INTERNATIONAL STANDARD

ISO/IEC 24709-3

First edition
2011-01-15

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

Part 3: Test assertions for BioAPI frameworks

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

Partie 3: Déclarations d'essai pour cadres BioAPI

Reference number
ISO/IEC 24709-3:2011(E)

Contents

Page

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 24709-3 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*

— *Part 3: Test assertions for BioAPI frameworks*

The following part is under preparation:

— *Part 4: Test assertions for BioAPI applications*

Introduction

The test assertions specified in this part of ISO/IEC 24709 enable a user of this part of ISO/IEC 24709 (such as a testing laboratory) to test the conformance to ISO/IEC 19784-1 (BioAPI2.0) of any BioAPI framework that claims to be a conforming implementation of that International Standard.

The organization of the test assertions in this part of ISO/IEC 24709 reflects the structure of Annex A of ISO/IEC 19784-1:2006, which specifies conformance to BioAPI for various types of implementations (BSPs, frameworks, and applications) and for BSPs belonging to several conformance subclasses.

This part of ISO/IEC 24709 contains test assertions for testing conformance of BioAPI frameworks to claim compliance to the BioAPI specification defined by ISO/IEC 19784-1:2006. The assertions are further organized according to conformance subclass (if any) and claimed support of optional features.

Each test assertion exercises one or more (possibly elementary) features of an implementation under test. Assertions are placed into packages (one or more assertions per package) as required by the assertion language.

Clause 6 specifies general principles.

Clause 7 specifies the principles and the testing mechanism for the conformance testing for BioAPI frameworks in addition to listing up the test assertions to be used in this conformance testing model.

Clause 8 specifies the assertions to be used in the conformance testing model for BioAPI frameworks.

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

Part 3: Test assertions for BioAPI frameworks

1 Scope

This part of ISO/IEC 24709 defines a number of test assertions written in the assertion language specified in ISO/IEC 24709-1:2007.

This part of ISO/IEC 24709 specifies all the test assertions that are to be executed for conformance testing of BioAPI frameworks claiming conformance to ISO/IEC 19784-1 (BioAPI 2.0).

Test assertions specified in this part of ISO/IEC 24709 are not claimed to be exhaustive (see also ISO/IEC 24709-1:2007, Clause 6). Implementations of BioAPI 2.0 that are tested according to the methodology specified in ISO/IEC 24709-1:2007 and with test assertions specified in this part of ISO/IEC 24709 can (only) claim conformance to those aspects of ISO/IEC 19784-1 that are covered by these test assertions.

2 Conformance

Implementations (BioAPI conformance test suites) claiming conformance to this part of ISO/IEC 24709 shall be able to process all the test assertions specified in Clause 8 according to the methodology specified in ISO/IEC 24709-1:2007 and the general principles and provisions specified in Clauses 6 and 7.

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 19784-1:2006, *Information technology — Biometric application programming interface — Part 1: BioAPI specification*

ISO/IEC 24709-1:2007, *Information technology — Conformance testing for the biometric application programming interface (BioAPI) — Part 1: Methods and procedures*

4 Terms and definitions

For the purposes of this document, the terms and definitions given in ISO/IEC 19784-1:2006, ISO/IEC 24709-1:2007 and the following apply.

4.1
API/SPI routing
feature provided by the BioAPI Framework to handle multiple applications and/or multiple biometric service providers (BSPs), with which a BioAPI call from an application is correctly given to the BSP specified by the application, and with which a BioSPI return from a BSP is correctly given to the application that specified the BSP

5 Abbreviated terms

For the purposes of this document, the following abbreviated terms apply.

API	application programming interface
BIR	biometric information record
BSP	biometric service provider
CBEFF	common biometric exchange format framework
FMR	false match rate
FPI	function provider interface
GUI	graphic user interface
ID	identity/identification/identifier
SPI	service provider interface
UUID	universally unique identifier
BCS	BioAPI conformity statement
CTS	BioAPI conformance test suite
IUT	implementation under test

6 General principles

6.1 The test assertions listed in Clause 7 and specified in Clause 8 are based on the conformance testing methodology specified in ISO/IEC 24709-1:2007, and can only be used in the context of that methodology. The assertions are written in the assertion language specified in ISO/IEC 24709-1:2007, which is part of that methodology.

6.2 An important concept of the conformance testing methodology specified in ISO/IEC 24709-1:2007 is the existence of three conformance testing models (see ISO/IEC 24709-1:2007, Clause 6).

a) the conformance testing model for BioAPI applications

b) the conformance testing model for BioAPI frameworks

c) the conformance testing model for BioAPI BSPs

6.3 Each testing model is concerned with testing one of the three standard components of the BioAPI architecture (see ISO/IEC 24709-1:2007, Clause 6). Clause 8 of this part of ISO/IEC 24709 specifies a number of test assertions for the conformance testing model for BioAPI frameworks. This part of ISO/IEC 24709 is not concerned with the conformance testing models for BioAPI applications, BSPs and corresponding test assertions.

6.4 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 6.2.5.2 of ISO/IEC 24709-1:2007).

7 Testing the conformance of BioAPI frameworks

7.1 General

7.1.1 This subclause describes the principles of the test conditions of the conformance test for BioAPI frameworks and the principles to create pass/fail results.

7.1.2 The principles of the test conditions of the BioAPI frameworks under test are as follows:

- All the BioAPI functions and all the related BioSPI functions shall be tested.
- All the parameters defined in each function shall be tested.
- All the values that are able to be specified in each of the parameters shall be tested.
- All the capabilities in the BSP schema of the testing BSP that are related to the test case shall be tested.

7.1.3 The principles to create pass/fails results of the test cases of the BioAPI frameworks are as follows:

- The return value shall be checked. If ISO/IEC 19784-1:2006 does not specify an error value for a specific BioAPI function, the test shall pass if one of the possible error values has been returned.
- As for the output parameters, they shall be checked if ISO/IEC 19784-1:2006 declares that the BioAPI framework sets the values to the output parameters.
- As for the parameters that are transferred from the BioAPI function to the BioSPI function by the BioAPI framework, the test assertions shall check if all the parameters are correctly transferred.

NOTE 1 In this edition of ISO/IEC 24709-3, the test assertions do not contain (i) the test cases with combinations of various parameters, (ii) the test cases with various sequences of BioAPI functions, (iii) the test cases with the callback functions, (iv) the test cases related to asynchronous behaviors, (v) error cases derived from the incorrect implementation of the testing BSP and (vi) the test cases that are only relevant to optional features of the testing BSP. The excluded test cases will be considered in the next edition of this part of ISO/IEC 24709. The excluded test cases will be considered in the next edition of this part of ISO/IEC 24709.

NOTE 2 A test case for the multiple component support of the BioAPI Framework is described in Annex A as a recommendation for the Conformance Test Suite to implement the test for the API/SPI routing feature of the BioAPI Framework.

NOTE 3 As for the error handling of invalid parameters, ISO/IEC 19784-1:2006 makes allowance for the freedom of implementation for BioAPI frameworks and BSPs. This text shall take into account the following cases to create the test assertions conformant to ISO/IEC 19784-1:2006:

(a) In the specification of ISO/IEC 19784-1:2006, it does not necessarily specify one error value to be returned from the BioAPI function if there is an incorrect value specified in one of the parameters in the BioAPI function. In such cases, the test assertions only check if one of the possible error values is returned and do not define in the test assertion that a specific error value shall be returned.

(b) There is no specific description in ISO/IEC 19784-1:2006 whether the BioAPI framework shall do the parameter check or the testing BSP shall do it. Therefore in this text, the test assertions do not care about which component (BioAPI framework or BSP) has found the errors when it checked the parameters. Therefore, the test assertions do not care about the error codes which includes the highest 8 bits that is allowed to be set by the BioAPI framework or by the testing BSP. It also do not care about whether the BioAPI framework calls the corresponding BioSPI function even after there is an error in the parameters in the BioAPI functions. The followings are the examples of the error cases that the BioAPI framework can positively handle the error without calling the corresponding BioSPI function.

1. The value in the parameter is irrelevant; the value not relevant to the enrollment purpose such as BioAPI_PURPOSE_AUDIT is set to the Purpose parameter in the BioAPI_Enroll function.

2. The value is not supported by the BSP; even though the BSP does not support the feature to specify Subtype, the value for the Subtype parameter such as BioAPI_BIR_RIGHT is given.

7.2 Configuration of test assertions

7.2.1 A test assertion consists of the three tables that exist per BioAPI function under test, and the XML text that exists per BioAPI function or BioSPI function.

7.2.2 The three tables that comprise a part of the test assertions are (i) Default Input Table, which assembles the default values for all the input parameters of the BioAPI function under test, (ii) Test Condition Table, which assembles all the conditions given during the BioAPI framework is tested, and (iii) Expected Result Table, which is used to create pass/fail results by comparing the expected test results in the table with the values given from the BioAPI framework. In Test Condition Table, each row shows a test case and each column shows (i) the values given to the parameters of the BioAPI function, (ii) the BSP Schema information related to the test case and (iii) the return value from the BSP via the BioSPI function. In Expected Result Table, each row shows a test case same as Test Condition Table and each column shows the information that is used to make a pass/fail decision by referring to it. To make a decision, the return value and the output parameters related to the test case are used.

NOTE In the test assertions in each subclause in Chapter 8, it simply uses the words Test Condition Table and Expected Result Table without putting the suffixes such as the BioAPI and/or BioSPI function names to avoid redundancy.

7.2.3 Before calling a BioAPI function to be tested, the testing application shall read the values described in the Default Input Table first then read one of test conditions described in the Test Condition Table by picking up one of the rows in the table, which means that all the input parameters are set by the testing application by referring to the Default Input Table but one of the parameters is overwritten by the value described in the Test Condition Table. The testing application shall repeat reading the two tables every time it executes a test case.

7.2.4 In Default Input Table, the input parameter names and the input parameter values for the BioAPI function under test are described. The sequence of the parameters are the same as the one described in ISO/IEC 24709-1.

7.2.5 In Test Condition Table, the following information is described.

(a) Input parameter name and input parameter value: Describes the parameters given to the BioAPI function to be tested.

(b) Supported options in BSP Schema: Chooses the options in BioAPI_OPERATIONS_MASK and BioAPI_OPTIONS_MASK in the BSP Schema that are related to the test case to show these options are supported or not in the test case. The detail of Test Condition Table is described in 7.3 and Clause 8.

(c) Return value (from BioSPI): Shows a value to be returned from the testing BSP after the BioSPI function is invoked by the BioAPI framework. One return value that is thought to be adequate is selected from the possible return values and described in the table.

7.2.6 In Expected Result Table, each row includes the return value and one of the names of the parameters and its value to make a decision of the pass/fail results. The detail of Expected Result Table is shown in 7.3 and Clause 8.

