, and has the following native prototype:

```
BioAPI_RETURN BioAPI
BioSPI_BSPDetach(
    BioAPI_HANDLE BSPHandle);
```

and the following parameters:

Parameter	**Function invocation (conformance testing model for BSPs)**	**Bound activity invocation (conformance testing model for frameworks)**
`BSPHandle`	input, outbound	input, inbound
`return`	return, inbound	output, outbound

9.52.2 Constraints on the Parameters

9.52.2.1 An outbound value of `BSPHandle` shall be a valid representation of an integer (see 7.4) in the range 0 to 4294967295. An inbound value shall be the canonical representation of an integer (see 7.4.3) in the same range.

9.52.2.2 An outbound value of `return` shall be a valid representation of an integer in the range 0 to 4294967295. An inbound value shall be the canonical representation of an integer in the same range.

9.52.3 Function Invocation Input

The integer represented by the outbound value of `BSPHandle` shall be assigned to the native parameter with the same name.

9.52.4 Function Invocation Output

There are no output parameters.

9.52.5 Bound Activity Invocation Input

The inbound value of `BSPHandle` shall be the canonical representation of the integer in the native parameter with the same name.

9.52.6 Bound Activity Invocation Output

The integer represented by the outbound value of `return` shall be returned by the native function.

9.52.7 Default Output

If there is no activity bound to this standard BioAPI interface function, the native function shall return zero.

9.53 BioSPI_QueryUnits

9.53.1 Function Invocation Scheme

This function belongs to the BioSPI interface, and has the following native prototype:

```
BioAPI_RETURN BioAPI
BioSPI_QueryUnits(
    const BioAPI_UUID *BSPUuid,
    BioAPI_UNIT_SCHEMA **UnitSchemaArray,
    uint32_t *NumberOfElements);
```

and the following parameters:

Parameter	Function invocation (conformance testing model for BSPs)	Bound activity invocation (conformance testing model for frameworks)
`BSPUuid`	input, outbound	input, inbound
`no_UnitSchemaArray`	input, outbound	input, inbound
the members of the parameter group "Unit schema" (see 9.2.12) with their names prefixed by "`UnitSchema_1_`"	output, inbound	output, outbound

the members of the parameter group "Unit schema" with their names prefixed by "`UnitSchema_2_`"	output, inbound	output, outbound
the members of the parameter group "Unit schema" with their names prefixed by "`UnitSchema_3_`"	output, inbound	output, outbound
the members of the parameter group "Unit schema" with their names prefixed by "`UnitSchema_4_`"	output, inbound	output, outbound
`no_NumberOfElements`	input, outbound	input, inbound
`NumberOfElements`	output, inbound	output, outbound
`Return`	return, inbound	output, outbound

The specification of the constraints on the parameters, function invocation input, function invocation output, and bound activity invocation input of the function `BioAPI_QueryUnits` also applies to this standard BioAPI interface function.

9.53.2 Bound Activity Invocation Output

9.53.2.1 If the native parameter `UnitSchemaArray` is not NULL and the integer (say, *N*) represented by the outbound value of `NumberOfElements` is greater than 0, then a memory block of sufficient size to contain an array of *N* elements of type `BioAPI_UNIT_SCHEMA` shall be allocated and its address shall be written to the variable pointed to by the native parameter `UnitSchemaArray`.

9.53.2.2 The outbound value of the parameter group "Unit schema" with the prefix `UnitSchema_`*X*`_` (with *X*=1, 2, 3, or 4) shall be processed as follows. If the native parameter `UnitSchemaArray` is NULL or the integer represented by the outbound value of `NumberOfElements` is less than *X*, then the outbound value of the parameter group shall be ignored. Otherwise, the value of type `BioAPI_UNIT_SCHEMA` represented by the outbound value of the parameter group shall be written to the element at position *X* of the array of elements of type `BioAPI_UNIT_SCHEMA` pointed to by the variable pointed to by the native parameter `UnitSchemaArray`.

9.53.2.3 The outbound value of `NumberOfElements` shall be processed as follows. If the native parameter with the same name is NULL, then the outbound value shall be ignored. Otherwise, the integer represented by the outbound value shall be written to the variable of type `uint32_t` pointed to by the native parameter `NumberOfElements`.

9.53.2.4 The integer represented by the outbound value of `return` shall be returned by the native function.

9.53.3 Default Output

If there is no activity bound to this standard BioAPI interface function, then:

a) the variable pointed to by the native parameter `NumberOfElements` shall be set to zero;

b) the variable pointed to by the native parameter `UnitSchemaArray` shall be set to NULL; and

c) the native function shall return zero.

9.54 BioSPI_QueryBFPs

9.54.1 Function Invocation Scheme

This function belongs to the BioSPI interface, and has the following native prototype:

```
BioAPI_RETURN BioAPI
BioSPI_QueryBFPs(
    const BioAPI_UUID *BSPUuid,
    BioAPI_BFP_LIST_ELEMENT **BFPList,
    uint32_t *NumberOfElements);
```

and the following parameters:

Parameter	Function invocation (conformance testing model for BSPs)	Bound activity invocation (conformance testing model for frameworks)
`BSPHandle`	input, outbound	input, inbound
`no_BFPList`	input, outbound	input, inbound
`BFP_1_BFPCategory`	output, inbound	output, outbound
`BFP_1_BFPUuid`	output, inbound	output, outbound
`BFP_2_BFPCategory`	output, inbound	output, outbound
`BFP_2_BFPUuid`	output, inbound	output, outbound
`BFP_3_BFPCategory`	output, inbound	output, outbound
`BFP_3_BFPUuid`	output, inbound	output, outbound
`BFP_4_BFPCategory`	output, inbound	output, outbound

`BFP_4_BFPUuid`	output, inbound	output, outbound
`no_NumberOfElements`	input, outbound	input, inbound
`NumberOfElements`	output, inbound	output, outbound
`return`	return, inbound	output, outbound

The specification of the constraints on the parameters, function invocation input, function invocation output, and bound activity invocation input of the function `BioAPI_QueryBFPs` also applies to this standard BioAPI interface function.

9.54.2 Bound Activity Invocation Output

9.54.2.1 If the native parameter `BFPList` is not NULL and the integer (say, *N*) represented by the outbound value of `NumberOfElements` is greater than 0, then a memory block of sufficient size to contain an array of *N* elements of type `BioAPI_BFP_LIST_ELEMENT` shall be allocated and its address shall be written to the variable pointed to by the native parameter `BFPList`.

9.54.2.2 The outbound value of `BFP_`*X*`_BFPCategory` and `BFP_`*X*`_BFPUuid` (with *X*=1, 2, 3, or 4) shall be processed as follows. If the native parameter `BFPList` is NULL or the integer represented by the outbound value of `NumberOfElements` is less than *X*, then the outbound value of `BFP_`*X*`_BFPCategory` and `BFP_`*X*`_BFPUuid` shall be ignored. Otherwise, the integer represented by the outbound value of `BFP_`*X*`_BFPCategory` and the UUID represented by the outbound value of `BFP_`*X*`_BFPUuid` shall be written to the fields `BFPCategory` and `BFPUuid` (respectively) of the element at position *X* of the array of elements of type `BioAPI_BFP_LIST_ELEMENT` pointed to by the variable pointed to by the native parameter `BFPList`.

9.54.2.3 The outbound value of `NumberOfElements` shall be processed as follows. If the native parameter with the same name is NULL, then the outbound value shall be ignored. Otherwise, the integer represented by the outbound value shall be written to the variable of type `uint32_t` pointed to by the native parameter `NumberOfElements`.

9.54.2.4 The integer represented by the outbound value of `return` shall be returned by the native function.

9.54.3 Default Output

If there is no activity bound to this standard BioAPI interface function, then:

a) the variable pointed to by the native parameter `NumberOfElements` shall be set to zero;

b) the variable pointed to by the native parameter `BFPList` shall be set to NULL; and

c) the native function shall return zero.

9.55 BioSPI_ControlUnit

9.55.1 Function Invocation Scheme

This function belongs to the BioSPI interface, and has the following native prototype:

BioAPI_RETURN BioAPI

BioSPI_ControlUnit(

    BioAPI_HANDLE BSPHandle,

```
    BioAPI_UNIT_ID UnitID,
    uint32_t ControlCode,
    const BioAPI_DATA *InputData,
    BioAPI_DATA *OutputData
);
```

and the following parameters:

Parameter	Function invocation (conformance testing model for BSPs)	Bound activity invocation (conformance testing model for frameworks)
`BSPHandle`	input, outbound	input, inbound
`UnitID`	input, outbound	input, inbound
`ControlCode`	input, outbound	input, inbound
`InputData`	input, outbound	input, inbound
`no_OutputData`	input, outbound	input, inbound
`OutputData`	output, inbound	output, outbound
`return`	return, inbound	output, outbound

The specification of the constraints on the parameters, function invocation input, function invocation output, and bound activity invocation input of the function `BioAPI_ControlUnit` also applies to this standard BioAPI interface function.

9.55.2 Bound Activity Invocation Output

9.55.2.1 The outbound value of `OutputData` shall be processed as follows. If the native parameter with the same name is NULL, then the outbound value shall be ignored. Otherwise, the octet string represented by the outbound value shall be written to a memory block, whose address and size shall be written to the fields `Data` and `Length` (respectively) of the variable of type `BioAPI_DATA` pointed to by the native parameter `OutputData`.

9.55.2.2 The integer represented by the outbound value of `return` shall be returned by the native function.

9.55.3 Default Output

If there is no activity bound to this standard BioAPI interface function, then:

a) the fields `Data` and `Length` of the variable pointed to by the native parameter `OutputData` shall be both set to zero; and

b) the native function shall return zero.

9.56 BioSPI_FreeBIRHandle

9.56.1 Function Invocation Scheme

This function belongs to the BioSPI interface, and has the following native prototype:

```
BioAPI_RETURN BioAPI
BioSPI_FreeBIRHandle(
    BioAPI_HANDLE BSPHandle,
    BioAPI_BIR_HANDLE Handle);
```

and the following parameters:

Parameter	**Function invocation (conformance testing model for BSPs)**	**Bound activity invocation (conformance testing model for frameworks)**
`BSPHandle`	input, outbound	input, inbound
`Handle`	input, outbound	input, inbound
`return`	return, inbound	output, outbound

The specification of the constraints on the parameters, function invocation input, function invocation output, and bound activity invocation input of the function `BioAPI_FreeBIRHandle` also applies to this standard BioAPI interface function.

9.56.2 Bound Activity Invocation Output

The integer represented by the outbound value of `return` shall be returned by the native function.

9.56.3 Default Output

If there is no activity bound to this standard BioAPI interface function, the native function shall return zero.

9.57 BioSPI_GetBIRFromHandle

9.57.1 Function Invocation Scheme

This function belongs to the BioSPI interface, and has the following native prototype:

```
BioAPI_RETURN BioAPI
BioSPI_GetBIRFromHandle(
    BioAPI_HANDLE BSPHandle,
    BioAPI_BIR_HANDLE Handle,
    BioAPI_BIR *BIR);
```

and the following parameters:

Parameter	**Function invocation (conformance testing model for BSPs)**	**Bound activity invocation (conformance testing model for frameworks)**
`BSPHandle`	input, outbound	input, inbound
`Handle`	input, outbound	input, inbound
`no_BIR`	input, outbound	input, inbound
the members of the parameter group "BIR" (see 9.2.14)	output, inbound	output, outbound
`return`	return, inbound	output, outbound

The specification of the constraints on the parameters, function invocation input, function invocation output, and bound activity invocation input of the function `BioAPI_GetBIRFromHandle` also applies to this standard BioAPI interface function.

9.57.2 Bound Activity Invocation Output

9.57.2.1 The outbound value of the parameter group "BIR" shall be processed as follows. If the native parameter `BIR` is NULL, then the outbound value shall be ignored. Otherwise, the value of type `BioAPI_BIR` represented by the outbound value shall be written to the variable pointed to by the native parameter `BIR`.