7.2.7 In the XML texts, it sets up the parameters from Default Input Table described in 7.2.4 and Test Condition Table described in 7.2.5 using the <input> element before calling the BioAPI function to be tested, then makes a pass/fail decision by obtaining the expected results from Expected Result Table described in 7.2.6 using the <input> element after the BioAPI function returns to the testing application. Each test case has the above mentioned logic in common for one BioAPI function or one BioSPI function, so in principle, there is only one XML text for one BioAPI function or one BioSPI function without having any exception that is to be applied to a particular test case.

7.2.8 The structure of CTS for BioAPI frameworks is depicted in Figure 1.

7.2.8.1 CTS consists of the testing application, the testing BSP, XML text, the two test tables and the BioAPI framework under test. Both the testing application and the testing BSP read the XML text and the test tables and run the script after translating it to the binaries executable on a computer. In addition, the testing application and the testing BSP check the behavior of the BioAPI framework through the information given by the BioAPI framework and make a pass/fail decision.

NOTE Setting the values given in the Test Condition Table to the variables defined in the XML text is responsible for CTS and it is not in the scope of this part of ISO/IEC 24709.

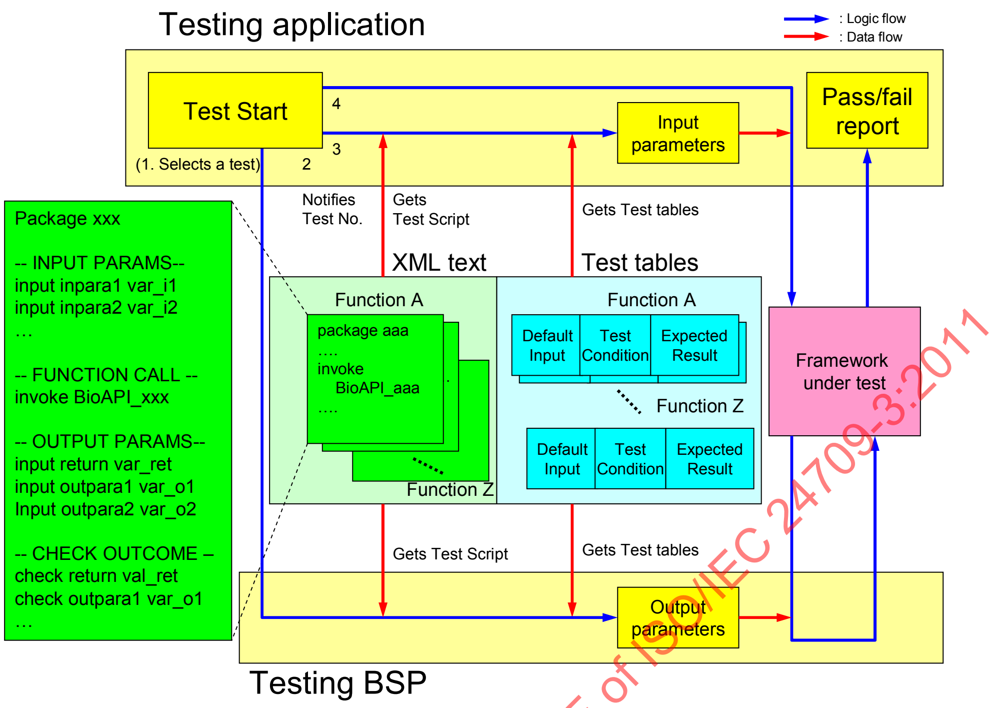

Figure 1 — Structure of CTS for BioAPI framework

7.2.8.2 In accordance with the descriptions in Test Condition Table, the testing BSP shall have a capability to change the members of BioAPI_OPERATIONS_MASK and BioAPI_OPTIONS_MASK, which are part of the BSP Schema. The members of the BSP Schema related to the conformance test for BioAPI frameworks are shown in Table 1.

Table 1 — Members of BSP Schema related to 24709-3

Number	Members	Value
1	BSPUuid	Depends on CTS
2	Description	Depends on CTS
3	SpecVersion	0x20 (Version 2.0)
4	ProductVersion	Depends on CTS
5	Vendor	Depends on CTS
6	FactorsMask (Biometric Type)	0x00000001 (BioAPI_TYPE_MULTIPLE)
7	Operations	Depends on Test Condition Table
8	Options	Depends on Test Condition Table
9	PayloadPolicy	Depends on CTS
10	MaxPayloadSize	1024 (bytes)
11	DefaultVerifyTimeout	10000 (milliseconds)
12	DefaultIdentifyTimeout	10000 (milliseconds)
13	DefaultCaptureTimeout	10000 (milliseconds)
14	DefaultEnrollTimeout	10000 (milliseconds)
15	DefaultCalibrateTimeout	10000 (milliseconds)
16	MaxBSPDbSize	10240 (bytes)
17	MaxIdentify	0xFFFFFFFF (Unlimited)

7.3 Test flow

7.3.1 Prior to executing a test, one test case is selected (see the number 1 in Figure –1).

7.3.2 The testing application notifies the testing BSP the information which indicates a unique test case is going to be executed (see the number 2 in Figure 1). The implementation of the notification is not in the scope of this part of ISO/IEC 24709, so it depends on each CTS. (e.g. The testing BSP supports a function specific to the purpose of the notification that is only known by the testing application, then it calls the function prior to the invocation of the BioAPI function to be tested so the testing BSP can find which test case has been selected.) With such a feature for notification, the testing BSP can prepare for the test case by reading the corresponding XML text and the test tables in advance and create the appropriate BSP Schema and the return value for the BioSPI function in case the BioAPI framework invokes the BioSPI function during the test.

7.3.3 The testing application reads the corresponding XML text and the test tables same as the testing BSP does, and prepares the parameters of the BioAPI function to be tested (see the number 3 in Figure 1).

NOTE 7.3.2 and 7.3.3 can be in no particular order.

7.3.4 The testing application calls the BioAPI function after setting the parameters corresponding to the selected test case (see the number 4 in Figure -1). It depends on the implementation of the BioAPI framework whether to return an error or not when it finds a contradiction between one of the specified parameters and one of the capabilities in BioAPI_OPERATIONS_MASK or BioAPI_OPTIONS_MASK in the BSP Schema. In the former case, the BioAPI framework returns an error value to the testing application without calling the testing BSP. A pass/fail report will be created by the testing application (see 7.3.7).

7.3.5 In 7.3.4, if the BioAPI framework calls the BioSPI function after it is invoked from the testing application via the BioAPI call, the testing BSP checks the validity of the behavior of the BioAPI framework by checking the parameters with the expected results described in Expected Result Table. (In many cases, it checks whether the parameters in the BioSPI function are the same as the parameters in the BioAPI function or not.) If it finds out that the parameters in the BioSPI function are different from the description in Expected Result Table, it creates a fail report and terminates the test case.

7.3.6 If the test assertion finds out that the parameters of the BioSPI functions are correct, the testing BSP sets a return value by referring to the corresponding area in Test Condition Table and gives the control to the BioAPI framework, then it returns to the testing application.

7.3.7 The testing application checks whether the information returned from the BioAPI framework is consistent with the descriptions in Expected Result Table. If all the members of the information are correct, it creates a pass report. If one or more members is not correct, it creates a fail report.

7.4 Initialisation and termination

7.4.1 All the test cases shall include the initialisation of the BioAPI framework and the testing BSP. By initialising these components, each test is independent from the other test cases by avoiding any influence from the tests done prior to the test.

7.4.2 The initialisation includes the BioAPI_Init and the BioAPI_Util_InstallBSP so that, in addition to the initialisation of the BioAPI framework, the capabilities of the testing BSP corresponding to the test case will be created in the BSP Schema in the component registry every time the test starts. (The testing BSP knows which capabilities it shall have in the BSP Schema by the notification of the test number from the testing application.) The termination includes the BioAPI_Util_InstallBSP and the BioAPI_Terminate, so the component registry will be deleted and the BioAPI framework will be terminated every time the test finishes.

7.5 List of test assertions

7.5.1 The following tables list the test assertions for the BioAPI and BioSPI functions in accordance with the categories of the BioAPI functions described in ISO/IEC19784-1:2006.

a) Component Management Functions
b) Data Handle Operations
c) Callback and Event Handling Operations
d) Biometric Operations
e) Database Operations
f) BioAPI Unit Operations
g) Utility Functions
h) Component Registry Functions

NOTE Successful processing of all applicable test assertions is prima facie evidence that the implementation satisfies the applicable requirements of ISO/IEC19784-1, but does not establish this, as the assertions are not (and cannot be) an exhaustive test of conformance (see also ISO/IEC 29709-1:2007, Clause 6).

7.5.1.1 For the Component Management Functions, the implementation shall be tested by executing all of the following assertions (in order):

Table 2 — Test assertions for component management functions

Number	Assertion Name	References in 19784-1:2006	Package
1.1	BioAPI_Init	8.1.1, 11.2.3	4839c860-7929-11de-8a39-0800200c9a66
1.2	BioAPI_Teminate	8.1.2	8782cd50-7929-11de-8a39-0800200c9a66
1.3	BioAPI_GetFrameworkInfo	8.1.3	b3a468d0-7929-11de-8a39-0800200c9a66
1.4	BioAPI_EnumBSPs	8.1.4	ce45e240-7929-11de-8a39-0800200c9a66
1.5	BioAPI_BSPLoad_And_BioSPI_BSPLoad	8.1.5, 9.3.1	f481f070-7929-11de-8a39-0800200c9a66
1.6	BioAPI_BSPUnload_And_BioSPI_BSPUnload	8.1.6, 9.3.1.2	1067a9b0-792a-11de-8a39-0800200c9a66
1.7	BioAPI_BSPAttach_And_ BioSPI_BSPAttach	8.1.7, 9.3.1.3	2ae45d10-792a-11de-8a39-0800200c9a66
1.8	BioAPI_BSPDetach_And_BioSPI_BSPDetach	8.1.8, 9.3.1.4	4149b370-792a-11de-8a39-0800200c9a66
1.9	BioAPI_QueryUnits_And_BioSPI_QueryUnits	8.1.9, 9.3.1.5	507a4030-792a-11de-8a39-0800200c9a66
1.10	BioAPI_EnumBFPs	8.1.10	62eb03d0-792a-11de-8a39-0800200c9a66
1.11	BioAPI_QueryBFPs_And_BioSPI_QueryBFPs	8.1.11, 9.3.1.6	70d92580-792a-11de-8a39-0800200c9a66
1.12	BioAPI_ControlUnit_And_BioSPI_ControlUnit	8.1.12, 9.3.1.7	819d98b0-792a-11de-8a39-0800200c9a66

7.5.1.2 For the Data Handle Operation Functions, the implementation shall be tested by executing all of the following assertions (in order):

Table 3 — Test assertions for data handle operation functions

Number	Assertion Name	References in 19784-1:2006	Package
2.1	BioAPI_FreeBIRHandle_And_BioSPI_FreeBIRHandle	8.2.1, 9.3.2.1	94a32240-792a-11de-8a39-0800200c9a66
2.2	BioAPI_GetBIRFromHandle_And_BioSPI_GetBIRFromHandle	8.2.2, 9.3.2.2	ca10cea0-792a-11de-8a39-0800200c9a66
2.3	BioAPI_GetHeaderFromHandle_And_BioSPI_GetHeaderFromHandle	8.2.3, 9.3.2.3	d9332a90-792a-11de-8a39-0800200c9a66

7.5.1.3 For the Callback and Event Operation Functions, the implementation shall be tested by executing all of the following assertions (in order):

Table 4 — Test assertions for callback and event operation functions

Number	Assertion Name	References in 19784-1:2006	Package
3.1	BioAPI_EnableEvents_And_BioSPI_EnableEvents	8.3.1, 9.3.3.1	c0c4abd0-792c-11de-8a39-0800200c9a66
3.2	BioAPI_SetGUICallbacks_And_BioSPI_SetGUICallbacks	8.3.2, 9.3.3.2	f0a2b310-792c-11de-8a39-0800200c9a66

7.5.1.4 For the Biometric Operation Functions, the implementation shall be tested by executing all of the following assertions (in order):

Table 5 — Test assertions for biometric operation functions

Number	Assertion Name	References in 19784-1:2006	Package
4.1	BioAPI_Capture_And_BioSPI_Capture	8.4.1, 9.3.4.1	aaec0fa0-792d-11de-8a39-0800200c9a66
4.2	BioAPI_CreateTemplate_And_BioSPI_CreateTemplate	8.4.2, 9.3.4.2	b851a060-792d-11de-8a39-0800200c9a66
4.3	BioAPI_Process_And_BioSPI_Process	8.4.3, 9.3.4.3	c9a0a050-792d-11de-8a39-0800200c9a66
4.4	BioAPI_ProcessWithAuxBIR_And_BioSPI_ProcessWithBIR	8.4.4, 9.3.4.4	eab95fc0-792d-11de-8a39-0800200c9a66
4.5	BioAPI_VerifyMatch_And_BioSPI_VerifyMatch	8.4.5, 9.3.4.5	f66559a0-792d-11de-8a39-0800200c9a66
4.6	BioAPI_IdentifyMatch_And_BioSPI_IdentifyMatch	8.4.6, 9.3.4.6	028794f0-792e-11de-8a39-0800200c9a66
4.7	BioAPI_Enroll_and_BioSPI_Enroll	8.4.7, 9.3.4.7	12892d50-792e-11de-8a39-0800200c9a66
4.8	BioAPI_Verify_And_BioSPI_Verify	8.4.8, 9.3.4.8	233d9af0-792e-11de-8a39-0800200c9a66
4.9	BioAPI_Identify_And_BioSPI_Identify	8.4.9, 9.3.4.9	30b63e80-792e-11de-8a39-0800200c9a66

4.10	BioAPI_Import_And_BioSPI_Import	8.4.10, 9.3.4.10	4019c220-792e-11de-8a39-0800200c9a66
4.11	BioAPI_PresetIdentifyPopulation_And_BioSPI_PresetIdentifyPopulation	8.4.11, 9.3.4.11	55440070-792e-11de-8a39-0800200c9a66

7.5.1.5 For the Database Operation Functions, the implementation shall be tested by executing all of the following assertions (in order):

Table 6 — Test assertions for database operation functions

Number	Assertion Name	References in 19784-1:2006	Package
5.1	BioAPI_DbOpen_And_BioSPI_DbOpen	8.5.1, 9.3.5.1	9f31c870-792e-11de-8a39-0800200c9a66
5.2	BioAPI_DbClose_And_BioSPI_DbClose	8.5.2, 9.3.5.2	aa8f2d20-792e-11de-8a39-0800200c9a66
5.3	BioAPI_DbCreate_And_BioSPI_DbCleate	8.5.3, 9.3.5.3	b5f8ede0-792e-11de-8a39-0800200c9a66
5.4	BioAPI_DbDelete_And_BioSPI_DbDelete	8.5.4, 9.3.5.4	c60b0100-792e-11de-8a39-0800200c9a66
5.5	BioAPI_DbSetMarker_And_BioSPI_DbSetMarker	8.5.5, 9.3.5.5	d06aa4c0-792e-11de-8a39-0800200c9a66
5.6	BioAPI_DbFreeMarker_And_BioSPI_DbFreeMarker	8.5.6, 9.3.5.6	e4647960-792e-11de-8a39-0800200c9a66
5.7	BioAPI_DbStoreBIR_And_BioSPI_DbStoreBIR	8.5.7, 9.3.5.7	f2614110-792e-11de-8a39-0800200c9a66
5.8	BioAPI_DbGetBIR_And_BioSPI_DbGetBIR	8.5.8, 9.3.5.8	020724e0-792f-11de-8a39-0800200c9a66
5.9	BioAPI_DbGetNextBIR_And_BioSPI_DbGetNextBIR	8.5.9, 9.3.5.9	13003ca0-792f-11de-8a39-0800200c9a66
5.10	BioAPI_DbDeleteBIR_And_BioSPI_DbDeleteBIR	8.5.10, 9.3.5.10	2068daa0-792f-11de-8a39-0800200c9a66

7.5.1.6 For the BioAPI Unit Operation Functions, the implementation shall be tested by executing all of the following assertions (in order):

Table 7 — Test assertions for BioAPI unit operation functions

Number	Assertion Name	References in 19784-1:2006	Package
6.1	BioAPI_SetPowerMode_And_BioSPI_SetPowerMode	8.6.1, 9.3.6.1	5cda0770-792f-11de-8a39-0800200c9a66
6.2	BioAPI_SetIndicatorStatus_And_BioSPI_SetIndicatorStatus	8.6.2, 9.3.6.2	678e0cc0-792f-11de-8a39-0800200c9a66
6.3	BioAPI_GetIndicatorStatus_And_BioSPI_GetIndicatorStatus	8.6.3, 9.3.6.3	72b67ec0-792f-11de-8a39-0800200c9a66
6.4	BioAPI_CalibrateSensor_And_BioSPI_CalibrateSensor	8.6.4, 9.3.6.4	833adbb0-792f-11de-8a39-0800200c9a66

7.5.1.7 For the Utility Functions, the implementation shall be tested by executing all of the following assertions (in order):

Table 8 — Test assertions for utility functions

Number	Assertion Name	References in 19784-1:2006	Package
7.1	BioAPI_Cancel_And_BioSPI_Cancel	8.7.1, 9.3.7.1	bd5c9a40-792f-11de-8a39-0800200c9a66
7.2	BioAPI_Free_And_BioSPI_Free	8.7.2, 9.3.7.2	c9b41660-792f-11de-8a39-0800200c9a66

7.5.1.8 For the Component Registry Functions, the implementation shall be tested by executing all of the following assertions (in order):

Table 9 — Test assertions for component registry functions

Number	Assertion Name	References in 19784-1:2006	Package
8.1	BioAPI_Util_InstallBSP	10.2.1.1	f7bc2520-792f-11de-8a39-0800200c9a66
8.2	BioAPI_Util_InstallBFP	10.2.2	033b1b90-7930-11de-8a39-0800200c9a66

7.5.2 Each one of the test assertions is associated with one of the BioAPI and/or BioSPI functions. One test assertion is used multiple times by changing the conditions and expected test results as specified in the two test tables for each test assertion described in Clause 8.

7.6 BioAPI conformity statement

7.6.1 For the conformance testing of BioAPI frameworks, a BioAPI Conformity Statement, as defined in ISO/IEC 24709-1:2007, Clause 6.1, shall be produced by the vendor of the BioAPI framework and shall consist of the following information:

Table 10 — BioAPI conformity statement

Vender contact information:		
Name:		
Address:		
Street		
City		
State or province		
Zip code or postal code		
Country		
Telephone		
Biometric product:		
Name		
Serial number		
description		
BioAPI conformance class (one of the following):		
BioAPI application		
BioAPI framework (this alternative shall be selected)		
BSP		
Product ID of Framework		
Maximum supported size for the payload		
Additional information		
Framework schema	UUID of the framework(FrameworkUuid)	
	Additional information (Description)	
	BioAPI specification version(SpecVersion)	
	Product version (ProductVersion)	
	Vendor (Vendor)	
	FW Property ID (FwPropertyId)	
	FW Property (FwProperty)	

8 Test assertions

8.1 Descriptions of Test Tables

In this part of ISO/IEC 24709, three types of Test Tables, which are Default Input Table, Test Condition Table and Expected Result Table, are used to describe the test cases.

8.1.1 Default Input Table

8.1.1.1 This table describes the default input parameters of the BioAPI function under test. This table has the columns "Input parameter name" and "Input parameter value (underscore: invalid)", which indicate the default values to be set to each parameter of the BioAPI function.

8.1.1.2 The parameter values described in this table shall be set by the testing application every time before executing each test case.

8.1.2 Test Condition Table

8.1.2.1 Each test number indicates a different test case in the test assertion. It forms a specific test case by giving a different value to one of the parameters for the BioAPI function or by setting different capabilities in the BSP Schema or by setting a different return value for the BioSPI function.

8.1.2.2 The name of each parameter is described in the column 'Input parameter name', and the capabilities of BSP Schema is described in the column 'Supported options in BSP Schema', which consists of two columns named 'OPERATIONS_MASK' and 'OPTIONS_MASK', each of which indicates that if there is one of the function names or one of the options, it means it is supported by the testing BSP in that test case.

8.1.2.3 In each test case, after setting all the parameters by referring to the Default Input Table as mentioned in 8.1.1, the testing application refers to the columns "Input parameter name" and "Input parameter value" in Test Condition Table and gives a specific value to the specified parameter to form an individual test case. By giving a different value and choosing a different parameter, there will be various test cases in Test Condition Table based on the principles described in 7.1.2.

8.1.2.4 If a parameter of the BioAPI function consists of a combination of more than one values such as Subtype or ReferenceTemplate, it is expressed with a combination of a group of parameters and a group of values with using the parenthes and commas.

NOTE The Subtype of left and pointer finger is expressed with the combination of the parameter name "(Left, PointerFinger)' and the parameter value "(true, true)".

8.1.3 Expected Result Table

8.1.3.1 Each test number is corresponding to the same test number in Test Condition Table, therefore the contents of the cells in each number indicate the expected results of the test case given by Test Condition Table.

8.1.3.2 The column named as 'BioSPI function (BioAPI/BioSPI parameter check)' is referred to by the testing BSP when the BioAPI framework calls the BioSPI function. If the cells in this column is marked as 'X', it indicates that the parameters in the BioSPI function shall be the same as the parameters given through the BioAPI function.

8.1.3.3 The column 'BioAPI function' is the information that is checked by the testing application when the BioAPI framework has returned from the BioAPI function. The column 'Return value' indicates the return value of the BioAPI function and the column 'Output parameter name' indicates the name of the output parameter to be checked and the column 'Output parameter value' indicates the expected value of the output parameter.