9.57.2.2 The integer represented by the outbound value of `return` shall be returned by the native function.

9.57.3 Default Output

If there is no activity bound to this standard BioAPI interface function, then:

a) the variable pointed to by the native parameter **`BIR`** shall have all its fields set to zero; and

b) the native function shall return zero.

9.58 BioSPI_GetHeaderFromHandle

9.58.1 Function Invocation Scheme

This function belongs to the BioSPI interface, and has the following native prototype:

```
BioAPI_RETURN BioAPI
BioSPI_GetHeaderFromHandle(
    BioAPI_HANDLE BSPHandle,
    BioAPI_BIR_HANDLE Handle,
    BioAPI_BIR_HEADER *Header);
```

and the following parameters:

Parameter	Function invocation (conformance testing model for BSPs)	Bound activity invocation (conformance testing model for frameworks)
`BSPHandle`	input, outbound	input, inbound
`Handle`	input, outbound	input, inbound
`no_Header`	input, outbound	input, inbound
the members of the parameter group "BIR header" (see 9.2.13)	output, inbound	output, outbound
`return`	return, inbound	output, outbound

The specification of the constraints on the parameters, function invocation input, function invocation output, and bound activity invocation input of the function **`BioAPI_GetHeaderFromHandle`** also applies to this standard BioAPI interface function.

9.58.2 Bound Activity Invocation Output

9.58.2.1 The outbound value of the parameter group "BIR header" shall be processed as follows. If the native parameter **`Header`** is NULL, then the outbound value shall be ignored. Otherwise, the value of type **`BioAPI_BIR_HEADER`** represented by the outbound value shall be written to the variable pointed to by the native parameter **`Header`**.

9.58.2.2 The integer represented by the outbound value of **`return`** shall be returned by the native function.

9.58.3 Default Output

If there is no activity bound to this standard BioAPI interface function, then:

a) the variable pointed to by the native parameter **`Header`** shall have all its fields set to zero; and

b) the native function shall return zero.

9.59 BioSPI_EnableEvents

9.59.1 Function Invocation Scheme

This function belongs to the BioSPI interface, and has the following native prototype:

BioAPI_RETURN BioAPI

BioSPI_EnableEvents(

BioAPI_HANDLE BSPHandle,

BioAPI_EVENT_MASK Events);

and the following parameters:

Parameter	**Function invocation (conformance testing model for BSPs)**	**Bound activity invocation (conformance testing model for frameworks)**
`BSPHandle`	input, outbound	input, inbound
the members of the parameter group "Events" (see 9.2.4)	input, outbound	input, inbound
`return`	return, inbound	output, outbound

The specification of the constraints on the parameters, function invocation input, function invocation output, and bound activity invocation input of the function **`BioAPI_EnableEvents`** also applies to this standard BioAPI interface function.

9.59.2 Bound Activity Invocation Output

The integer represented by the outbound value of `return` shall be returned by the native function.

9.59.3 Default Output

If there is no activity bound to this standard BioAPI interface function, the native function shall return zero.

9.60 BioSPI_SetGUICallbacks

9.60.1 Function Invocation Scheme

This function belongs to the BioSPI interface, and has the following native prototype:

```
BioAPI_RETURN BioAPI
BioSPI_SetGUICallbacks(
    BioAPI_HANDLE BSPHandle,
    BioAPI_GUI_STREAMING_CALLBACK GuiStreamingCallback,
    void *GuiStreamingCallbackCtx,
    BioAPI_GUI_STATE_CALLBACK GuiStateCallback,
    void *GuiStateCallbackCtx);
```

and the following parameters:

Parameter	Function invocation (conformance testing model for BSPs)	Bound activity invocation (conformance testing model for frameworks)
`BSPHandle`	input, outbound	input, inbound
`GuiStreamingCallback`	input, outbound	input, inbound
`GuiStreamingCallbackCtx`	input, outbound	input, inbound
`GuiStateCallback`	input, outbound	input, inbound
`GuiStateCallbackCtx`	input, outbound	input, inbound
`return`	return, inbound	output, outbound

The specification of the constraints on the parameters, function invocation input, function invocation output, and bound activity invocation input of the function `BioAPI_SetGUICallbacks` also applies to this standard BioAPI interface function.

9.60.2 Bound Activity Invocation Output

The integer represented by the outbound value of `return` shall be returned by the native function.

9.60.3 Default Output

If there is no activity bound to this standard BioAPI interface function, the native function shall return zero.

9.61 BioSPI_Capture

9.61.1 Function Invocation Scheme

This function belongs to the BioSPI interface, and has the following native prototype:

```
BioAPI_RETURN BioAPI
BioSPI_Capture(
    BioAPI_HANDLE BSPHandle,
    BioAPI_BIR_PURPOSE Purpose,
    BioAPI_BIR_SUBTYPE Subtype,
    const BioAPI_BIR_BIOMETRIC_DATA_FORMAT *OutputFormat,
    BioAPI_BIR_HANDLE *CapturedBIR,
    int32_t Timeout,
    BioAPI_BIR_HANDLE *AuditData);
```

and the following parameters:

Parameter	**Function invocation (conformance testing model for BSPs)**	**Bound activity invocation (conformance testing model for frameworks)**
`BSPHandle`	input, outbound	input, inbound
`Purpose`	input, outbound	input, inbound
`Subtype`	input, outbound	input, inbound
`OutputFormatOwner`	input, outbound	input, inbound
`OutputFormatType`	input, outbound	input, inbound
`no_CapturedBIR`	input, outbound	input, inbound
`Timeout`	input, outbound	input, inbound

`no_AuditData`	input, outbound	input, inbound
`CapturedBIR`	output, inbound	output, outbound
`AuditData`	output, inbound	output, outbound
`return`	return, inbound	output, outbound

The specification of the constraints on the parameters, function invocation input, function invocation output, and bound activity invocation input of the function `BioAPI_Capture` also applies to this standard BioAPI interface function.

9.61.2 Bound Activity Invocation Output

9.61.2.1 The outbound value of `CapturedBIR` shall be processed as follows. If the native parameter with the same name is NULL, then the outbound value shall be ignored. Otherwise, the integer represented by the outbound value shall be written to the variable of type `BioAPI_BIR_HANDLE` pointed to by the native parameter `CapturedBIR`.

9.61.2.2 The outbound value of `AuditData` shall be processed as follows. If the native parameter with the same name is NULL, then the outbound value shall be ignored. Otherwise, the integer represented by the outbound value shall be written to the variable of type `BioAPI_BIR_HANDLE` pointed to by the native parameter `AuditData`.

9.61.2.3 The integer represented by the outbound value of `return` shall be returned by the native function.

9.61.3 Default Output

If there is no activity bound to this standard BioAPI interface function, then:

a) the variable pointed to by the native parameter `CapturedBIR` shall be set to zero;

b) the variable pointed to by the native parameter `AuditData` shall be set to zero; and

c) the native function shall return zero.

9.62 BioSPI_CreateTemplate

9.62.1 Function Invocation Scheme

This function belongs to the BioSPI interface, and has the following native prototype:

```
BioAPI_RETURN BioAPI
BioSPI_CreateTemplate(
    BioAPI_HANDLE BSPHandle,
    const BioAPI_INPUT_BIR *CapturedBIR,
    const BioAPI_INPUT_BIR *ReferenceTemplate,
    const BioAPI_BIR_BIOMETRIC_DATA_FORMAT *OutputFormat,
    BioAPI_BIR_HANDLE *NewTemplate,
    const BioAPI_DATA *Payload,
```

BioAPI_UUID *TemplateUUID);

and the following parameters:

Parameter	Function invocation (conformance testing model for BSPs)	Bound activity invocation (conformance testing model for frameworks)
`BSPHandle`	input, outbound	input, inbound
the members of the parameter group "Input BIR" (see 9.2.15) with their names prefixed by "`CapturedBIR_`"	input, outbound	input, inbound
the members of the parameter group "Input BIR" with their names prefixed by "`ReferenceTemplate_`"	input, outbound	input, inbound
`OutputFormatOwner`	input, outbound	input, inbound
`OutputFormatType`	input, outbound	input, inbound
`no_NewTemplate`	input, outbound	input, inbound
`Payload`	input, outbound	input, inbound
`no_TemplateUUID`	input, outbound	input, inbound
`NewTemplate`	output, inbound	output, outbound
`TemplateUUID`	output, inbound	output, outbound
`Return`	return, inbound	output, outbound

The specification of the constraints on the parameters, function invocation input, function invocation output, and bound activity invocation input of the function `BioAPI_CreateTemplate` also applies to this standard BioAPI interface function.

9.62.2 Bound Activity Invocation Output

9.62.2.1 The outbound value of `NewTemplate` shall be processed as follows. If the native parameter with the same name is NULL, then the outbound value shall be ignored. Otherwise, the integer represented by the outbound value shall be written to the variable of type `BioAPI_BIR_HANDLE` pointed to by the native parameter `NewTemplate`.

9.62.2.2 The outbound value of `TemplateUUID` shall be processed as follows. If the native parameter with the same name is NULL, then the outbound value shall be ignored. Otherwise, the UUID represented by the outbound value shall be written to the variable of type `BioAPI_UUID` pointed to by the native parameter `TemplateUUID`.

9.62.2.3 The integer represented by the outbound value of `return` shall be returned by the native function.

9.62.3 Default Output

If there is no activity bound to this standard BioAPI interface function, then:

a) the variable pointed to by the native parameter `NewTemplate` shall be set to zero;

b) the variable pointed to by the native parameter `TemplateUUID` shall be set to the UUID "`00000000-0000-0000-0000-000000000000`"; and

c) the native function shall return zero.

9.63 BioSPI_Process

9.63.1 Function Invocation Scheme

This function belongs to the BioSPI interface, and has the following native prototype:

```
BioAPI_RETURN BioAPI
BioSPI_Process(
    BioAPI_HANDLE BSPHandle,
    const BioAPI_INPUT_BIR *CapturedBIR,
    const BioAPI_BIR_BIOMETRIC_DATA_FORMAT *OutputFormat,
    BioAPI_BIR_HANDLE *ProcessedBIR);
```

and the following parameters:

Parameter	**Function invocation (conformance testing model for BSPs)**	**Bound activity invocation (conformance testing model for frameworks)**
`BSPHandle`	input, outbound	input, inbound
the members of the parameter group "Input BIR" (see 9.2.15) with their names prefixed by "`CapturedBIR_`"	input, outbound	input, inbound

`OutputFormatOwner`	input, outbound	input, inbound
`OutputFormatType`	input, outbound	input, inbound
`no_ProcessedBIR`	input, outbound	input, inbound
`ProcessedBIR`	output, inbound	output, outbound
`return`	return, inbound	output, outbound

The specification of the constraints on the parameters, function invocation input, function invocation output, and bound activity invocation input of the function `BioAPI_Process` also applies to this standard BioAPI interface function.

9.63.2 Bound Activity Invocation Output

9.63.2.1 The outbound value of `ProcessedBIR` shall be processed as follows. If the native parameter with the same name is NULL, then the outbound value shall be ignored. Otherwise, the integer represented by the outbound value shall be written to the variable of type `BioAPI_BIR_HANDLE` pointed to by the native parameter `ProcessedBIR`.

9.63.2.2 The integer represented by the outbound value of `return` shall be returned by the native function.

9.63.3 Default Output

If there is no activity bound to this standard BioAPI interface function, then:

a) the variable pointed to by the native parameter `ProcessedBIR` shall be set to zero; and

b) the native function shall return zero.