8.1.3.4 In the abnormal test cases, if it is obvious that ISO/IEC 19784-1:2006 defines one specific error value at a test case, the return value described in the column 'Return values' shall be one specific text string for the error value defined in 19784-1:2006. However, if a specific error value is not clearly defined in ISO/IEC 19784-1:2006, the text string 'indeterminate error' is used, which indicates that one specific error value cannot be determined. If the testing application finds 'indeterminate error' in the column 'Return value', it knows that there will be some possible errors returned and checks whether the returned value matches one of the possible error values described in the definition of the corresponding BioAPI function in ISO/IEC 19784-1:2006. The actual test logic is described in each of the XML texts.

NOTE The error values of the BioAPI functions are described in Clause 8 in ISO/IEC 19784-1:2006. If there is no specific error value for a BioAPI function, the test shall pass if the return value is not BioAPI_OK.

8.1.3.5 The column 'Other conditions' indicates the conditions other than the BioSPI function or the BioAPI function.

8.1.3.6 The elements in the table irrelevant to the test case are shown by using the hyphen-minus character ("-").

8.1.4 Expressions in Test Tables

8.1.4.1 The detail expressions in the Test Tables are based on ISO/IEC 24709-1 with some exceptions as described as follows.

Table 11 — Expressions in Test Tables referenced from ISO/IEC 24709-1

No	Column of Test Table	Reference from ISO/IEC 24709-1	Test Tables		
			DIT(1)	TCT(2)	ERT(3)
1	Input parameter name	Character strings defined in ISO/IEC 24709-1 are used as is basis.	X	X	
2	Input parameter value	(a) For the parameter values that ISO/IEC 24709-1 defines specific character strings, these strings are used as is basis. (b) For the parameter values that ISO/IEC 24709-1 defines the range of values such as handle values, character strings for simplified expressions are defined in this part of ISO/IEC 24709 (shown with *italic* fonts) and used in the tables. The details of these parameter values are described in Table 12 “Simplified expressions in Test Tables”. (c) If the parameter is only relevant to optional features of the testing BSP and not essential to implementations of the BioAPI framework, it is described as “N/A”.	X	X	
3	Output parameter name	The group names defined in Clause 9.2 in ISO/IEC 24709-1 are used in the tables and additional information for the definitions in the groups is described in the Notes under the tables. (At the current version of this part of ISO/IEC 24709, only parameter groups are used for output parameters.)			X
4	Output parameter value	Same as above.			X
5	Return value	Character strings for simplified expressions are defined in this part of ISO/IEC 24709 (shown with *italic* fonts) and used in the tables. The details of these parameter values are described in Table 12 “Simplified expressions in Test Tables”.		X	X
6	Supported options in BSP Schema	Parameters of Options Mask or Operations Mask in BSP Schema are simply expressed by removing the prefix “Option” or “Operation”. (i.e. “Enroll” for Operations Mask, which is actually “OperationEnroll”.) See also Table 12 “Simplified expressions in Test Tables”.		X	

Note 1: Default Input Table

Note 2: Test Condition Table

Note 3: Expected Result Table

8.1.4.2 In this part of 24709, the descriptions of some of the parameter values or return values are simplified in order to make the table more comprehensible. The table below explains the actual meanings of the simplified expressions used in the three Test Tables.

Table 12 — Simplified expressions in Test Tables

No	Simplified names	Test Tables			Name of Column	Explanation
		DIT(1)	TCT(2)	ERT(3)		
1	OK		X	X	Return value	__BioAPI_OK
2	Error names without the prefix "__BioAPIERR_"		X	X	Return value	CTS shall add the prefix "__BioAPIERR_" to handle it.
3	indeterminate error			X	Return value	According to the BioAPI Specification (ISO/IEC 19784-1), some of the error values returned by the BioAPI Framework depend on its implementation, which means the test assertion cannot set a specific error value for the assert condition, and the expression "indeterminate error" is used to indicate such cases. (This error value is set as "0xFFFFFFFF" in XML to suggest that there is no specific error value for the test case to check the conformity to the BioAPI Specification. See 8.2.4 for more details.)
4	Element(s) in Operations Mask		X		OPERATIONS_MASK	CTS shall add the prefix "Operations" then set "true" to handle it. If there are more than one elements, all of them shall be set to true.
5	Element(s) in Options Mask		X		OPTIONS_MASK	CTS shall add the prefix "Options" then set "true" to handle it. If there are more than one elements, all of them shall be set to true.
6	Valid Uuid	X	X		Input parameter value	This value shall be a valid representation of a UUID (see. ISO/IEC24709-1 7.6)
7	Invalid Uuid		X		Input parameter value	"0" is to be used for the invalid value.
8	Valid BSPHandle	X	X		Input parameter value	This value shall be a valid value of the output parameter Newbsphandle when BSPAttach in Common Activities is invoked.
9	Invalid BSPHandle		X		Input parameter value	This value shall be created by adding 1 to "Valid BSPHandle".
10	Invalid Purpose		X		Input parameter value	This value shall be 7 meaning __BioAPI_PURPOSE_AUDIT(6) plus one.
11	Valid BIRHandle	X	X		Input parameter value	This value shall be a valid value of the output parameter that stores a BIR handle from one of Common Activities such as Enroll, Capture, Process or CreateTemplate. The actual function in Common Activities to be used depends on each test scenario. The parameter names for the relevant functions in Common Activities are as follows: - Capture function : Capturedbir - Enroll function : Newtemplate - Process function : Processedbir - CreateTemplate : Newtemplate

12	Invalid BIRHandle		X		Input parameter value	This value shall be created by adding 1 to "Valid BIRHandle".
13	Valid DBHandle	X	X		Input parameter value	This value shall be a valid value of the output parameter Dbhandle when DbOpen in Common Activities is invoked.
14	Invalid DBHandle		X		Input parameter value	This value shall be created by adding 1 to "Valid DbHandle".
15	Valid KeyValue	X	X		Input parameter value	This value shall be a valid representation of an integer in the range 0 to 4294967295.
16	Invalid KeyValue		X		Input parameter value	This value shall be any negative value.
17	Valid MarkerHandle	X	X		Input parameter value	This value shall be a valid value of the output parameter Markerhandle when DbOpen in Common Activities is invoked.
18	Invalid MarkerHandle		X		Input parameter value	This value shall be created by adding 1 to "Valid MarkerHandle".
19	Invalid PowerMode		X		Input parameter value	This valud shall be 4 meaning __BioAPI_POWER_SLEEP(3) plus one.
20	Invalid Action		X		Input parameter value	This value shall be 3 meaning __BioAPI_INSTALL_ACTION_UNINSTALL(2) plus one.
21	Invalid BSPSchema		X		Input parameter value	This value shall be created by setting "0" to BSPUuid as an invalid BSPUuid.
22	Invalid BFPSchema		X		Input parameter value	This value shall be created by setting "0" to BFPUuid as an invalid BFPUuid.

Note 1: Default Input Table

Note 2: Test Condition Table

Note 3: Expected Result Table

8.2 Descriptions of XML text

8.2.1 There are two types of XML texts in this part of ISO/IEC 24709; one is the test script for each of the BioAPI functions; another is the test script called the common activities, which are commonly used among the different test assertions. The details of XML text is described after 8.4 as the actual test assertions for the BioAPI functions. The common activities are described in 8.3.

NOTE The XML texts in this part of ISO/IEC 24709 are not guaranteed to be error free. The purpose of using XML is to describe each test case as rigorous as possible.

8.2.2 XML text for the BioAPI functions is described as follows:

8.2.2.1 It invokes one of the activities for initialisation to prepare to call the selected BioAPI function to be tested. If there is an error occurred in the common activities, the testing application creates a report to indicate an error at initialisation.

8.2.2.2 After the initialisation finishes, it prepares all the parameters of the BioAPI function to be tested. To set the values to the variables from Test Condition Table (see 8.1.2), the <input> element is used.

8.2.2.3 It calls the BioAPI function to be tested with using the <invoke> element. As a result, the BioAPI framework is invoked and it is expected to call the testing BSP or return to the testing application due to the situations given by the test case. If the BioAPI framework calls the testing BSP, it is expected that the BioAPI framework returns from the BioAPI function after the BioSPI function returns to the BioAPI framework, then the testing application does the check for the pass/fail decision.

8.2.2.4 When the BioAPI function is returned from the BioAPI framework to the testing application, the application checks whether the return value and the related parameters are correct or not by using the <input> element to get the information described in Expected Result Table (see 8.1.3). Only if all the expected results are correctly returned, the test result shall be a pass. If one or more results is not correct, the test result shall be a fail.

8.2.2.5 To terminate the test case, it calls the termination sequence by calling one of the common activities and terminates the BioAPI framework. If an error occurred in the common activities, the testing application creates a report to indicate an error has occurred.

8.2.3 XML text for the BioSPI functions are as follows:

8.2.3.1 As for the case that the BioAPI framework transfers the parameters given through the BioAPI function to the parameters in the BioSPI function, the values in all of the parameters set by the testing application and the ones set by the BioAPI framework are checked by XML text. If all the values in the parameters of the BioAPI function and the BioSPI function are the same, the testing BSP continues the test. If one or more values are not the same, it detects a fault in the BioAPI framework and the testing BSP terminates the test.

8.2.3.2 If the parameter check is successful, XML text returns a successful result to the testing BSP.

NOTE: After XML text returns to the testing BSP, it sets a return value described in Test Condition Table and returns from the BioSPI function.

8.2.4 The return value described as 'indeterminate error' in Expected Result Table (see 8.1.3) is defined as '4294967295' (0xFFFFFFFF) in the XML texts. This value should not be actually returned from the BioAPI framework according to the BioAPI Specification (ISO/IEC 19784-1:2006), and the XML texts check whether the return value from the BioAPI framework matches with it. After checking that it does not match, then the XML texts check whether the return value is one of the possible error values. (If it does match, the testing application makes a fail report.) Since it is not necessary to create different definitions for each test case in the Expected Result Table nor in the XML texts, this implementation makes both the Expected Result Table and the XML texts simple and definite.

8.3 Common activities

8.3.1 The following package contains common activities that are referenced by many test assertions specified in the remainder of this clause.