9.64 BioSPI_ProcessWithAuxBIR

9.64.1 Function Invocation Scheme

This function belongs to the BioSPI interface, and has the following native prototype:

```
BioAPI_RETURN BioAPI
BioSPI_ProcessWithAuxBIR(
    BioAPI_HANDLE BSPHandle,
    const BioAPI_INPUT_BIR *CapturedBIR,
    const BioAPI_INPUT_BIR *AuxiliaryData,
    const BioAPI_BIR_BIOMETRIC_DATA_FORMAT *OutputFormat,
    BioAPI_HANDLE *ProcessedBIR);
```

and the following parameters:

Parameter	Function invocation (conformance testing model for BSPs)	Bound activity invocation (conformance testing model for frameworks)
`BSPHandle`	input, outbound	input, inbound
the members of the parameter group "Input BIR" (see 9.2.15) with their names prefixed by "`CapturedBIR_`"	input, outbound	input, inbound
the members of the parameter group "Input BIR" with their names prefixed by "`AuxiliaryData_`"	input, outbound	input, inbound
`OutputFormatOwner`	input, outbound	input, inbound
`OutputFormatType`	input, outbound	input, inbound
`no_ProcessedBIR`	input, outbound	input, inbound
`ProcessedBIR`	output, inbound	output, outbound
`return`	return, inbound	output, outbound

The specification of the constraints on the parameters, function invocation input, function invocation output, and bound activity invocation input of the function `BioAPI_ProcessWithAuxBIR` also applies to this standard BioAPI interface function.

9.64.2 Bound Activity Invocation Output

9.64.2.1 The outbound value of `ProcessedBIR` shall be processed as follows. If the native parameter with the same name is NULL, then the outbound value shall be ignored. Otherwise, the integer represented by the outbound value shall be written to the variable of type `BioAPI_BIR_HANDLE` pointed to by the native parameter `ProcessedBIR`.

9.64.2.2 The integer represented by the outbound value of `return` shall be returned by the native function.

9.64.3 Default Output

If there is no activity bound to this standard BioAPI interface function, then:

a) the variable pointed to by the native parameter `ProcessedBIR` shall be set to zero; and

b) the native function shall return zero.

9.65 BioSPI_VerifyMatch

9.65.1 Function Invocation Scheme

This function belongs to the BioSPI interface, and has the following native prototype:

```
BioAPI_RETURN BioAPI
BioSPI_VerifyMatch(
    BioAPI_HANDLE BSPHandle,
    BioAPI_FMR MaxFMRRequested,
    const BioAPI_INPUT_BIR *ProcessedBIR,
    const BioAPI_INPUT_BIR *ReferenceTemplate,
    BioAPI_BIR_HANDLE *AdaptedBIR,
    BioAPI_BOOL *Result,
    BioAPI_FMR *FMRAchieved,
    BioAPI_DATA *Payload);
```

and the following parameters:

Parameter	**Function invocation (conformance testing model for BSPs)**	**Bound activity invocation (conformance testing model for frameworks)**
`BSPHandle`	input, outbound	input, inbound
`MaxFMRRequested`	input, outbound	input, inbound
the members of the parameter group "Input BIR" (see 9.2.15) with their names prefixed by "`ProcessedBIR_`"	input, outbound	input, inbound
the members of the parameter group "Input BIR" with their names prefixed by "`ReferenceTemplate_`"	input, outbound	input, inbound

no_AdaptedBIR	input, outbound	input, inbound
no_Result	input, outbound	input, inbound
no_FMRAchieved	input, outbound	input, inbound
no_Payload	input, outbound	input, inbound
AdaptedBIR	output, inbound	output, outbound
Result	output, inbound	output, outbound
FMRAchieved	output, inbound	output, outbound
Payload	output, inbound	output, outbound
Return	return, inbound	output, outbound

The specification of the constraints on the parameters, function invocation input, function invocation output, and bound activity invocation input of the function **BioAPI_VerifyMatch** also applies to this standard BioAPI interface function.

9.65.2 Bound Activity Invocation Output

9.65.2.1 The outbound value of **AdaptedBIR** shall be processed as follows. If the native parameter with the same name is NULL, then the outbound value shall be ignored. Otherwise, the integer represented by the outbound value shall be written to the variable of type **BioAPI_BIR_HANDLE** pointed to by the native parameter **AdaptedBIR**.

9.65.2.2 The outbound value of **Result** shall be processed as follows. If the native parameter with the same name is NULL, then the outbound value shall be ignored. Otherwise, the boolean represented by the outbound value shall be written to the variable of type **BioAPI_BOOL** pointed to by the native parameter **Result**.

9.65.2.3 The outbound value of **FMRAchieved** shall be processed as follows. If the native parameter with the same name is NULL, then the outbound value shall be ignored. Otherwise, the integer represented by the outbound value shall be written to the variable of type **BioAPI_FMR** pointed to by the native parameter **FMRAchieved**.

9.65.2.4 The outbound value of **Payload** shall be processed as follows. If the native parameter with the same name is NULL, then the outbound value shall be ignored. Otherwise, the octet string represented by the outbound value shall be written to a memory block, whose address and size shall be written to the fields **Data** and **Length** (respectively) of the variable of type **BioAPI_DATA** pointed to by the native parameter **Payload**.

9.65.2.5 The integer represented by the outbound value of **return** shall be returned by the native function.

9.65.3 Default Output

If there is no activity bound to this standard BioAPI interface function, then:

a) the variable pointed to by the native parameter **AdaptedBIR** shall be set to zero;

b) the variable pointed to by the native parameter `Result` shall be set to zero;

c) the variable pointed to by the native parameter `FMRAchieved` shall be set to zero;

d) the fields `Data` and `Length` of the variable pointed to by the native parameter `Payload` shall be both set to zero; and

e) the native function shall return zero.

9.66 BioSPI_IdentifyMatch

9.66.1 Function Invocation Scheme

This function belongs to the BioSPI interface, and has the following native prototype:

```
BioAPI_RETURN BioAPI
BioSPI_IdentifyMatch(
    BioAPI_HANDLE BSPHandle,
    BioAPI_FMR MaxFMRRequested,
    const BioAPI_INPUT_BIR *ProcessedBIR,
    const BioAPI_IDENTIFY_POPULATION *Population,
    uint32_t TotalNumberOfTemplates,
    BioAPI_BOOL Binning,
    uint32_t MaxNumberOfResults,
    uint32_t *NumberOfResults,
    BioAPI_CANDIDATE_ARRAY  **Candidates,
    int32_t Timeout);
```

and the following parameters:

Parameter	**Function invocation (conformance testing model for BSPs)**	**Bound activity invocation (conformance testing model for frameworks)**
`BSPHandle`	input, outbound	input, inbound
`MaxFMRRequested`	input, outbound	input, inbound
the members of the parameter group "Input BIR" (see 9.2.15) with their names prefixed by "`ProcessedBIR_`"	input, outbound	input, inbound
the members of the parameter group "Identify population" (see 9.2.16)	input, outbound	input, inbound

`TotalNumberOfTemplates`	input, outbound	input, inbound
`Binning`	input, outbound	input, inbound
`MaxNumberOfResults`	input, outbound	input, inbound
`no_NumberOfResults`	input, outbound	input, inbound
`no_Candidates`	input, outbound	input, inbound
`Timeout`	input, outbound	input, inbound
`NumberOfResults`	output, inbound	output, outbound
the members of the parameter group "Candidate" (see 9.2.17) with their names prefixed by "**`Candidate_1_`**"	output, inbound	output, outbound
the members of the parameter group "Candidate" with their names prefixed by "**`Candidate_2_`**"	output, inbound	output, outbound
the members of the parameter group "Candidate" with their names prefixed by "**`Candidate_3_`**"	output, inbound	output, outbound
the members of the parameter group "Candidate" with their names prefixed by "**`Candidate_4_`**"	output, inbound	output, outbound
`Return`	return, inbound	output, outbound

The specification of the constraints on the parameters, function invocation input, function invocation output, and bound activity invocation input of the function **`BioAPI_IdentifyMatch`** also applies to this standard BioAPI interface function.

9.66.2 Bound Activity Invocation Output

9.66.2.1 If the native parameter `Candidates` is not NULL and the integer (say, *N*) represented by the outbound value of `NumberOfResults` is greater than 0, then a memory block of sufficient size to contain an array of *N* elements of type `BioAPI_CANDIDATE` shall be allocated and its address shall be written to the variable pointed to by the native parameter `Candidates`.

9.66.2.2 The outbound value of `NumberOfResults` shall be processed as follows. If the native parameter with the same name is NULL, then the outbound value shall be ignored. Otherwise, the integer represented by the outbound value shall be written to the variable of type `uint32_t` pointed to by the native parameter `NumberOfResults`.

9.66.2.3 The outbound value of the parameter group "Candidate" with the prefix `Candidate_`*X*`_` (with *X*=1, 2, 3, or 4) shall be processed as follows. If the native parameter `Candidates` is NULL or the integer represented by the outbound value of `NumberOfResults` is less than *X*, then the outbound value of the parameter group shall be ignored. Otherwise, the value of type `BioAPI_CANDIDATE` represented by the outbound value of the parameter group shall be written to the element at position *X* of the array of elements of type `BioAPI_CANDIDATE` pointed to by the variable pointed to by the native parameter `Candidates`.

9.66.2.4 The integer represented by the outbound value of `return` shall be returned by the native function.

9.66.3 Default Output

If there is no activity bound to this standard BioAPI interface function, then:

a) the variable pointed to by the native parameter `NumberOfResults` shall be set to zero;

b) the variable pointed to by the native parameter `Candidates` shall be set to NULL; and

c) the native function shall return zero.

9.67 BioSPI_Enroll

9.67.1 Function Invocation Scheme

This function belongs to the BioSPI interface, and has the following native prototype:

```
BioAPI_RETURN BioAPI
BioSPI_Enroll(
    BioAPI_HANDLE BSPHandle,
    BioAPI_BIR_PURPOSE Purpose,
    BioAPI_BIR_SUBTYPE Subtype,
    const BioAPI_BIR_BIOMETRIC_DATA_FORMAT *OutputFormat,
    const BioAPI_INPUT_BIR *ReferenceTemplate,
    BioAPI_BIR_HANDLE *NewTemplate,
    const BioAPI_DATA *Payload,
    int32_t Timeout,
    BioAPI_BIR_HANDLE *AuditData,
    BioAPI_UUID *TemplateUUID);
```

and the following parameters:

Parameter	**Function invocation (conformance testing model for BSPs)**	**Bound activity invocation (conformance testing model for frameworks)**
`BSPHandle`	input, outbound	input, inbound
`Purpose`	input, outbound	input, inbound
`Subtype`	input, outbound	input, inbound
`OutputFormatOwner`	input, outbound	input, inbound
`OutputFormatType`	input, outbound	input, inbound
the members of the parameter group "Input BIR" (see 9.2.15) with their names prefixed by "`ReferenceTemplate_`"	input, outbound	input, inbound
`no_NewTemplate`	input, outbound	input, inbound
`Payload`	input, outbound	input, inbound
`Timeout`	input, outbound	input, inbound
`no_AuditData`	input, outbound	input, inbound
`no_TemplateUUID`	input, outbound	input, inbound
`NewTemplate`	output, inbound	output, outbound
`AuditData`	output, inbound	output, outbound
`TemplateUUID`	output, inbound	output, outbound
`Return`	return, inbound	output, outbound

The specification of the constraints on the parameters, function invocation input, function invocation output, and bound activity invocation input of the function `BioAPI_Enroll` also applies to this standard BioAPI interface function.

9.67.2 Bound Activity Invocation Output

9.67.2.1 The outbound value of `NewTemplate` shall be processed as follows. If the native parameter with the same name is NULL, then the outbound value shall be ignored. Otherwise, the integer represented by the outbound value shall be written to the variable of type `BioAPI_BIR_HANDLE` pointed to by the native parameter `NewTemplate`.

9.67.2.2 The outbound value of `AuditData` shall be processed as follows. If the native parameter with the same name is NULL, then the outbound value shall be ignored. Otherwise, the integer represented by the outbound value shall be written to the variable of type `BioAPI_BIR_HANDLE` pointed to by the native parameter `AuditData`.

9.67.2.3 The outbound value of `TemplateUUID` shall be processed as follows. If the native parameter with the same name is NULL, then the outbound value shall be ignored. Otherwise, the UUID represented by the outbound value shall be written to the variable of type `BioAPI_UUID` pointed to by the native parameter `TemplateUUID`.

9.67.2.4 The integer represented by the outbound value of `return` shall be returned by the native function.

9.67.3 Default Output

If there is no activity bound to this standard BioAPI interface function, then:

a) the variable pointed to by the native parameter `NewTemplate` shall be set to zero;

b) the variable pointed to by the native parameter `AuditData` shall be set to zero;

c) the variable pointed to by the native parameter `TemplateUUID` shall be set to the UUID "`00000000-0000-0000-0000-000000000000`"; and

d) the native function shall return zero.

9.68 BioSPI_Verify

9.68.1 Function Invocation Scheme

This function belongs to the BioSPI interface, and has the following native prototype:

```
BioAPI_RETURN BioAPI
BioSPI_Verify(
    BioAPI_HANDLE BSPHandle,
    BioAPI_FMR MaxFMRRequested,
    const BioAPI_INPUT_BIR *ReferenceTemplate,
    BioAPI_BIR_SUBTYPE Subtype,
    BioAPI_BIR_HANDLE *AdaptedBIR,
    BioAPI_BOOL *Result,
    BioAPI_FMR *FMRAchieved,
    BioAPI_DATA *Payload,
    int32_t Timeout,
    BioAPI_BIR_HANDLE *AuditData);
```

and the following parameters:

Parameter	**Function invocation (conformance testing model for BSPs)**	**Bound activity invocation (conformance testing model for frameworks)**
`BSPHandle`	input, outbound	input, inbound
`MaxFMRRequested`	input, outbound	input, inbound
the members of the parameter group "Input BIR" (see 9.2.15) with their names prefixed by "`ReferenceTemplate_`"	input, outbound	input, inbound
`Subtype`	input, outbound	input, inbound
`no_AdaptedBIR`	input, outbound	input, inbound
`no_Result`	input, outbound	input, inbound
`no_FMRAchieved`	input, outbound	input, inbound
`no_Payload`	input, outbound	input, inbound
`Timeout`	input, outbound	input, inbound
`no_AuditData`	input, outbound	input, inbound
`AdaptedBIR`	output, inbound	output, outbound
`Result`	output, inbound	output, outbound
`FMRAchieved`	output, inbound	output, outbound
`Payload`	output, inbound	output, outbound
`AuditData`	output, inbound	output, outbound
`Return`	return, inbound	output, outbound

The specification of the constraints on the parameters, function invocation input, function invocation output, and bound activity invocation input of the function **`BioAPI_Verify`** also applies to this standard BioAPI interface function.

9.68.2 Bound Activity Invocation Output

9.68.2.1 The outbound value of **`AdaptedBIR`** shall be processed as follows. If the native parameter with the same name is NULL, then the outbound value shall be ignored. Otherwise, the integer represented by the outbound value shall be written to the variable of type **`BioAPI_BIR_HANDLE`** pointed to by the native parameter **`AdaptedBIR`**.

9.68.2.2 The outbound value of **`Result`** shall be processed as follows. If the native parameter with the same name is NULL, then the outbound value shall be ignored. Otherwise, the boolean represented by the outbound value shall be written to the variable of type **`BioAPI_BOOL`** pointed to by the native parameter **`Result`**.

9.68.2.3 The outbound value of **`FMRAchieved`** shall be processed as follows. If the native parameter with the same name is NULL, then the outbound value shall be ignored. Otherwise, the integer represented by the outbound value shall be written to the variable of type **`BioAPI_FMR`** pointed to by the native parameter **`FMRAchieved`**.

9.68.2.4 The outbound value of **`Payload`** shall be processed as follows. If the native parameter with the same name is NULL, then the outbound value shall be ignored. Otherwise, the octet string represented by the outbound value shall be written to a memory block, whose address and size shall be written to the fields **`Data`** and **`Length`** (respectively) of the variable of type **`BioAPI_DATA`** pointed to by the native parameter **`Payload`**.

9.68.2.5 The outbound value of **`AuditData`** shall be processed as follows. If the native parameter with the same name is NULL, then the outbound value shall be ignored. Otherwise, the integer represented by the outbound value shall be written to the variable of type **`BioAPI_BIR_HANDLE`** pointed to by the native parameter **`AuditData`**.

9.68.2.6 The integer represented by the outbound value of **`return`** shall be returned by the native function.

9.68.3 Default Output

If there is no activity bound to this standard BioAPI interface function, then:

a) the variable pointed to by the native parameter **`AdaptedBIR`** shall be set to zero;

b) the variable pointed to by the native parameter **`Result`** shall be set to zero;

c) the variable pointed to by the native parameter **`FMRAchieved`** shall be set to zero;

d) the fields **`Data`** and **`Length`** of the variable pointed to by the native parameter **`Payload`** shall be both set to zero;

e) the variable pointed to by the native parameter **`AuditData`** shall be set to zero; and

f) the native function shall return zero.

9.69 BioSPI_Identify

9.69.1 Function Invocation Scheme

This function belongs to the BioSPI interface, and has the following native prototype:

```
BioAPI_RETURN BioAPI
BioSPI_Identify(
    BioAPI_HANDLE BSPHandle,
    BioAPI_FMR MaxFMRRequested,
    BioAPI_BIR_SUBTYPE Subtype,
    const BioAPI_IDENTIFY_POPULATION *Population,
    uint32_t TotalNumberOfTemplates,
    BioAPI_BOOL Binning,
    uint32_t MaxNumberOfResults,
    uint32_t *NumberOfResults,
    BioAPI_CANDIDATE **Candidates,
    int32_t Timeout,
    BioAPI_BIR_HANDLE *AuditData);
```

and the following parameters:

Parameter	Function invocation (conformance testing model for BSPs)	Bound activity invocation (conformance testing model for frameworks)
`BSPHandle`	input, outbound	input, inbound
`MaxFMRRequested`	input, outbound	input, inbound
`Subtype`	input, outbound	input, inbound
the members of the parameter group "Identify population" (see 9.2.16)	input, outbound	input, inbound
`TotalNumberOfTemplates`	input, outbound	input, inbound
`Binning`	input, outbound	input, inbound
`MaxNumberOfResults`	input, outbound	input, inbound
`no_NumberOfResults`	input, outbound	input, inbound

no_Candidates	input, outbound	input, inbound
Timeout	input, outbound	input, inbound
no_AuditData	input, outbound	input, inbound
NumberOfResults	output, inbound	output, outbound
the members of the parameter group "Candidate" (see 9.2.17) with their names prefixed by "Candidate_1_"	output, inbound	output, outbound
the members of the parameter group "Candidate" with their names prefixed by "Candidate_2_"	output, inbound	output, outbound
the members of the parameter group "Candidate" with their names prefixed by "Candidate_3_"	output, inbound	output, outbound
the members of the parameter group "Candidate" with their names prefixed by "Candidate_4_"	output, inbound	output, outbound
AuditData	output, inbound	output, outbound
Return	return, inbound	output, outbound

The specification of the constraints on the parameters, function invocation input, function invocation output, and bound activity invocation input of the function BioAPI_Identify also applies to this standard BioAPI interface function.

9.69.2 Bound Activity Invocation Output

9.69.2.1 If the native parameter Candidates is not NULL and the integer (say, *N*) represented by the outbound value of NumberOfResults is greater than 0, then a memory block of sufficient size to contain an array of *N* elements of type BioAPI_CANDIDATE shall be allocated and its address shall be written to the variable pointed to by the native parameter Candidates.

9.69.2.2 The outbound value of NumberOfResults shall be processed as follows. If the native parameter with the same name is NULL, then the outbound value shall be ignored. Otherwise, the integer represented by the outbound value shall be written to the variable of type uint32_t pointed to by the native parameter NumberOfResults.

9.69.2.3 The outbound value of the parameter group "Candidate" with the prefix `Candidate_`*X*`_` (with *X*=1, 2, 3, or 4) shall be processed as follows. If the native parameter `Candidates` is NULL or the integer represented by the outbound value of `NumberOfResults` is less than *X*, then the outbound value of the parameter group shall be ignored. Otherwise, the value of type `BioAPI_CANDIDATE` represented by the outbound value of the parameter group shall be written to the element at position *X* of the array of elements of type `BioAPI_CANDIDATE` pointed to by the variable pointed to by the native parameter `Candidates`.

9.69.2.4 The outbound value of `AuditData` shall be processed as follows. If the native parameter with the same name is NULL, then the outbound value shall be ignored. Otherwise, the integer represented by the outbound value shall be written to the variable of type `BioAPI_BIR_HANDLE` pointed to by the native parameter `AuditData`.

9.69.2.5 The integer represented by the outbound value of `return` shall be returned by the native function.

9.69.3 Default Output

If there is no activity bound to this standard BioAPI interface function, then:

a) the variable pointed to by the native parameter `NumberOfResults` shall be set to zero;

b) the variable pointed to by the native parameter `Candidates` shall be set to NULL;

c) the variable pointed to by the native parameter `AuditData` shall be set to zero; and

d) the native function shall return zero.

9.70 BioSPI_Import

9.70.1 Function Invocation Scheme

This function belongs to the BioSPI interface, and has the following native prototype:

```
BioAPI_RETURN BioAPI
BioSPI_Import(
    BioAPI_HANDLE BSPHandle,
    const BioAPI_DATA *InputData,
    const BioAPI_BIR_BIOMETRIC_DATA_FORMAT *InputFormat,
    const BioAPI_BIR_BIOMETRIC_DATA_FORMAT *OutputFormat,
    BioAPI_BIR_PURPOSE Purpose,
    BioAPI_BIR_HANDLE *ConstructedBIR);
```

and the following parameters:

Parameter	Function invocation (conformance testing model for BSPs)	Bound activity invocation (conformance testing model for frameworks)
`BSPHandle`	input, outbound	input, inbound
`InputData`	input, outbound	input, inbound

`InputFormatOwner`	input, outbound	input, inbound
`InputFormatType`	input, outbound	input, inbound
`OutputFormatOwner`	input, outbound	input, inbound
`OutputFormatType`	input, outbound	input, inbound
`Purpose`	input, outbound	input, inbound
`no_ConstructedBIR`	input, outbound	input, inbound
`ConstructedBIR`	output, inbound	output, outbound
`return`	return, inbound	output, outbound

The specification of the constraints on the parameters, function invocation input, function invocation output, and bound activity invocation input of the function **`BioAPI_Import`** also applies to this standard BioAPI interface function.

9.70.2 Bound Activity Invocation Output

9.70.2.1 The outbound value of **`ConstructedBIR`** shall be processed as follows. If the native parameter with the same name is NULL, then the outbound value shall be ignored. Otherwise, the integer represented by the outbound value shall be written to the variable of type **`BioAPI_BIR_HANDLE`** pointed to by the native parameter **`ConstructedBIR`**.

9.70.2.2 The integer represented by the outbound value of **`return`** shall be returned by the native function.

9.70.3 Default Output

If there is no activity bound to this standard BioAPI interface function, then:

a) the variable pointed to by the native parameter **`ConstructedBIR`** shall be set to zero; and

b) the native function shall return zero.

9.71 BioSPI_PresetIdentifyPopulation

9.71.1 Function Invocation Scheme

This function belongs to the BioSPI interface, and has the following native prototype:

BioAPI_RETURN BioAPI

BioSPI_PresetIdentifyPopulation(

BioAPI_HANDLE BSPHandle,

const BioAPI_IDENTIFY_POPULATION *Population);

and the following parameters:

Parameter	**Function invocation (conformance testing model for BSPs)**	**Bound activity invocation (conformance testing model for frameworks)**
`BSPHandle`	input, outbound	input, inbound
members of the parameter group "Identify population" (see 9.2.16)	input, outbound	input, inbound
`return`	return, inbound	output, outbound

The specification of the constraints on the parameters, function invocation input, function invocation output, and bound activity invocation input of the function `BioAPI_PresetIdentifyPopulation` also applies to this standard BioAPI interface function.

9.71.2 Bound Activity Invocation Output

The integer represented by the outbound value of `return` shall be returned by the native function.

9.71.3 Default Output

If there is no activity bound to this standard BioAPI interface function, the native function shall return zero.