8.3.2 The names and roles of the activities contained in this package are listed as follows:

a) InitCommonActivities – Initialise the Common Activities including initialisation of global variables.

b) ChangeBSPProperty – Change parameters of OptionsMask and OperationsMask in BSP Property in preparation of a call BioAPI_Util_InstallBSP.

c) ResetBSPProperty – Reset parameters of OptionsMask and OperationsMask in BSP Property called by ChangeBSPProperty.

d) SetBSPProperty – Set parameters of OptionsMask and OperationsMask in BSP Property called by ChangeBSPProperty.

e) Initialisation – Initialise the framework under test.

f) InstallBSP – Perform the sequence from the initialisation of the framework under test to the installation of the framework-testing BSP using the function BioAPI_Util_IntallBSP.

g) EnumBSPs – Perform the sequence form the initialisation of the framework under test to enumerating the BSPs using the function BioAPI_EnumBSPs.

h) BSPLoad – Load the BSP using BioAPI_BSPLoad function after enumerating the BSP and initialising the framework under test.

i) BSPAttach – Attach the BSP using the function BioAPI_BSPAttach after initialising the framework, enumerating the BSP and loading the BSP.

j) Enroll – Enrolling using the BioAPI_Enroll after the sequence of framework initialisation and attaching BSP after loading.

k) Capture – Capture data using BioAPI_Capture function for the enrolment.

l) Process – Processing the BIR using BioAPI_Process function.

m) DbCreate – Create the database using BioAPI_DbCreate function.

n) DbOpen – Open the database using BioAPI_DbOpen function.

o) DbClose – Close the database using BioAPI_DbClose function.

p) BSPDetach – Detach the BSP using BioAPI_BSPDetach function.

q) BSPUnload – This activity will be invoked on incoming calls to the function BioSPI_BSPUnload.

r) UnInstallBSP – UnInstall the framework-testing BSP.

s) Termination – Terminate the use of the framework under test with CTS application and the BSP.

t) ExtractErrorCode – Extract the error code from the error value.

u) AttachWithSomeOptions – Attach the BSP(Some Options).

v) PrepareReferenceTemplate – PrepareReferenceTemplate for the tests of BioAPI_Enroll, BioAPI_CreateTemplate, BioAPI_VerifyMatch and BioAPI_Verify.

w) PrepareCapturedBIR – PrepareCapturedBIR for the tests of BioAPI_CreateTemplate, BioAPI_Process and BioAPI_ProcessWithAuxBIR.

x) PrepareProcessedBIR– Capture and PrepareProcessedBIR for the tests of BioAPI_VerifyMatch and BioAPI_IdentifyMatch.

```
<?xml version="1.0" encoding="UTF-8"?>
<package name="fb6ff5b0-7d5e-11de-8a39-0800200c9a66">
  <author>ISO/IEC JTC1 SC37</author>
  <description>
    This package contains several useful activities that are invoked by activities in other packages.
  </description>

  <!-- ********************** -->
  <!-- Common Input Parameters -->
  <!-- ********************** -->
  <assertion name="CommonActivities" model="frameworkTesting">
    <description>
      This assertion is Initialisation for the test script.
      It should be processed before the test script for each of the BioAPI functions.
    </description>
    <!--Parameter given by the CTS. Prefix "_ca" means Common Activities.-->
    <!--Parameters from "_ca_bspschema_bspuuid" to "_ca_bspschema_maxidentify" are the members of BSPSchema. -->
    <input name="_ca_bspschema_bspuuid"/>
    <input name="_ca_bspschema_description"/>
    <input name="_ca_bspschema_path"/>
    <input name="_ca_bspschema_specversion"/>
    <input name="_ca_bspschema_productversion"/>
    <input name="_ca_bspschema_vendor"/>
    <input name="_ca_bspschema_format_1_formatowner"/>
    <input name="_ca_bspschema_format_1_formattype"/>
    <input name="_ca_bspschema_format_2_formatowner"/>
    <input name="_ca_bspschema_format_2_formattype"/>
    <input name="_ca_bspschema_format_3_formatowner"/>
    <input name="_ca_bspschema_format_3_formattype"/>
    <input name="_ca_bspschema_format_4_formatowner"/>
    <input name="_ca_bspschema_format_4_formattype"/>
    <input name="_ca_bspschema_numsupportedformats"/>
    <input name="_ca_bspschema_typemultiple"/>
    <input name="_ca_bspschema_typefacialfeatures"/>
    <input name="_ca_bspschema_typevoice"/>
    <input name="_ca_bspschema_typefingerprint"/>
    <input name="_ca_bspschema_typeiris"/>
    <input name="_ca_bspschema_typeretina"/>
    <input name="_ca_bspschema_typehandgeometry"/>
    <input name="_ca_bspschema_typesignaturedynamics"/>
    <input name="_ca_bspschema_typekeystrokedynamics"/>
    <input name="_ca_bspschema_typelipmovement"/>
    <input name="_ca_bspschema_typethermalfaceimage"/>
    <input name="_ca_bspschema_typethermalhandimage"/>
    <input name="_ca_bspschema_typegait"/>
    <input name="_ca_bspschema_typeother"/>
    <input name="_ca_bspschema_typepassword"/>d
    <input name="_ca_bspschema_payloadpolicy"/>
    <input name="_ca_bspschema_maxpayloadsize"/>
    <input name="_ca_bspschema_defaultverifytimeout"/>
    <input name="_ca_bspschema_defaultidentifytimeout"/>
    <input name="_ca_bspschema_defaultcapturetimeout"/>
    <input name="_ca_bspschema_defaultenrolltimeout"/>
    <input name="_ca_bspschema_defaultcalibratetimeout"/>
    <input name="_ca_bspschema_maxbspdbsize"/>
    <input name="_ca_bspschema_maxidentify"/>
    <!-- Invocation of the primary activity of this assertion. -->
    <invoke activity="InitCommonActivities"/>
  </assertion>

  <!-- ********************************** -->
  <!-- Activity to init Common Activities -->
  <!-- ********************************** -->
  <activity name="InitCommonActivities">
    <!-- Common constant -->
    <!--  The return value described as 'indeterminate error' in Expected Result Table (see 8.1.3)
          is defined as '4294967295' (0xFFFFFFFF) in the XML texts. -->
    <set name="_indeterminate_error" value="4294967295"/>
  </activity>

  <!-- ****************************** -->
  <!-- Activity to Change BSP Property -->
  <!-- ****************************** -->
  <activity name="ChangeBSPProperty">
    <input name="Operationsmask"/>
    <input name="Optionsmask"/>

    <!-- Reset BSP Property -->
    <invoke activity="ResetBSPProperty"/>
    <!-- Set BSP Property -->
    <invoke activity="SetBSPProperty">
      <input name="Operationsmask" var="Operationsmask"/>
      <input name="Optionsmask" var="Optionsmask"/>
    </invoke>
  </activity>

  <!-- ***************************** -->
  <!-- Activity to Reset BSP Property -->
  <!-- ***************************** -->
  <activity name="ResetBSPProperty">
    <set name="_ca_addbspproperty_check" value="false"/>
    <!-- Reset Parameters -->
    <set name="_ca_bspschema_OperationEnableEvents" value="false"/>
    <set name="_ca_bspschema_OperationSetGUICallbacks" value="false"/>
    <set name="_ca_bspschema_OperationCapture" value="false"/>
    <set name="_ca_bspschema_OperationCreateTemplate" value="false"/>
    <set name="_ca_bspschema_OperationProcess" value="false"/>
    <set name="_ca_bspschema_OperationProcessWithAuxBIR" value="false"/>
    <set name="_ca_bspschema_OperationVerifyMatch" value="false"/>
    <set name="_ca_bspschema_OperationIdentifyMatch" value="false"/>
    <set name="_ca_bspschema_OperationEnroll" value="false"/>
    <set name="_ca_bspschema_OperationVerify" value="false"/>
    <set name="_ca_bspschema_OperationIdentify" value="false"/>
    <set name="_ca_bspschema_OperationImport" value="false"/>
    <set name="_ca_bspschema_OperationPresetIdentifyPopulation" value="false"/>
    <set name="_ca_bspschema_OperationDatabaseOperations" value="false"/>
    <set name="_ca_bspschema_OperationSetPowerMode" value="false"/>
    <set name="_ca_bspschema_OperationSetIndicatorStatus" value="false"/>
    <set name="_ca_bspschema_OperationGetIndicatorStatus" value="false"/>
    <set name="_ca_bspschema_OperationCalibrateSensor" value="false"/>
    <set name="_ca_bspschema_OperationUtilities" value="false"/>
    <set name="_ca_bspschema_OperationQueryUnits" value="false"/>
    <set name="_ca_bspschema_OperationQueryBFPs" value="false"/>
    <set name="_ca_bspschema_OperationControlUnit" value="false"/>
    <set name="_ca_bspschema_OptionRaw" value="false"/>
    <set name="_ca_bspschema_OptionQualityRaw" value="false"/>
    <set name="_ca_bspschema_OptionQualityIntermediate" value="false"/>
    <set name="_ca_bspschema_OptionQualityProcessed" value="false"/>
    <set name="_ca_bspschema_OptionAppGUI" value="false"/>
    <set name="_ca_bspschema_OptionSourcePresent" value="false"/>
    <set name="_ca_bspschema_OptionPayload" value="false"/>
    <set name="_ca_bspschema_OptionBIRSign" value="false"/>
    <set name="_ca_bspschema_OptionBIREncrypt" value="false"/>
    <set name="_ca_bspschema_OptionTemplateUpdate" value="false"/>
    <set name="_ca_bspschema_OptionAdaptation" value="false"/>
    <set name="_ca_bspschema_OptionBinning" value="false"/>
    <set name="_ca_bspschema_OptionSelfContainedDevice" value="false"/>
    <set name="_ca_bspschema_OptionMOC" value="false"/>
    <set name="_ca_bspschema_OptionSubtypeToCapture" value="false"/>
    <set name="_ca_bspschema_OptionSensorBFP" value="false"/>
    <set name="_ca_bspschema_OptionArchiveBFP" value="false"/>
    <set name="_ca_bspschema_OptionMatchingBFP" value="false"/>
    <set name="_ca_bspschema_OptionProcessingBFP" value="false"/>
    <set name="_ca_bspschema_OptionCoarseScores" value="false"/>
  </activity>

  <!-- **************************** -->
  <!-- Activity to Set BSP Property -->
  <!-- **************************** -->
  <activity name="SetBSPProperty">
    <input name="Operationsmask"/>
    <input name="Optionsmask"/>

    <set name="_ca_addbspproperty_check" value="true"/>

    <set name="_ca_bspschema_OperationEnableEvents" value="true">
      <only_if><equal_to var1="Operationsmask" value2="1"/></only_if></set>
    <set name="_ca_bspschema_OperationSetGUICallbacks" value="true">
      <only_if><equal_to var1="Operationsmask" value2="2"/></only_if></set>
    <set name="_ca_bspschema_OperationCapture" value="true">
      <only_if><equal_to var1="Operationsmask" value2="4"/></only_if></set>
    <set name="_ca_bspschema_OperationCreateTemplate" value="true">
      <only_if><equal_to var1="Operationsmask" value2="8"/></only_if></set>
    <set name="_ca_bspschema_OperationProcess" value="true">
      <only_if><equal_to var1="Operationsmask" value2="16"/></only_if></set>
    <set name="_ca_bspschema_OperationProcessWithAuxBIR" value="true">
      <only_if><equal_to var1="Operationsmask" value2="32"/></only_if></set>
    <set name="_ca_bspschema_OperationVerifyMatch" value="true">
      <only_if><equal_to var1="Operationsmask" value2="64"/></only_if></set>
    <set name="_ca_bspschema_OperationIdentifyMatch" value="true">
      <only_if><equal_to var1="Operationsmask" value2="128"/></only_if></set>
    <set name="_ca_bspschema_OperationEnroll" value="true">
      <only_if><equal_to var1="Operationsmask" value2="256"/></only_if></set>
    <set name="_ca_bspschema_OperationVerify" value="true">
      <only_if><equal_to var1="Operationsmask" value2="512"/></only_if></set>
    <set name="_ca_bspschema_OperationIdentify" value="true">
      <only_if><equal_to var1="Operationsmask" value2="1024"/></only_if></set>
    <set name="_ca_bspschema_OperationImport" value="true">
      <only_if><equal_to var1="Operationsmask" value2="2048"/></only_if></set>
    <set name="_ca_bspschema_OperationPresetIdentifyPopulation" value="true">
      <only_if><equal_to var1="Operationsmask" value2="4096"/></only_if></set>
    <set name="_ca_bspschema_OperationDatabaseOperations" value="true">
      <only_if><equal_to var1="Operationsmask" value2="8192"/></only_if></set>
    <set name="_ca_bspschema_OperationSetPowerMode" value="true">
      <only_if><equal_to var1="Operationsmask" value2="16384"/></only_if></set>
    <set name="_ca_bspschema_OperationSetIndicatorStatus" value="true">
      <only_if><equal_to var1="Operationsmask" value2="32768"/></only_if></set>
    <set name="_ca_bspschema_OperationGetIndicatorStatus" value="true">
      <only_if><equal_to var1="Operationsmask" value2="65536"/></only_if></set>
    <set name="_ca_bspschema_OperationCalibrateSensor" value="true">
      <only_if><equal_to var1="Operationsmask" value2="131072"/></only_if></set>
    <set name="_ca_bspschema_OperationUtilities" value="true">
      <only_if><equal_to var1="Operationsmask" value2="262144"/></only_if></set>
    <set name="_ca_bspschema_OperationQueryUnits" value="true">
      <only_if><equal_to var1="Operationsmask" value2="1048576"/></only_if></set>
    <set name="_ca_bspschema_OperationQueryBFPs" value="true">
      <only_if><equal_to var1="Operationsmask" value2="2097152"/></only_if></set>
    <set name="_ca_bspschema_OperationControlUnit" value="true">
      <only_if><equal_to var1="Operationsmask" value2="4194304"/></only_if></set>

    <set name="_ca_bspschema_OptionRaw" value="true">
      <only_if><equal_to var1="Optionsmask" value2="1"/></only_if></set>
    <set name="_ca_bspschema_OptionQualityRaw" value="true">
      <only_if><equal_to var1="Optionsmask" value2="2"/></only_if></set>
    <set name="_ca_bspschema_OptionQualityIntermediate" value="true">
      <only_if><equal_to var1="Optionsmask" value2="4"/></only_if></set>
    <set name="_ca_bspschema_OptionQualityProcessed" value="true">
      <only_if><equal_to var1="Optionsmask" value2="8"/></only_if></set>
    <set name="_ca_bspschema_OptionAppGUI" value="true">
      <only_if><equal_to var1="Optionsmask" value2="16"/></only_if></set>
    <set name="_ca_bspschema_OptionSourcePresent" value="true">
      <only_if><equal_to var1="Optionsmask" value2="64"/></only_if></set>
    <set name="_ca_bspschema_OptionPayload" value="true">
      <only_if><equal_to var1="Optionsmask" value2="128"/></only_if></set>
    <set name="_ca_bspschema_OptionBIRSign" value="true">
      <only_if><equal_to var1="Optionsmask" value2="256"/></only_if></set>
    <set name="_ca_bspschema_OptionBIREncrypt" value="true">
      <only_if><equal_to var1="Optionsmask" value2="512"/></only_if></set>
    <set name="_ca_bspschema_OptionTemplateUpdate" value="true">
      <only_if><equal_to var1="Optionsmask" value2="1024"/></only_if></set>
    <set name="_ca_bspschema_OptionAdaptation" value="true">
      <only_if><equal_to var1="Optionsmask" value2="2048"/></only_if></set>
    <set name="_ca_bspschema_OptionBinning" value="true">
      <only_if><equal_to var1="Optionsmask" value2="4096"/></only_if></set>
    <set name="_ca_bspschema_OptionSelfContainedDevice" value="true">
      <only_if><equal_to var1="Optionsmask" value2="8192"/></only_if></set>
    <set name="_ca_bspschema_OptionMOC" value="true">
      <only_if><equal_to var1="Optionsmask" value2="16384"/></only_if></set>
    <set name="_ca_bspschema_OptionSubtypeToCapture" value="true">
      <only_if><equal_to var1="Optionsmask" value2="32768"/></only_if></set>
    <set name="_ca_bspschema_OptionSensorBFP" value="true">
      <only_if><equal_to var1="Optionsmask" value2="65536"/></only_if></set>
    <set name="_ca_bspschema_OptionArchiveBFP" value="true">
      <only_if><equal_to var1="Optionsmask" value2="131072"/></only_if></set>
    <set name="_ca_bspschema_OptionMatchingBFP" value="true">
      <only_if><equal_to var1="Optionsmask" value2="262144"/></only_if></set>
    <set name="_ca_bspschema_OptionProcessingBFP" value="true">