9.72 BioSPI_DbOpen

9.72.1 Function Invocation Scheme

This function belongs to the BioSPI interface, and has the following native prototype:

```
BioAPI_RETURN BioAPI
BioSPI_DbOpen(
    BioAPI_HANDLE BSPHandle,
    const BioAPI_UUID *DbUuid,
    BioAPI_DB_ACCESS_TYPE AccessRequest,
    BioAPI_DB_HANDLE *DbHandle,
    BioAPI_DB_MARKER_HANDLE *MarkerHandle);
```

and the following parameters:

Parameter	Function invocation (conformance testing model for BSPs)	Bound activity invocation (conformance testing model for frameworks)
`BSPHandle`	input, outbound	input, inbound
`DbUuid`	input, outbound	input, inbound
the members of the parameter group "Access type" (see 9.2.19)	input, outbound	input, inbound
`no_DbHandle`	input, outbound	input, inbound
`no_MarkerHandle`	input, outbound	input, inbound
`DbHandle`	output, inbound	output, outbound
`MarkerHandle`	output, inbound	output, outbound
`return`	return, inbound	output, outbound

The specification of the constraints on the parameters, function invocation input, function invocation output, and bound activity invocation input of the function `BioAPI_DbOpen` also applies to this standard BioAPI interface function.

9.72.2 Bound Activity Invocation Output

9.72.2.1 The outbound value of `DbHandle` shall be processed as follows. If the native parameter with the same name is NULL, then the outbound value shall be ignored. Otherwise, the integer represented by the outbound value shall be written to the variable of type `BioAPI_DB_HANDLE` pointed to by the native parameter `DbHandle`.

9.72.2.2 The outbound value of `MarkerHandle` shall be processed as follows. If the native parameter with the same name is NULL, then the outbound value shall be ignored. Otherwise, the integer represented by the outbound value shall be written to the variable of type `BioAPI_DB_MARKER_HANDLE` pointed to by the native parameter `MarkerHandle`.

9.72.2.3 The integer represented by the outbound value of `return` shall be returned by the native function.

9.72.3 Default Output

If there is no activity bound to this standard BioAPI interface function, then:

a) the variable pointed to by the native parameter `DbHandle` shall be set to zero;

b) the variable pointed to by the native parameter `MarkerHandle` shall be set to zero; and

c) the native function shall return zero.

9.73 BioSPI_DbClose

9.73.1 Function Invocation Scheme

This function belongs to the BioSPI interface, and has the following native prototype:

```
BioAPI_RETURN BioAPI
BioSPI_DbClose(
    BioAPI_HANDLE BSPHandle,
    BioAPI_DB_HANDLE DbHandle);
```

and the following parameters:

Parameter	Function invocation (conformance testing model for BSPs)	Bound activity invocation (conformance testing model for frameworks)
`BSPHandle`	input, outbound	input, inbound
`DbHandle`	input, outbound	input, inbound
`return`	return, inbound	output, outbound

The specification of the constraints on the parameters, function invocation input, function invocation output, and bound activity invocation input of the function `BioAPI_DbClose` also applies to this standard BioAPI interface function.

9.73.2 Bound Activity Invocation Output

The integer represented by the outbound value of `return` shall be returned by the native function.

9.73.3 Default Output

If there is no activity bound to this standard BioAPI interface function, the native function shall return zero.

9.74 BioSPI_DbCreate

9.74.1 Function Invocation Scheme

This function belongs to the BioSPI interface, and has the following native prototype:

BioAPI_RETURN BioAPI

BioSPI_DbCreate(

BioAPI_HANDLE BSPHandle,

const BioAPI_UUID *DbUuid,

uint32_t NumberOfRecords,

BioAPI_DB_ACCESS_TYPE AccessRequest,

BioAPI_DB_HANDLE *DbHandle);

and the following parameters:

Parameter	**Function invocation (conformance testing model for BSPs)**	**Bound activity invocation (conformance testing model for frameworks)**
`BSPHandle`	input, outbound	input, inbound
`DbUuid`	input, outbound	input, inbound
`NumberOfRecords`	input, outbound	input, inbound
the members of the parameter group "Access type" (see 9.2.19)	input, outbound	input, inbound
`no_DbHandle`	input, outbound	input, inbound
`DbHandle`	output, inbound	output, outbound
`Return`	return, inbound	output, outbound

The specification of the constraints on the parameters, function invocation input, function invocation output, and bound activity invocation input of the function `BioAPI_DbCreate` also applies to this standard BioAPI interface function.

9.74.2 Bound Activity Invocation Output

9.74.2.1 The outbound value of `DbHandle` shall be processed as follows. If the native parameter with the same name is NULL, then the outbound value shall be ignored. Otherwise, the integer represented by the outbound value shall be written to the variable of type `BioAPI_DB_HANDLE` pointed to by the native parameter `DbHandle`.

9.74.2.2 The integer represented by the outbound value of `return` shall be returned by the native function.

9.74.3 Default Output

If there is no activity bound to this standard BioAPI interface function, then:

a) the variable pointed to by the native parameter `DbHandle` shall be set to zero; and

b) the native function shall return zero.

9.75 BioSPI_DbDelete

9.75.1 Function Invocation Scheme

This function belongs to the BioSPI interface, and has the following native prototype:

```
BioAPI_RETURN BioAPI
BioSPI_DbDelete(
    BioAPI_HANDLE BSPHandle,
    const BioAPI_UUID *DbUuid);
```

and the following parameters:

Parameter	Function invocation (conformance testing model for BSPs)	Bound activity invocation (conformance testing model for frameworks)
`BSPHandle`	input, outbound	input, inbound
`DbUuid`	input, outbound	input, inbound
`return`	return, inbound	output, outbound

The specification of the constraints on the parameters, function invocation input, function invocation output, and bound activity invocation input of the function `BioAPI_DbDelete` also applies to this standard BioAPI interface function.

9.75.2 Bound Activity Invocation Output

The integer represented by the outbound value of `return` shall be returned by the native function.

9.75.3 Default Output

If there is no activity bound to this standard BioAPI interface function, the native function shall return zero.

9.76 BioSPI_DbSetMarker

9.76.1 Function Invocation Scheme

This function belongs to the BioSPI interface, and has the following native prototype:

BioAPI_RETURN BioAPI

BioSPI_DbSetMarker(

BioAPI_HANDLE BSPHandle,

BioAPI_DB_HANDLE DbHandle,

const BioAPI_UUID *KeyValue,

BioAPI_DB_MARKER_HANDLE MarkerHandle);

and the following parameters:

Parameter	**Function invocation (conformance testing model for BSPs)**	**Bound activity invocation (conformance testing model for frameworks)**
`BSPHandle`	input, outbound	input, inbound
`DbHandle`	input, outbound	input, inbound
`KeyValue`	input, outbound	input, inbound
`MarkerHandle`	input, outbound	input, inbound
`return`	return, inbound	output, outbound

The specification of the constraints on the parameters, function invocation input, function invocation output, and bound activity invocation input of the function `BioAPI_DbSetMarker` also applies to this standard BioAPI interface function.

9.76.2 Bound Activity Invocation Output

The integer represented by the outbound value of `return` shall be returned by the native function.

9.76.3 Default Output

If there is no activity bound to this standard BioAPI interface function, then the native function shall return zero.

9.77 BioSPI_DbFreeMarker

9.77.1 Function Invocation Scheme

This function belongs to the BioSPI interface, and has the following native prototype:

BioAPI_RETURN BioAPI

BioSPI_DbFreeMarker(

BioAPI_HANDLE BSPHandle,

BioAPI_DB_MARKER_HANDLE MarkerHandle);

and the following parameters:

Parameter	Function invocation (conformance testing model for BSPs)	Bound activity invocation (conformance testing model for frameworks)
`BSPHandle`	input, outbound	input, inbound
`MarkerHandle`	input, outbound	input, inbound
`return`	return, inbound	output, outbound

The specification of the constraints on the parameters, function invocation input, function invocation output, and bound activity invocation input of the function `BioAPI_DbFreeMarker` also applies to this standard BioAPI interface function.

9.77.2 Bound Activity Invocation Output

The integer represented by the outbound value of `return` shall be returned by the native function.

9.77.3 Default Output

If there is no activity bound to this standard BioAPI interface function, the native function shall return zero.

9.78 BioSPI_DbStoreBIR

9.78.1 Function Invocation Scheme

This function belongs to the BioSPI interface, and has the following native prototype:

```
BioAPI_RETURN BioAPI
BioSPI_DbStoreBIR(
    BioAPI_HANDLE BSPHandle,
    const BioAPI_INPUT_BIR *BIRToStore,
    BioAPI_DB_HANDLE DbHandle,
    BioAPI_UUID *BirUuid);
```

and the following parameters:

Parameter	**Function invocation (conformance testing model for BSPs)**	**Bound activity invocation (conformance testing model for frameworks)**
`BSPHandle`	input, outbound	input, inbound
the members of the parameter group "Input BIR" (see 9.2.15) with their names prefixed by "`BIRToStore_`"	input, outbound	input, inbound
`DBHandle`	input, outbound	input, inbound
`no_BirUuid`	input, outbound	input, inbound
`BirUuid`	output, inbound	output, outbound
`return`	return, inbound	output, outbound

The specification of the constraints on the parameters, function invocation input, function invocation output, and bound activity invocation input of the function `BioAPI_DbStoreBIR` also applies to this standard BioAPI interface function.

9.78.2 Bound Activity Invocation Output

9.78.2.1 The outbound value of `BirUuid` shall be processed as follows. If the native parameter with the same name is NULL, then the outbound value shall be ignored. Otherwise, the UUID represented by the outbound value shall be written to the variable of type `BioAPI_UUID` pointed to by the native parameter `BirUuid`.

9.78.2.2 The integer represented by the outbound value of `return` shall be returned by the native function.

9.78.3 Default Output

If there is no activity bound to this standard BioAPI interface function, then:

a) the variable pointed to by the native parameter `BirUuid` shall be set to the UUID "00000000-0000-0000-0000-000000000000"; and

b) the native function shall return zero.

9.79 BioSPI_DbGetBIR

9.79.1 Function Invocation Scheme

This function belongs to the BioSPI interface, and has the following native prototype:

BioAPI_RETURN BioAPI

BioSPI_DbGetBIR(

BioAPI_HANDLE BSPHandle,

BioAPI_DB_HANDLE DbHandle,

const BioAPI_UUID *KeyValue,

BioAPI_BIR_HANDLE *RetrievedBIR,

BioAPI_DB_MARKER_HANDLE *MarkerHandle);

and the following parameters:

Parameter	**Function invocation (conformance testing model for BSPs)**	**Bound activity invocation (conformance testing model for frameworks)**
`BSPHandle`	input, outbound	input, inbound
`DbHandle`	input, outbound	input, inbound
`KeyValue`	input, outbound	input, inbound
`no_RetrievedBIR`	input, outbound	input, inbound
`no_MarkerHandle`	input, outbound	input, inbound

`RetrievedBIR`	output, inbound	output, outbound
`MarkerHandle`	output, inbound	output, outbound
`return`	return, inbound	output, outbound

The specification of the constraints on the parameters, function invocation input, function invocation output, and bound activity invocation input of the function `BioAPI_DbGetBIR` also applies to this standard BioAPI interface function.

9.79.2 Bound Activity Invocation Output

9.79.2.1 The outbound value of `RetrievedBIR` shall be processed as follows. If the native parameter with the same name is NULL, then the outbound value shall be ignored. Otherwise, the integer represented by the outbound value shall be written to the variable of type `BioAPI_BIR_HANDLE` pointed to by the native parameter `RetrievedBIR`.

9.79.2.2 The outbound value of `MarkerHandle` shall be processed as follows. If the native parameter with the same name is NULL, then the outbound value shall be ignored. Otherwise, the integer represented by the outbound value shall be written to the variable of type `BioAPI_DB_MARKER_HANDLE` pointed to by the native parameter `MarkerHandle`.

9.79.2.3 The integer represented by the outbound value of `return` shall be returned by the native function.

9.79.3 Default Output

If there is no activity bound to this standard BioAPI interface function, then:

a) the variable pointed to by the native parameter `RetrievedBIR` shall be set to zero;

b) the variable pointed to by the native parameter `MarkerHandle` shall be set to zero; and

c) the native function shall return zero.