      <only_if><equal_to var1="Optionsmask" value2="524288"/></only_if></set>
    <set name="_ca_bspschema_OptionCoarseScores" value="true">
      <only_if><equal_to var1="Optionsmask" value2="1048576"/></only_if></set>
  </activity>

  <!-- ****************************** -->
  <!-- Activity to invoke BioAPI_Init -->
  <!-- ****************************** -->
  <activity name="Initialisation">
    <!-- Invoke the function BioAPI_Init -->
    <invoke function="BioAPI_Init">
      <input name="Version"  value="32"/>
      <return setvar="return"/>
    </invoke>
    <assert_condition response_if_false="undecided" break_if_false="true">
      <description>
        The function BioAPI_Init has returned BioAPI_OK.
      </description>
      <equal_to var1="return" var2="__BioAPI_OK"/>
    </assert_condition>
  </activity>

  <!-- ************************************************************* -->
  <!-- Activity to invoke BioAPI_Util_InstallBSP  with initialisation  -->
  <!-- ************************************************************* -->
  <activity name="InstallBSP">
    <input name="Operationsmask"/>
    <input name="Optionsmask"/>

    <!-- Invoke the activity Initialisation. -->
    <invoke activity="Initialisation" break_on_break="true"/>
    <!-- Invoke the activity SetBSPProperty to set OperationsMask and OptionsMask. -->
    <invoke activity="ChangeBSPProperty" break_on_break="true">
      <only_if>
        <not var="_ca_addbspproperty_check"/>
      </only_if>
      <input name="Operationsmask" var="Operationsmask"/>
      <input name="Optionsmask" var="Optionsmask"/>
    </invoke>

    <!-- Invoke the function BioAPI_Util_InstallBSP -->
    <invoke function="BioAPI_Util_InstallBSP">
      <input name="Action" var="__BioAPI_INSTALL_ACTION_INSTALL"/>
      <input name="no_Error" value="false"/>
      <input name="BSPUuid" var="_ca_bspschema_bspuuid"/>
      <input name="Description" var="_ca_bspschema_description"/>
      <input name="Path" var="_ca_bspschema_path"/>
      <input name="SpecVersion" var="_ca_bspschema_specversion"/>
      <input name="ProductVersion" var="_ca_bspschema_productversion"/>
      <input name="Vendor" var="_ca_bspschema_vendor"/>
      <input name="Format_1_FormatOwner" var="_ca_bspschema_format_1_formatowner"/>
      <input name="Format_1_FormatType" var="_ca_bspschema_format_1_formattype"/>
      <input name="Format_2_FormatOwner" var="_ca_bspschema_format_2_formatowner"/>
      <input name="Format_2_FormatType" var="_ca_bspschema_format_2_formattype"/>
      <input name="Format_3_FormatOwner" var="_ca_bspschema_format_3_formatowner"/>
      <input name="Format_3_FormatType" var="_ca_bspschema_format_3_formattype"/>
      <input name="Format_4_FormatOwner" var="_ca_bspschema_format_4_formatowner"/>
      <input name="Format_4_FormatType" var="_ca_bspschema_format_4_formattype"/>
      <input name="NumSupportedFormats" var="_ca_bspschema_numsupportedformats"/>
      <input name="TypeMultiple" var="_ca_bspschema_typemultiple"/>
      <input name="TypeFacialFeatures" var="_ca_bspschema_typefacialfeatures"/>
      <input name="TypeVoice" var="_ca_bspschema_typevoice"/>
      <input name="TypeFingerprint" var="_ca_bspschema_typefingerprint"/>
      <input name="TypeIris" var="_ca_bspschema_typeiris"/>
      <input name="TypeRetina" var="_ca_bspschema_typeretina"/>
      <input name="TypeHandGeometry" var="_ca_bspschema_typehandgeometry"/>
      <input name="TypeSignatureDynamics" var="_ca_bspschema_typesignaturedynamics"/>
      <input name="TypeKeystrokeDynamics" var="_ca_bspschema_typekeystrokedynamics"/>
      <input name="TypeLipMovement" var="_ca_bspschema_typelipmovement"/>
      <input name="TypeThermalFaceImage" var="_ca_bspschema_typethermalfaceimage"/>
      <input name="TypeThermalHandImage" var="_ca_bspschema_typethermalhandimage"/>
      <input name="TypeGait" var="_ca_bspschema_typegait"/>
      <input name="TypeOther" var="_ca_bspschema_typeother"/>
      <input name="TypePassword" var="_ca_bspschema_typepassword"/>
      <input name="OperationEnableEvents" var="_ca_bspschema_operationenableevents"/>
      <input name="OperationSetGUICallbacks" var="_ca_bspschema_operationsetguicallbacks"/>
      <input name="OperationCapture" var="_ca_bspschema_operationcapture"/>
      <input name="OperationCreateTemplate" var="_ca_bspschema_operationcreatetemplate"/>
      <input name="OperationProcess" var="_ca_bspschema_operationprocess"/>
      <input name="OperationProcessWithAuxBIR" var="_ca_bspschema_operationprocesswithauxbir"/>
      <input name="OperationVerifyMatch" var="_ca_bspschema_operationverifymatch"/>
      <input name="OperationIdentifyMatch" var="_ca_bspschema_operationidentifymatch"/>
      <input name="OperationEnroll" var="_ca_bspschema_operationenroll"/>
      <input name="OperationVerify" var="_ca_bspschema_operationverify"/>
      <input name="OperationIdentify" var="_ca_bspschema_operationidentify"/>
      <input name="OperationImport" var="_ca_bspschema_operationimport"/>
      <input name="OperationPresetIdentifyPopulation" var="_ca_bspschema_operationpresetidentifypopulation"/>
      <input name="OperationDatabaseOperations" var="_ca_bspschema_operationdatabaseoperations"/>
      <input name="OperationSetPowerMode" var="_ca_bspschema_operationsetpowermode"/>
      <input name="OperationSetIndicatorStatus" var="_ca_bspschema_operationsetindicatorstatus"/>
      <input name="OperationGetIndicatorStatus" var="_ca_bspschema_operationgetindicatorstatus"/>
      <input name="OperationCalibrateSensor" var="_ca_bspschema_operationcalibratesensor"/>
      <input name="OperationUtilities" var="_ca_bspschema_operationutilities"/>
      <input name="OperationQueryUnits" var="_ca_bspschema_operationqueryunits"/>
      <input name="OperationQueryBFPs" var="_ca_bspschema_operationquerybfps"/>
      <input name="OperationControlUnit" var="_ca_bspschema_operationcontrolunit"/>
      <input name="OptionRaw" var="_ca_bspschema_optionraw"/>
      <input name="OptionQualityRaw" var="_ca_bspschema_optionqualityraw"/>
      <input name="OptionQualityIntermediate" var="_ca_bspschema_optionqualityintermediate"/>
      <input name="OptionQualityProcessed" var="_ca_bspschema_optionqualityprocessed"/>
      <input name="OptionAppGUI" var="_ca_bspschema_optionappgui"/>
      <input name="OptionSourcePresent" var="_ca_bspschema_optionsourcepresent"/>
      <input name="OptionPayload" var="_ca_bspschema_optionpayload"/>
      <input name="OptionBIRSign" var="_ca_bspschema_optionbirsign"/>
      <input name="OptionBIREncrypt" var="_ca_bspschema_optionbirencrypt"/>
      <input name="OptionTemplateUpdate" var="_ca_bspschema_optiontemplateupdate"/>
      <input name="OptionAdaptation" var="_ca_bspschema_optionadaptation"/>
      <input name="OptionBinning" var="_ca_bspschema_optionbinning"/>
      <input name="OptionSelfContainedDevice" var="_ca_bspschema_optionselfcontaineddevice"/>
      <input name="OptionMOC" var="_ca_bspschema_optionmoc"/>
      <input name="OptionSubtypeToCapture" var="_ca_bspschema_optionsubtypetocapture"/>
      <input name="OptionSensorBFP" var="_ca_bspschema_optionsensorbfp"/>
      <input name="OptionArchiveBFP" var="_ca_bspschema_optionarchivebfp"/>
      <input name="OptionMatchingBFP" var="_ca_bspschema_optionmatchingbfp"/>
      <input name="OptionProcessingBFP" var="_ca_bspschema_optionprocessingbfp"/>
      <input name="OptionCoarseScores" var="_ca_bspschema_optioncoarsescores"/>

      <input name="PayloadPolicy" var="_ca_bspschema_payloadpolicy"/>
      <input name="MaxPayloadSize" var="_ca_bspschema_maxpayloadsize"/>
      <input name="DefaultVerifyTimeout" var="_ca_bspschema_defaultverifytimeout"/>
      <input name="DefaultIdentifyTimeout" var="_ca_bspschema_defaultidentifytimeout"/>
      <input name="DefaultCaptureTimeout" var="_ca_bspschema_defaultcapturetimeout"/>
      <input name="DefaultEnrollTimeout" var="_ca_bspschema_defaultenrolltimeout"/>
      <input name="DefaultCalibrateTimeout" var="_ca_bspschema_defaultcalibratetimeout"/>
      <input name="MaxBSPDbSize" var="_ca_bspschema_maxbspdbsize"/>
      <input name="MaxIdentify" var="_ca_bspschema_maxidentify"/>
      <output name="Error" setvar="error"/>
      <return setvar="return"/>
    </invoke>
    <assert_condition response_if_false="undecided" break_if_false="true">
      <description>
        The function BioAPI_Util_InstallBSP has returned BioAPI_OK.
      </description>
      <equal_to var1="return" var2="__BioAPI_OK"/>
    </assert_condition>
  </activity>

  <!-- ****************************************************** -->
  <!-- Activity to invoke BioAPI_EnumBSPs with initialisation -->
  <!-- ****************************************************** -->
  <activity name="EnumBSPs">
    <input name="Operationsmask"/>
    <input name="Optionsmask"/>

    <!-- Invoke the activity InstallBSP to call the function,
         BioAPI_Util_InstallBSP after the initialisation for the Framework -->
    <invoke activity="InstallBSP" break_on_break="true">
      <input name="Operationsmask" var="Operationsmask"/>
      <input name="Optionsmask" var="Optionsmask"/>
    </invoke>
    <!-- Invoke the function BioAPI_EnumBSPs -->
    <invoke function="BioAPI_EnumBSPs">
      <input name="no_BSPSchemaArray" value="false"/>
      <input name="no_NumberOfElements" value="false"/>
      <return setvar="return"/>
    </invoke>
    <assert_condition response_if_false="undecided" break_if_false="true">
```

```
      <description>
        The function BioAPI_EnumBSPs has returned BioAPI_OK.
      </description>
      <equal_to var1="return" var2="__BioAPI_OK"/>
    </assert_condition>
  </activity>

  <!-- ***************************************************** -->
  <!-- Activity to invoke BioAPI_BSPLoad with initialisation -->
  <!-- ***************************************************** -->
  <activity name="BSPLoad">
    <input name="Operationsmask"/>
    <input name="Optionsmask"/>
    <input name="Bspuuid"/>

    <!-- Invoke the function BioAPI_EnumBSPs after the initialisation for the Framework -->
    <invoke activity="EnumBSPs" break_on_break="true">
      <input name="Operationsmask" var="Operationsmask"/>
      <input name="Optionsmask" var="Optionsmask"/>
    </invoke>
    <!-- Invoke the function BioAPI_BSPLoad -->
    <invoke function="BioAPI_BSPLoad">
      <input name="BSPUuid" var="Bspuuid"/>
      <input name="AppNotifyCallback" value="*"/>
      <input name="AppNotifyCallbackCtx" value="*"/>
      <return setvar="return"/>
    </invoke>
    <assert_condition response_if_false="undecided" break_if_false="true">
      <description>
        The function BioAPI_BSPLoad has returned BioAPI_OK.
      </description>
      <equal_to var1="return" var2="__BioAPI_OK"/>
    </assert_condition>
  </activity>

  <!-- ************************************************************************ -->
  <!-- Activity to invoke BioAPI_BSPAttach with initialisation of the Framework -->
  <!-- ************************************************************************ -->
  <activity name="BSPAttach">
    <input name="Operationsmask"/>
    <input name="Optionsmask"/>
    <input name="Bspuuid"/>
    <output name="Newbsphandle"/>

    <!-- Invoke the function BioAPI_BSPLoad -->
    <invoke activity="BSPLoad" break_on_break="true">
      <input name="Operationsmask" var="Operationsmask"/>
      <input name="Optionsmask" var="Optionsmask"/>
      <input name="Bspuuid" var="Bspuuid"/>
    </invoke>
    <!-- Invoke the function BioAPI_BSPAttach -->
    <invoke function="BioAPI_BSPAttach">
      <input name="BSPUuid" var="Bspuuid"/>
      <input name="Version" value="32"/>
      <input name="NumUnits" value="0"/>
      <output name="NewBSPHandle" setvar="Newbsphandle"/>
      <return setvar="return"/>
    </invoke>
    <assert_condition response_if_false="undecided" break_if_false="true">
      <description>
        The function BioAPI_BSPAttach has returned BioAPI_OK.
      </description>
      <equal_to var1="return" var2="__BioAPI_OK"/>
    </assert_condition>
  </activity>

  <!-- ******************************** -->
  <!-- Activity to invoke BioAPI_Enroll -->
  <!-- ******************************** -->
  <activity name="Enroll">
    <input name="Bsphandle"/>
    <input name="Purpose"/>
    <input name="Outputformatowner"/>
    <input name="Outputformattype"/>
    <output name="Newtemplate"/>

    <!-- Invoke the function BioAPI_Enroll -->
    <invoke function="BioAPI_Enroll">
      <input name="BspHandle" var="Bsphandle"/>
      <input name="Purpose" value="Purpose"/>
      <input name="Left" value="false"/>
      <input name="Right" value="false"/>
      <input name="Thumb" value="false"/>
      <input name="PointerFinger" value="false"/>
      <input name="MiddleFinger" value="false"/>
      <input name="RingFinger" value="false"/>
      <input name="LittleFinger" value="false"/>
      <input name="Multiple" value="false"/>
      <input name="OutputformatOwner" var="Outputformatowner"/>