9.80 BioSPI_DbGetNextBIR

9.80.1 Function Invocation Scheme

This function belongs to the BioSPI interface, and has the following native prototype:

```
BioAPI_RETURN BioAPI
BioSPI_DbGetNextBIR(
    BioAPI_HANDLE BSPHandle,
    BioAPI_DB_HANDLE DbHandle,
    BioAPI_DB_MARKER_HANDLE MarkerHandle,
    BioAPI_BIR_HANDLE *RetrievedBIR,
    BioAPI_UUID *BirUuid);
```

and the following parameters:

Parameter	Function invocation (conformance testing model for BSPs)	Bound activity invocation (conformance testing model for frameworks)
BSPHandle	input, outbound	input, inbound
DbHandle	input, outbound	input, inbound
MarkerHandle	input, outbound	input, inbound
no_RetrievedBIR	input, outbound	input, inbound
no_BirUuid	input, outbound	input, inbound
RetrievedBIR	output, inbound	output, outbound
BirUuid	output, inbound	output, outbound
return	return, inbound	output, outbound

The specification of the constraints on the parameters, function invocation input, function invocation output, and bound activity invocation input of the function **BioAPI_DbGetNextBIR** also applies to this standard BioAPI interface function.

9.80.2 Bound Activity Invocation Output

9.80.2.1 The outbound value of **RetrievedBIR** shall be processed as follows. If the native parameter with the same name is NULL, then the outbound value shall be ignored. Otherwise, the integer represented by the outbound value shall be written to the variable of type **BioAPI_BIR_HANDLE** pointed to by the native parameter **RetrievedBIR**.

9.80.2.2 The outbound value of **BirUuid** shall be processed as follows. If the native parameter with the same name is NULL, then the outbound value shall be ignored. Otherwise, the UUID represented by the outbound value shall be written to the variable of type **BioAPI_UUID** pointed to by the native parameter **BirUuid**.

9.80.2.3 The integer represented by the outbound value of **return** shall be returned by the native function.

9.80.3 Default Output

If there is no activity bound to this standard BioAPI interface function, then:

a) the variable pointed to by the native parameter **RetrievedBIR** shall be set to zero;

b) the variable pointed to by the native parameter **BirUuid** shall be set to the UUID "00000000-0000-0000-0000-000000000000"; and

c) the native function shall return zero.

9.81 BioSPI_DbDeleteBIR

9.81.1 Function Invocation Scheme

This function belongs to the BioSPI interface, and has the following native prototype:

BioAPI_RETURN BioAPI

BioSPI_DbDeleteBIR(

BioAPI_HANDLE BSPHandle,

BioAPI_DB_HANDLE DbHandle,

const BioAPI_UUID *KeyValue);

and the following parameters:

Parameter	**Function invocation (conformance testing model for BSPs)**	**Bound activity invocation (conformance testing model for frameworks)**
`BSPHandle`	input, outbound	input, inbound
`DbHandle`	input, outbound	input, inbound
`KeyValue`	input, outbound	input, inbound
`return`	return, inbound	output, outbound

The specification of the constraints on the parameters, function invocation input, function invocation output, and bound activity invocation input of the function `BioAPI_DbDeleteBIR` also applies to this standard BioAPI interface function.

9.81.2 Bound Activity Invocation Output

The integer represented by the outbound value of `return` shall be returned by the native function.

9.81.3 Default Output

If there is no activity bound to this standard BioAPI interface function, the native function shall return zero.

9.82 BioSPI_SetPowerMode

9.82.1 Function Invocation Scheme

This function belongs to the BioSPI interface, and has the following native prototype:

BioAPI_RETURN BioAPI

BioSPI_SetPowerMode(

BioAPI_HANDLE BSPHandle,

BioAPI_UNIT_ID UnitID,

BioAPI_POWER_MODE PowerMode);

and the following parameters:

Parameter	**Function invocation (conformance testing model for BSPs)**	**Bound activity invocation (conformance testing model for frameworks)**
`BSPHandle`	input, outbound	input, inbound
`UnitID`	input, outbound	input, inbound
`PowerMode`	input, outbound	input, inbound
`return`	return, inbound	output, outbound

The specification of the constraints on the parameters, function invocation input, function invocation output, and bound activity invocation input of the function `BioAPI_SetPowerMode` also applies to this standard BioAPI interface function.

9.82.2 Bound Activity Invocation Output

The integer represented by the outbound value of `return` shall be returned by the native function.

9.82.3 Default Output

If there is no activity bound to this standard BioAPI interface function, the native function shall return zero.

9.83 BioSPI_SetIndicatorStatus

9.83.1 Function Invocation Scheme

This function belongs to the BioSPI interface, and has the following native prototype:

BioAPI_RETURN BioAPI

BioSPI_SetIndicatorStatus(

BioAPI_HANDLE BSPHandle,

BioAPI_UNIT_ID UnitID,

BioAPI_INDICATOR_STATUS IndicatorStatus);

and the following parameters:

Parameter	**Function invocation (conformance testing model for BSPs)**	**Bound activity invocation (conformance testing model for frameworks)**
`BSPHandle`	input, outbound	input, inbound
`UnitID`	input, outbound	input, inbound
`IndicatorStatus`	input, outbound	input, inbound
`return`	return, inbound	output, outbound

The specification of the constraints on the parameters, function invocation input, function invocation output, and bound activity invocation input of the function `BioAPI_SetIndicatorStatus` also applies to this standard BioAPI interface function.

9.83.2 Bound Activity Invocation Output

The integer represented by the outbound value of `return` shall be returned by the native function.

9.83.3 Default Output

If there is no activity bound to this standard BioAPI interface function, the native function shall return zero.

9.84 BioSPI_GetIndicatorStatus

9.84.1 Function Invocation Scheme

This function belongs to the BioSPI interface, and has the following native prototype:

BioAPI_RETURN BioAPI

BioSPI_GetIndicatorStatus(

BioAPI_HANDLE BSPHandle,

BioAPI_UNIT_ID UnitID,

BioAPI_INDICATOR_STATUS *IndicatorStatus);

and the following parameters:

Parameter	**Function invocation (conformance testing model for BSPs)**	**Bound activity invocation (conformance testing model for frameworks)**
`BSPHandle`	input, outbound	input, inbound
`UnitID`	input, outbound	input, inbound
`no_IndicatorStatus`	input, outbound	input, inbound
`IndicatorStatus`	output, inbound	output, outbound
`return`	return, inbound	output, outbound

The specification of the constraints on the parameters, function invocation input, function invocation output, and bound activity invocation input of the function `BioAPI_GetIndicatorStatus` also applies to this standard BioAPI interface function.

9.84.2 Bound Activity Invocation Output

9.84.2.1 The outbound value of `IndicatorStatus` shall be processed as follows. If the native parameter with the same name is NULL, then the outbound value shall be ignored. Otherwise, the integer represented by the outbound value shall be written to the variable of type `BioAPI_INDICATOR_STATUS` pointed to by the native parameter `IndicatorStatus`.

9.84.2.2 The integer represented by the outbound value of `return` shall be returned by the native function.

9.84.3 Default Output

If there is no activity bound to this standard BioAPI interface function, then:

a) the variable pointed to by the native parameter `IndicatorStatus` shall be set to zero;

b) the native function shall return zero.

9.85 BioSPI_CalibrateSensor

9.85.1 Function Invocation Scheme

This function belongs to the BioSPI interface, and has the following native prototype:

BioAPI_RETURN BioAPI

BioSPI_CalibrateSensor(

BioAPI_HANDLE BSPHandle,

int32_t Timeout);

and the following parameters:

Parameter	**Function invocation (conformance testing model for BSPs)**	**Bound activity invocation (conformance testing model for frameworks)**
`BSPHandle`	input, outbound	input, inbound
`Timeout`	input, outbound	input, inbound
`return`	return, inbound	output, outbound

The specification of the constraints on the parameters, function invocation input, function invocation output, and bound activity invocation input of the function `BioAPI_CalibrateSensor` also applies to this standard BioAPI interface function.

9.85.2 Bound Activity Invocation Output

The integer represented by the outbound value of `return` shall be returned by the native function.

9.85.3 Default Output

If there is no activity bound to this standard BioAPI interface function, the native function shall return zero.

9.86 BioSPI_Cancel

9.86.1 Function Invocation Scheme

This function belongs to the BioSPI interface, and has the following native prototype:

BioAPI_RETURN BioAPI

BioSPI_Cancel(

BioAPI_HANDLE BSPHandle);

and the following parameters:

Parameter	Function invocation (conformance testing model for BSPs)	Bound activity invocation (conformance testing model for frameworks)
`BSPHandle`	input, outbound	input, inbound
`return`	return, inbound	output, outbound

The specification of the constraints on the parameters, function invocation input, function invocation output, and bound activity invocation input of the function `BioAPI_Cancel` also applies to this standard BioAPI interface function.

9.86.2 Bound Activity Invocation Output

The integer represented by the outbound value of `return` shall be returned by the native function.

9.86.3 Default Output

If there is no activity bound to this standard BioAPI interface function, the native function shall return zero.

9.87 BioSPI_Free

9.87.1 Function Invocation Scheme

This function belongs to the BioSPI interface, and has the following native prototype:

```
BioAPI_RETURN BioAPI
BioSPI_Free(
    void* Ptr);
```

and the following parameters:

Parameter	Function invocation (conformance testing model for BSPs)	Bound activity invocation (conformance testing model for frameworks)
`Ptr`	input, outbound	input, inbound
`return`	return, inbound	output, outbound

The specification of the constraints on the parameters, function invocation input, function invocation output, and bound activity invocation input of the function `BioAPI_Free` also applies to this standard BioAPI interface function.

9.87.2 Bound Activity Invocation Output

The integer represented by the outbound value of `return` shall be returned by the native function.

9.87.3 Default Output

If there is no activity bound to this standard BioAPI interface function, the native function shall return zero.

9.88 BioAPI_EventHandler

9.88.1 Function Invocation Scheme

This function belongs to the application callback interface, and has the following native prototype:

```
typedef BioAPI_RETURN (*BioAPI_EventHandler)(
    const BioAPI_UUID *BSPUuid,
    BioAPI_UNIT_ID UnitID,
    void* AppNotifyCallbackCtx,
    const BioAPI_UNIT_SCHEMA *UnitSchema,
    BioAPI_EVENT EventType);
```

and the following parameters:

Parameter	Bound activity invocation (conformance testing model for applications)	Bound activity invocation (conformance testing model for frameworks)
`BSPUuid`	input, inbound	input, inbound
`UnitID`	input, inbound	input, inbound
`AppNotifyCallbackCtx`	input, inbound	input, inbound
the members of the parameter group "Unit schema" (see 9.2.12) with their names prefixed by "`UnitSchema_`"	input, inbound	input, inbound
`EventType`	input, inbound	input, inbound
`Return`	input, inbound	output, outbound

9.88.2 Constraints on the Parameters

9.88.2.1 An outbound value of `BSPUuid` shall be a valid representation of a UUID (see 7.6). An inbound value shall be the canonical representation of a UUID (see 7.6.3).

9.88.2.2 An outbound value of `UnitID`, `AppNotifyCallbackCtx`, and `EventType` shall be a valid representation of an integer (see 7.4) in the range 0 to 4294967295. An inbound value shall be the canonical representation of an integer (see 7.4.3) in the same range.

9.88.2.3 An outbound value of the parameter group "Unit schema" shall be a valid representation of a value of the native type `BioAPI_UNIT_SCHEMA`. An inbound value shall be either the canonical representation of a value of that type, or a set of empty strings.

9.88.2.4 An outbound value of `return` shall be a valid representation of an integer in the range 0 to 4294967295. An inbound value shall be the canonical representation of an integer in the same range.

9.88.3 Bound Activity Invocation Input

9.88.3.1 The inbound value of `BSPUuid` shall be the canonical representation of the UUID in the variable of type `BioAPI_UUID` pointed to by the native parameter `BSPUuid`.

9.88.3.2 The inbound value of `UnitID`, `AppNotifyCallbackCtx` and `EventType` shall be the canonical representation of the integer in the native parameter with the same name.

9.88.3.3 The inbound value of the parameter group "Unit schema" shall be the canonical representation (see 9.2.12.5) of the value of type `BioAPI_UNIT_SCHEMA` in the variable pointed to by the native parameter `UnitSchema`.

9.88.3.4 The inbound value of `return` shall be the canonical representation of the integer returned by the native function.

9.88.4 Bound Activity Invocation Output

The integer represented by the outbound value of `return` shall be returned by the native function.

9.88.5 Default Output

If there is no activity bound to this standard BioAPI interface function, the native function shall return zero.

9.89 BioAPI_GUI_STATE_CALLBACK

9.89.1 Function Invocation Scheme

This function belongs to the application callback interface, and has the following native prototype:

```
typedef BioAPI_RETURN (BioAPI *BioAPI_GUI_STATE_CALLBACK)(

    void *GuiStateCallbackCtx,
    BioAPI_GUI_STATE GuiState,
    BioAPI_GUI_RESPONSE *Response,
    BioAPI_GUI_MESSAGE Message,
    BioAPI_GUI_PROGRESS Progress,
    const BioAPI_GUI_BITMAP *SampleBuffer);
```

and the following parameters:

Parameter	Bound activity invocation (conformance testing model for applications)	Bound activity invocation (conformance testing model for frameworks)
`GuiStateCallbackCtx`	input, inbound	input, inbound
the members of the parameter group "GUI state" (see 9.2.18)	input, inbound	input, inbound
`no_Response`	input, inbound	input, inbound
`Message`	input, inbound	input, inbound
`Progress`	input, inbound	input, inbound
`BitmapWidth`	input, inbound	input, inbound
`BitmapHeight`	input, inbound	input, inbound
`Bitmap`	input, inbound	input, inbound
`Response`	input, inbound	output, outbound
`Return`	input, inbound	output, outbound

9.89.2 Constraints on the Parameters

9.89.2.1 An outbound value of **`GuiStateCallbackCtx`** and **`Message`** shall be a valid representation of an integer (see 7.4) in the range 0 to 4294967295. An inbound value shall be the canonical representation of an integer (see 7.4.3) in the same range.

9.89.2.2 An outbound value of the parameter group "GUI state" shall be a valid representation of a value of the native type **`BioAPI_GUI_STATE`**. An inbound value shall be the canonical representation of a value of that type.

9.89.2.3 An outbound value of **`no_Response`** shall be either a valid representation of a boolean (see 7.5), or an empty string. An inbound value shall be the canonical representation of a boolean (see 7.5.2).

9.89.2.4 An outbound value of **`Progress`** shall be a valid representation of an integer in the range 0 to 255. An inbound value shall be the canonical representation of an integer in the same range.

9.89.2.5 An outbound value of `BitmapWidth` and `BitmapHeight` shall be either a valid representation of an integer (see 7.4) in the range 0 to 4294967295, or an empty string. An inbound value shall be the canonical representation of an integer (see 7.4.3) in the same range.

9.89.2.6 An outbound value of `Bitmap` shall be a valid representation of an octet string (see 7.7). An inbound value shall be the canonical representation of an octet string (see 7.7.2).

9.89.2.7 An outbound value of `Response` shall be either a valid representation of an integer in the range 0 to 255, or an empty string. An inbound value shall be either the canonical representation of an integer in the same range, or an empty string.

9.89.2.8 An outbound value of `return` shall be a valid representation of an integer in the range 0 to 4294967295. An inbound value shall be the canonical representation of an integer in the same range.

9.89.3 Bound Activity Invocation Input

9.89.3.1 The inbound value of `GuiStateCallbackCtx`, `Message`, and `Progress` shall be the canonical representation of the integer in the native parameter with the same name.

9.89.3.2 The inbound value of the parameter group "GUI state" shall be the canonical representation (see 9.2.18.5) of the integer in the native parameter `GuiState`.

9.89.3.3 The inbound value of `no_Response` shall be "`true`" if the native parameter `Response` is NULL, otherwise it shall be "`false`".

9.89.3.4 The inbound value of `Bitmap` shall be determined as follows. If the native parameter `SampleBuffer` is NULL, then the inbound value shall be an empty string. Otherwise, the inbound value shall be the canonical representation (see 7.7.2) of the octet string in the memory block whose address and length are in the fields `Data` and `Length` (respectively) of the variable of type `BioAPI_DATA` pointed to by the field `Bitmap` of the variable of type `BioAPI_GUI_BITMAP` pointed to by the native parameter `SampleBuffer`.

9.89.3.5 The inbound value of `BitmapWidth` and `BitmapHeight` shall be determined as follows. If the native parameter `SampleBuffer` is NULL, then the inbound value shall be "0". Otherwise, the inbound value shall be the canonical representation (see 7.4.3) of the integer in the fields `Width` and `Height` (respectively) of the variable of type `BioAPI_GUI_BITMAP` pointed to by the native parameter `SampleBuffer`.

9.89.3.6 The inbound value of `Response` shall be determined as follows. If the native parameter with the same name is NULL, then the inbound value shall be an empty string. Otherwise, the inbound value shall be the canonical representation of the integer in the variable of type `BioAPI_GUI_RESPONSE` pointed to by the native parameter `Response`.

9.89.3.7 The inbound value of `return` shall be the canonical representation of the integer returned by the native function..

9.89.4 Bound Activity Invocation Output

9.89.4.1 The outbound value of `Response` shall be processed as follows. If the native parameter with the same name is NULL, then the outbound value shall be ignored. Otherwise, the integer represented by the outbound value shall be written to the variable of type `BioAPI_GUI_RESPONSE` pointed to by the native parameter `Response`.

9.89.4.2 The integer represented by the outbound value of `return` shall be returned by the native function.

9.89.5 Default Output

If there is no activity bound to this standard BioAPI interface function, then:

a) the variable pointed to by the native parameter `Response` shall be set to zero; and

b) the native function shall return zero.

9.90 BioAPI_GUI_STREAMING_CALLBACK

9.90.1 Function Invocation Scheme

This function belongs to the application callback interface, and has the following native prototype:

```
typedef BioAPI_RETURN (BioAPI *BioAPI_GUI_STREAMING_CALLBACK)(
    void *GuiStreamingCallbackCtx,
    const BioAPI_GUI_BITMAP *Bitmap);
```

and the following parameters:

Parameter	Bound activity invocation (conformance testing model for applications)	Bound activity invocation (conformance testing model for frameworks)
`GuiStreamingCallbackCtx`	input, inbound	input, inbound
`BitmapWidth`	input, inbound	input, inbound
`BitmapHeight`	input, inbound	input, inbound
`Bitmap`	input, inbound	input, inbound
`return`	input, inbound	input, inbound

9.90.2 Constraints on the Parameters

9.90.2.1 An outbound value of `GuiStreamingCallbackCtx` shall be a valid representation of an integer (see 7.4) in the range 0 to 4294967295. An inbound value shall be the canonical representation of an integer (see 7.4.3) in the same range.

9.90.2.2 An outbound value of `BitmapWidth` and `BitmapHeight` shall be either a valid representation of an integer (see 7.4) in the range 0 to 4294967295, or an empty string. An inbound value shall be the canonical representation of an integer in the same range.

9.90.2.3 An outbound value of `Bitmap` shall be a valid representation of an octet string (see 7.7). An inbound value shall be the canonical representation of an octet string (see 7.7.2).

9.90.2.4 An outbound value of `return` shall be a valid representation of an integer in the range 0 to 4294967295. An inbound value shall be the canonical representation of an integer in the same range.

9.90.3 Bound Activity Invocation Input

9.90.3.1 The inbound value of `GuiStateCallbackCtx` shall be the canonical representation of the integer in the native parameter with the same name.

9.90.3.2 The inbound value of `Bitmap` shall be determined as follows. If the native parameter with the same name is NULL, then the inbound value shall be an empty string. Otherwise, the inbound value shall be the canonical representation (see 7.7.2) of the octet string in the memory block whose address and length are in the fields `Data` and `Length` (respectively) of the variable of type `BioAPI_DATA` pointed to by the field `Bitmap` of the variable of type `BioAPI_GUI_BITMAP` pointed to by the native parameter `Bitmap`.

9.90.3.3 The inbound value of `BitmapWidth` and `BitmapHeight` shall be determined as follows. If the native parameter `Bitmap` is NULL, then the inbound value shall be "0". Otherwise, the inbound value shall be the canonical representation (see 7.4.3) of the integer in the fields `Width` and `Height` (respectively) of the variable of type `BioAPI_GUI_BITMAP` pointed to by the native parameter `Bitmap`.

9.90.3.4 The inbound value of `return` shall be the canonical representation of the integer returned by the native function.

9.90.4 Bound Activity Invocation Output

The integer represented by the outbound value of `return` shall be returned by the native function.

9.90.5 Default Output

If there is no activity bound to this standard BioAPI interface function, the native function shall return zero.

9.91 BioSPI_EventHandler

9.91.1 Function Invocation Scheme

This function belongs to the framework callback interface, and has the following native prototype:

```
typedef BioAPI_RETURN (*BioSPI_EventHandler)(
    const BioAPI_UUID *BSPUuid,
    BioAPI_UNIT_ID     UnitID,
    const BioAPI_UNIT_SCHEMA *UnitSchema,
    BioAPI_EVENT EventType);
```

and the following parameters:

Parameter	Function invocation (conformance testing model for frameworks)	Bound activity invocation (conformance testing model for BSPs)
`BSPUuid`	input, outbound	input, inbound
`UnitID`	input, outbound	input, inbound
the members of the parameter group "Unit schema" (see 9.2.12) with their names prefixed by "`UnitSchema_`"	input, outbound	input, inbound

EventType	input, outbound	input, inbound
Return	return, inbound	output, outbound

9.91.2 Constraints on the Parameters

9.91.2.1 An outbound value of **BSPUuid** shall be a valid representation of a UUID (see 7.6). An inbound value shall be the canonical representation of a UUID (see 7.6.3).

9.91.2.2 An outbound value of **UnitID** and **EventType** shall be a valid representation of an integer (see 7.4) in the range 0 to 4294967295. An inbound value shall be the canonical representation of an integer (see 7.4.3) in the same range.

9.91.2.3 An outbound value of the parameter group "Unit schema" shall be either a valid representation of a value of the native type **BioAPI_UNIT_SCHEMA**, or a set of empty strings. An inbound value shall be either the canonical representation of a value of that type, or a set of empty strings.

9.91.2.4 An outbound value of **return** shall be a valid representation of an integer in the range 0 to 4294967295. An inbound value shall be the canonical representation of an integer in the same range.

9.91.3 Function Invocation Input

9.91.3.1 The UUID represented by the outbound value of **BSPUuid** shall be written to a variable of type **BioAPI_UUID**, whose address shall be assigned to the native parameter **BSPUuid**.

9.91.3.2 The integer represented by the outbound value of **UnitID** and **EventType** shall be assigned to the native parameter with the same name.

9.91.3.3 If the outbound value of the parameter group "Unit schema" is a set of empty strings, then the native parameter **UnitSchema** shall be set to NULL. Otherwise, the value of type **BioAPI_UNIT_SCHEMA** represented (see 9.2.12.4) by the outbound value of the parameter group "Unit schema" shall be written to a variable of type **BioAPI_UNIT_SCHEMA**, whose address shall be assigned to the native parameter **UnitSchema**.

9.91.4 Function Invocation Output

There are no output parameters.

9.91.5 Bound Activity Invocation Input

9.91.5.1 The inbound value of **BSPUuid** shall be the canonical representation of the UUID in the variable of type **BioAPI_UUID** pointed to by the native parameter **BSPUuid**.

9.91.5.2 The inbound value of **UnitID** and **EventType** shall be the canonical representation of the integer in the native parameter with the same name.

9.91.5.3 If the native parameter **UnitSchema** is NULL, then the inbound value of the parameter group "Unit schema" shall be a set of empty strings. Otherwise, the inbound value shall be the canonical representation (see 9.2.12.5) of the value of type **BioAPI_UNIT_SCHEMA** in the variable pointed to by the native parameter **UnitSchema**.

9.91.6 Bound Activity Invocation Output

The integer represented by the outbound value of **return** shall be returned by the native function.

9.91.7 Default Output

If there is no activity bound to this standard BioAPI interface function, the native function shall return zero.

9.92 BioSPI_GUI_STATE_CALLBACK

9.92.1 Function Invocation Scheme

This function belongs to the framework callback interface, and has the following native prototype:

```
typedef BioAPI_RETURN (BioAPI *BioAPI_GUI_STATE_CALLBACK)(
    void *GuiStateCallbackCtx,
    BioAPI_GUI_STATE GuiState,
    BioAPI_GUI_RESPONSE *Response,
    BioAPI_GUI_MESSAGE Message,
    BioAPI_GUI_PROGRESS Progress,
    const BioAPI_GUI_BITMAP *SampleBuffer);
```

and the following parameters:

Parameter	**Function invocation (conformance testing model for frameworks)**	**Bound activity invocation (conformance testing model for BSPs)**
`GuiStateCallbackCtx`	input, outbound	input, inbound
the members of the parameter group "GUI state" (see 9.2.18)	input, outbound	input, inbound
`no_Response`	input, outbound	input, inbound
`Message`	input, outbound	input, inbound
`Progress`	input, outbound	input, inbound
`BitmapWidth`	input, outbound	input, inbound
`BitmapHeight`	input, outbound	input, inbound
`Bitmap`	input, outbound	input, inbound
`Response`	output, inbound	output, outbound
`Return`	return, inbound	output, outbound

The specification of the constraints on the parameters, bound activity invocation input, bound activity invocation output, and default output of the function **BioAPI_GUI_STATE_CALLBACK** also applies to this standard BioAPI interface function.

9.92.2 Function Invocation Input

9.92.2.1 The integer represented by the outbound value of **GuiStateCallbackCtx**, **Message**, and **Progress** shall be assigned to the native parameter with the same name.

9.92.2.2 The integer represented (see 9.2.18.4) by the outbound value of the parameter group "GUI state" shall be assigned to the native parameter **GuiState**.

9.92.2.3 If the outbound value of **no_Response** is "**true**", then the native parameter **Response** shall be set to NULL, otherwise it shall be set to the address of a variable of type **BioAPI_GUI_RESPONSE**.

9.92.2.4 If the value of **Bitmap** is an empty string, then the native parameter **SampleBuffer** shall be set to NULL. Otherwise:

a) the integer represented by the outbound value of **BitmapWidth** and **BitmapHeight** (or zero if the outbound value is an empty string) shall be written to the fields **Width** and **Height** (respectively) of a variable of type **BioAPI_GUI_BITMAP**, whose address shall be assigned to the native parameter **SampleBuffer**; and

b) the octet string represented by the outbound value of **Bitmap** shall be written to a memory block of sufficient size;

c) the address and length of that memory block shall be written to the fields **Data** and **Length** (respectively) of a variable of type **BioAPI_DATA**;

d) the address of that variable shall be written to the field **Bitmap** of the variable of type **BioAPI_GUI_BITMAP** specified in a).

9.92.3 Function Invocation Output

The inbound value of **Response** shall be determined as follows. If the native parameter with the same name is NULL, then the inbound value shall be an empty string. Otherwise, the inbound value shall be the canonical representation of the integer in the variable of type **BioAPI_GUI_RESPONSE** pointed to by the native parameter **Response**.

9.93 BioSPI_GUI_STREAMING_CALLBACK

9.93.1 Function Invocation Scheme

This function belongs to the framework callback interface, and has the following native prototype:

```
typedef BioAPI_RETURN (BioAPI *BioAPI_GUI_STREAMING_CALLBACK)(
    void *GuiStreamingCallbackCtx,
    const BioAPI_GUI_BITMAP *Bitmap);
```

and the following parameters:

Parameter	Function invocation (conformance testing model for frameworks)	Bound activity invocation (conformance testing model for BSPs)
`GuiStreamingCallbackCtx`	input, outbound	input, inbound
`BitmapWidth`	input, outbound	input, inbound
`BitmapHeight`	input, outbound	input, inbound
`Bitmap`	input, outbound	input, inbound
`return`	return, inbound	output, outbound

The specification of the constraints on the parameters, bound activity invocation input, bound activity invocation output, and default output of the function `BioAPI_GUI_STREAMING_CALLBACK` also applies to this standard BioAPI interface function.

9.93.2 Function Invocation Input

9.93.2.1 The integer represented by the outbound value of `GuiStreamingCallbackCtx` shall be assigned to the native parameter with the same name.

9.93.2.2 If the outbound value of `Bitmap` is an empty string, then the native parameter `Bitmap` shall be set to NULL. Otherwise:

a) the integer represented by the outbound value of `BitmapWidth` and `BitmapHeight` (or zero if the outbound value is an empty string) shall be written to the fields `Width` and `Height` (respectively) of a variable of type `BioAPI_GUI_BITMAP`, whose address shall be assigned to the native parameter `Bitmap`;

b) the octet string represented by the outbound value of `Bitmap` shall be written to a memory block of sufficient size;

c) the address and length of that memory block shall be written to the fields `Data` and `Length` (respectively) of a variable of type `BioAPI_DATA`; and

d) the address of that variable shall be written to the field `Bitmap` of the variable of type `BioAPI_GUI_BITMAP` specified in a).

9.93.3 Function Invocation Output

There are no output parameters.

9.94 BioSPI_BFP_ENUMERATION_HANDLER

9.94.1 Function Invocation Scheme

This function belongs to the application callback interface, and has the following native prototype:

typedef BioAPI_RETURN (BioAPI *BioSPI_BFP_ENUMERATION_HANDLER)(

BioAPI_BFP_SCHEMA **BFPSchemaArray,

uint32_t *NumberOfElements);

and the following parameters:

Parameter	Function invocation (conformance testing model for frameworks)	Bound activity invocation (conformance testing model for BSPs)
`no_BFPSchemaArray`	input, outbound	input, inbound
the members of the parameter group "BFP schema" (see 9.2.11) with their names prefixed by "`BFPSchema_1_`"	output, inbound	output, outbound
the members of the parameter group "BFP schema" with their names prefixed by "`BFPSchema_2_`"	output, inbound	output, outbound
the members of the parameter group "BFP schema" with their names prefixed by "`BFPSchema_3_`"	output, inbound	output, outbound
the members of the parameter group "BFP schema" with their names prefixed by "`BFPSchema_4_`"	output, inbound	output, outbound
`no_NumberOfElements`	input, outbound	input, inbound
`NumberOfElements`	output, inbound	output, outbound
`Return`	return, inbound	output, outbound

9.94.2 Constraints on the Parameters

9.94.2.1 An outbound value of `no_BFPSchemaArray` and `no_NumberOfElements` shall be either a valid representation of a boolean (see 7.5) or an empty string. An inbound value shall be the canonical representation of a boolean (see 7.5.2).

9.94.2.2 An outbound value of each parameter group "BFP schema" shall be a valid representation of a value of the native type **`BioAPI_BFP_SCHEMA`**. An inbound value shall be either the canonical representation of a value of that type, or a set of empty strings.

9.94.2.3 An outbound value of **`NumberOfElements`** shall be either a valid representation of an integer in the range 0 to 4294967295, or an empty string. An inbound value shall be either the canonical representation of an integer in the same range, or an empty string.

9.94.2.4 An outbound value of **`return`** shall be a valid representation of an integer in the range 0 to 4294967295. An inbound value shall be the canonical representation of an integer in the same range.

9.94.3 Function Invocation Input

9.94.3.1 If the outbound value of **`no_BFPSchemaArray`** is "**`true`**", then the native parameter **`BFPSchemaArray`** shall be set to NULL, otherwise it shall be set to the address of a variable of type **`BioAPI_BFP_SCHEMA*`** (a pointer).

9.94.3.2 If the outbound value of **`no_NumberOfElements`** is "**`true`**", then the native parameter **`NumberOfElements`** shall be set to NULL, otherwise it shall be set to the address of a variable of type **`uint32_t`**.

9.94.4 Function Invocation Output

9.94.4.1 The inbound value of the parameter group "BFP schema" with the prefix "**`BFPSchema_`***X***`_`**" (with *X*=1, 2, 3, or 4) shall be determined as follows. If the native parameter **`BFPSchemaArray`** is NULL or the variable pointed to by that native parameter is NULL or the native parameter **`NumberOfElements`** is NULL or the value of the integer variable pointed to that native parameter is less than *X*, then the inbound value shall be a set of empty strings. Otherwise, the inbound value shall be the canonical representation (see 9.2.11.5) of the value of type **`BioAPI_BFP_SCHEMA`** at position *X* in the array pointed to by the variable pointed to by the native parameter **`BFPSchemaArray`**.

9.94.4.2 The inbound value of **`NumberOfElements`** shall be determined as follows. If the native parameter with the same name is NULL, then the inbound value shall be an empty string. Otherwise, the inbound value shall be the canonical representation of the integer in the variable of type **`uint32_t`** pointed to by the native parameter **`NumberOfElements`**.

9.94.5 Bound Activity Invocation Input

9.94.5.1 The inbound value of **`no_BFPSchemaArray`** shall be "**`true`**" if the native parameter **`BFPSchemaArray`** is NULL, otherwise it shall be "**`false`**".

9.94.5.2 The inbound value of **`no_NumberOfElements`** shall be "**`true`**" if the native parameter **`NumberOfElements`** is NULL, otherwise it shall be "**`false`**".

9.94.6 Bound Activity Invocation Output

9.94.6.1 If the native parameter **`BFPSchemaArray`** is not NULL and the integer (say, *N*) represented by the outbound value of **`NumberOfElements`** is greater than 0, then a memory block of sufficient size to contain an array of *N* elements of type **`BioAPI_BFP_SCHEMA`** shall be allocated and its address shall be written to the variable pointed to by the native parameter **`BFPSchemaArray`**.

9.94.6.2 The outbound value of the parameter group "BFP schema" with the prefix **`BFPSchema_`***X***`_`** (with *X*=1, 2, 3, or 4) shall be processed as follows. If the native parameter **`BFPSchemaArray`** is NULL or the integer represented by the outbound value of **`NumberOfElements`** is less than *X*, then the outbound value of the parameter group shall be ignored. Otherwise, the value of type **`BioAPI_BFP_SCHEMA`** represented by the outbound value of the parameter group shall be written to the element at position *X* of the array of elements of type **`BioAPI_BFP_SCHEMA`** pointed to by the variable pointed to by the native parameter **`BFPSchemaArray`**.

9.94.6.3 The outbound value of **`NumberOfElements`** shall be processed as follows. If the native parameter with the same name is NULL, then the outbound value shall be ignored. Otherwise, the integer represented by the outbound value shall be written to the variable of type **`uint32_t`** pointed to by the native parameter **`NumberOfElements`**.

9.94.6.4 The integer represented by the outbound value of **`return`** shall be returned by the native function.

9.94.7 Default Output

If there is no activity bound to this standard BioAPI interface function, then:

a) the variable pointed to by the native parameter **`NumberOfElements`** shall be set to zero;

b) the variable pointed to by the native parameter **`BFPSchemaArray`** shall be set to NULL; and

c) the native function shall return zero.

9.95 BioSPI_MEMORY_FREE_HANDLER

9.95.1 Function Invocation Scheme

This function belongs to the application callback interface, and has the following native prototype:

```
typedef BioAPI_RETURN (BioAPI *BioSPI_MEMORY_FREE_HANDLER)(
    void* Ptr);
```

and the following parameters:

Parameter	Function invocation (conformance testing model for frameworks)	Bound activity invocation (conformance testing model for BSPs)
`Ptr`	input, outbound	input, inbound
`return`	return, inbound	output, outbound

9.95.2 Constraints on the Parameters

9.95.2.1 An outbound value of **`Ptr`** shall be a valid representation of an integer (see 7.4) in the range 0 to 4294967295. An inbound value shall be the canonical representation of an integer (see 7.4.3) in the same range.

9.95.2.2 An outbound value of **`return`** shall be a valid representation of an integer in the range 0 to 4294967295. An inbound value shall be the canonical representation of an integer in the same range.