      <input name="OutputformatType" var="Outputformattype"/>
      <input name="Referencetemplate_Form" value="0"/>
      <input name="no_NewTemplate" value="false"/>
      <input name="Payload" value=""/>
      <input name="Timeout" value="-1"/>
      <input name="no_AuditData" value="true"/>
      <input name="no_TemplateUUID" value="true"/>
      <output name="NewTemplate" setvar="Newtemplate"/>
      <return setvar="return"/>
    </invoke>
    <assert_condition response_if_false="undecided" break_if_false="true">
      <description>
        The function BioAPI_Enroll has returned BioAPI_OK.
      </description>
      <equal_to var1="return" var2="__BioAPI_OK"/>
    </assert_condition>
  </activity>

  <!-- ********************************* -->
  <!-- Activity to invoke BioAPI_Capture -->
  <!-- ********************************* -->
  <activity name="Capture">
    <input name="Bsphandle"/>
    <input name="Purpose"/>
    <input name="Outputformatowner"/>
    <input name="Outputformattype"/>
    <output name="Capturedbirhandle"/>

    <invoke function="BioAPI_Capture">
      <input name="BSPHandle" var="Bsphandle"/>
      <input name="Purpose" var="Purpose"/>
      <input name="Left" value="false"/>
      <input name="Right" value="false"/>
      <input name="Thumb" value="false"/>
      <input name="PointerFinger" value="false"/>
      <input name="MiddleFinger" value="false"/>
      <input name="RingFinger" value="false"/>
      <input name="LittleFinger" value="false"/>
      <input name="Multiple" value="false"/>
      <input name="OutputformatOwner" var="Outputformatowner"/>
      <input name="OutputformatType" var="Outputformattype"/>
      <input name="no_CapturedBIR" value="false"/>
      <input name="Timeout" value="-1"/>
      <input name="no_AuditData" value="true"/>
      <output name="CapturedBIR" setvar="Capturedbirhandle"/>
      <return setvar="return"/>
    </invoke>
    <assert_condition response_if_false="undecided" break_if_false="true">
      <description>
        The function BioAPI_Capture has returned BioAPI_OK.
      </description>
      <equal_to var1="return" var2="__BioAPI_OK"/>
    </assert_condition>
  </activity>

  <!-- ********************************* -->
  <!-- Activity to invoke BioAPI_Process -->
  <!-- ********************************* -->
  <activity name="Process">
    <input name="Bsphandle"/>
    <input name="Capturedbir_form"/>
    <input name="Capturedbir_birhandle"/>
    <input name="Outputformatowner"/>
    <input name="Outputformattype"/>
    <output name="Processedbirhandle"/>

    <invoke function="BioAPI_Process">
      <input name="BSPHandle" var="Bsphandle"/>
      <input name="CapturedBIR_Form" var="Capturedbir_form"/>
      <input name="CapturedBIR_BIRHandle" var="Capturedbir_birhandle"/>
      <input name="OutputFormatOwner" var="Outputformatowner"/>
      <input name="OutputFormatType" var="Outputformattype"/>
      <input name="no_ProcessedBIR" var="false"/>
      <output name="ProcessedBIR" setvar="Processedbirhandle"/>
      <return setvar="return"/>
    </invoke>
    <assert_condition response_if_false="undecided" break_if_false="true">
      <description>
        The function BioAPI_Process has returned BioAPI_OK.
      </description>
      <equal_to var1="return" var2="__BioAPI_OK"/>
    </assert_condition>
  </activity>

  <!-- ********************************** -->
```

```
<!-- Activity to invoke BioAPI_DbCreate -->
<!-- ********************************* -->
<activity name="DbCreate">
  <input name="Bsphandle"/>
  <input name="Dbuuid"/>
  <output name="Dbhandle"/>

  <invoke function="BioAPI_DbCreate">
    <input name="BSPHandle" var="Bsphandle"/>
    <input name="DbUuid" var="Dbuuid"/>
    <input name="Numberofrecords" value="10"/>
    <input name="ReadAccess" value="true"/>
    <input name="WriteAccess" value="true"/>
    <input name="no_DbHandle" var="false"/>
    <input name="no_DbUuid" var="false"/>
    <output name="DbHandle" setvar="Dbhandle"/>
    <return setvar="return"/>
  </invoke>
  <assert_condition response_if_false="undecided" break_if_false="true">
    <description>
      The function BioAPI_DbCreate has returned BioAPI_OK.
    </description>
    <equal_to var1="return" var2="__BioAPI_OK"/>
  </assert_condition>
</activity>

<!-- ******************************** -->
<!-- Activity to invoke BioAPI_DbOpen -->
<!-- ******************************** -->
<activity name="DbOpen">
  <input name="Bsphandle"/>
  <input name="Dbuuid"/>
  <output name="Dbhandle"/>
  <output name="Markerhandle"/>

  <invoke function="BioAPI_DbOpen">
    <input name="BSPHandle" var="Bsphandle"/>
    <input name="DbUuid" value="Dbuuid"/>
    <input name="ReadAccess" var="true"/>
    <input name="WriteAccess" var="true"/>
    <input name="no_DbHandle" var="false"/>
    <input name="no_MarkerHandle" var="false"/>
    <output name="DbHandle" setvar="Dbhandle"/>
    <output name="MarkerHandle" setvar="Markerhandle"/>
    <return setvar="return"/>
  </invoke>
  <assert_condition response_if_false="undecided" break_if_false="true">
    <description>
      The function BioAPI_DbOpen has returned BioAPI_OK.
    </description>
    <equal_to var1="return" var2="__BioAPI_OK"/>
  </assert_condition>
</activity>

<!-- ********************************* -->
<!-- Activity to invoke BioAPI_DbClose -->
<!-- ********************************* -->
<activity name="DbClose">
  <input name="Bsphandle"/>
  <input name="Dbhandle"/>

  <invoke function="BioAPI_DbClose">
    <input name="BSPHandle" var="Bsphandle"/>
    <input name="DbHandle" var="Dbhandle"/>
    <return setvar="return"/>
  </invoke>
  <assert_condition response_if_false="undecided" break_if_false="true">
    <description>
      The function BioAPI_DbClose has returned BioAPI_OK.
    </description>
    <equal_to var1="return" var2="__BioAPI_OK"/>
  </assert_condition>
</activity>

<!-- ********************************************************************** -->
<!-- Activity to invoke BioAPI_BSPDetach  with termination of the Framework -->
<!-- ********************************************************************** -->
<activity name="BSPDetach">
  <input name="Bsphandle"/>
  <input name="Bspuuid"/>

  <!-- Invoke the function BioAPI_BSPDetach -->
  <invoke function="BioAPI_BSPDetach">
    <input name="BSPHandle" var="Bsphandle"/>
    <return setvar="return"/>
  </invoke>
  <assert_condition response_if_false="undecided" break_if_false="true">
```

```
    <description>
      The function BioAPI_BSPDetach has returned BioAPI_OK.
    </description>
    <equal_to var1="return" var2="__BioAPI_OK"/>
  </assert_condition>
  <!-- Invoke the function BioAPI_BSPUnload -->
  <invoke activity="BSPUnload" break_on_break="true">
    <input name="Bspuuid" var="Bspuuid"/>
  </invoke>
</activity>

<!-- *********************************************************************** -->
<!-- Activity to invoke BioAPI_BSPUnload  with termination of the Framework -->
<!-- *********************************************************************** -->
<activity name="BSPUnload">
  <input name="Bspuuid"/>

  <!-- Invoke the function BioAPI_BSPUnload -->
  <invoke function="BioAPI_BSPUnload">
    <input name="BSPUuid" var="bspUuid"/>
    <input name="AppNotifyCallback" value="*"/>
    <input name="AppNotifyCallbackCtx" value="*"/>
    <return setvar="return"/>
  </invoke>
  <assert_condition response_if_false="undecided" break_if_false="true">
    <description>
      The function BioAPI_BSPUnload has returned BioAPI_OK.
    </description>
    <equal_to var1="return" var2="__BioAPI_OK"/>
  </assert_condition>
  <!-- Invoke UnInstallBSP activity -->
  <invoke activity="UnInstallBSP" break_on_break="true">
  </invoke>
   <!-- Invoke Termination activity -->
  <invoke activity="Termination" break_on_break="true">
  </invoke>
</activity>

<!-- *************************************************************** -->
<!-- Invoke UninstallBSP activity with termination of the framework -->
<!-- *************************************************************** -->
<activity name="UnInstallBSP">
  <!-- Invoke the function BioAPI_Util_InstallBSP -->
  <invoke function="BioAPI_Util_InstallBSP">
    <input name="Action"  var="__BioAPI_INSTALL_ACTION_UNINSTALL"/>
    <input name="no_Error"  value="false"/>
    <input name="BSPUuid" var="_ca_bspschema_bspuuid"/>
    <input name="Description" var="_ca_bspschema_description"/>
    <input name="Path" var="_ca_bspschema_path"/>
    <input name="SpecVersion" var="_ca_bspschema_specversion"/>
    <input name="ProductVersion" var="_ca_bspschema_productversion"/>
    <input name="Vendor" var="_ca_bspschema_vendor"/>
    <input name="Format_1_FormatOwner" var="_ca_bspschema_format_1_formatowner"/>
    <input name="Format_1_FormatType" var="_ca_bspschema_format_1_formattype"/>
    <input name="Format_2_FormatOwner" var="_ca_bspschema_format_2_formatowner"/>
    <input name="Format_2_FormatType" var="_ca_bspschema_format_2_formattype"/>
    <input name="Format_3_FormatOwner" var="_ca_bspschema_format_3_formatowner"/>
    <input name="Format_3_FormatType" var="_ca_bspschema_format_3_formattype"/>
    <input name="Format_4_FormatOwner" var="_ca_bspschema_format_4_formatowner"/>
    <input name="Format_4_FormatType" var="_ca_bspschema_format_4_formattype"/>
    <input name="NumSupportedFormats" var="_ca_bspschema_numsupportedformats"/>
    <input name="TypeMultiple" var="_ca_bspschema_typemultiple"/>
    <input name="TypeFacialFeatures" var="_ca_bspschema_typefacialfeatures"/>
    <input name="TypeVoice" var="_ca_bspschema_typevoice"/>
    <input name="TypeFingerprint" var="_ca_bspschema_typefingerprint"/>
    <input name="TypeIris" var="_ca_bspschema_typeiris"/>
    <input name="TypeRetina" var="_ca_bspschema_typeretina"/>
    <input name="TypeHandGeometry" var="_ca_bspschema_typehandgeometry"/>
    <input name="TypeSignatureDynamics" var="_ca_bspschema_typesignaturedynamics"/>
    <input name="TypeKeystrokeDynamics" var="_ca_bspschema_typekeystrokedynamics"/>
    <input name="TypeLipMovement" var="_ca_bspschema_typelipmovement"/>
    <input name="TypeThermalFaceImage" var="_ca_bspschema_typethermalfaceimage"/>
    <input name="TypeThermalHandImage" var="_ca_bspschema_typethermalhandimage"/>
    <input name="TypeGait" var="_ca_bspschema_typegait"/>
    <input name="TypeOther" var="_ca_bspschema_typeother"/>
    <input name="TypePassword" var="_ca_bspschema_typepassword"/>
    <input name="OperationEnableEvents" var="_ca_bspschema_operationenableevents"/>
    <input name="OperationSetGUICallbacks" var="_ca_bspschema_operationsetguicallbacks"/>
    <input name="OperationCapture" var="_ca_bspschema_operationcapture"/>
    <input name="OperationCreateTemplate" var="_ca_bspschema_operationcreatetemplate"/>
    <input name="OperationProcess" var="_ca_bspschema_operationprocess"/>
    <input name="OperationProcessWithAuxBIR" var="_ca_bspschema_operationprocesswithauxbir"/>
    <input name="OperationVerifyMatch" var="_ca_bspschema_operationverifymatch"/>
    <input name="OperationIdentifyMatch" var="_ca_bspschema_operationidentifymatch"/>
    <input name="OperationEnroll" var="_ca_bspschema_operationenroll"/>
    <input name="OperationVerify" var="_ca_bspschema_operationverify"/>
    <input name="OperationIdentify" var="_ca_bspschema_operationidentify"/>
```

```
      <input name="OperationImport" var="_ca_bspschema_operationimport"/>
      <input name="OperationPresetIdentifyPopulation" var="_ca_bspschema_operationpresetidentifypopulation"/>
      <input name="OperationDatabaseOperations" var="_ca_bspschema_operationdatabaseoperations"/>
      <input name="OperationSetPowerMode" var="_ca_bspschema_operationsetpowermode"/>
      <input name="OperationSetIndicatorStatus" var="_ca_bspschema_operationsetindicatorstatus"/>
      <input name="OperationGetIndicatorStatus" var="_ca_bspschema_operationgetindicatorstatus"/>
      <input name="OperationCalibrateSensor" var="_ca_bspschema_operationcalibratesensor"/>
      <input name="OperationUtilities" var="_ca_bspschema_operationutilities"/>
      <input name="OperationQueryUnits" var="_ca_bspschema_operationqueryunits"/>
      <input name="OperationQueryBFPs" var="_ca_bspschema_operationquerybfps"/>
      <input name="OperationControlUnit" var="_ca_bspschema_operationcontrolunit"/>
      <input name="OptionRaw" var="_ca_bspschema_optionraw"/>
      <input name="OptionQualityRaw" var="_ca_bspschema_optionqualityraw"/>
      <input name="OptionQualityIntermediate" var="_ca_bspschema_optionqualityintermediate"/>
      <input name="OptionQualityProcessed" var="_ca_bspschema_optionqualityprocessed"/>
      <input name="OptionAppGUI" var="_ca_bspschema_optionappgui"/>
      <input name="OptionSourcePresent" var="_ca_bspschema_optionsourcepresent"/>
      <input name="OptionPayload" var="_ca_bspschema_optionpayload"/>
      <input name="OptionBIRSign" var="_ca_bspschema_optionbirsign"/>
      <input name="OptionBIREncrypt" var="_ca_bspschema_optionbirencrypt"/>
      <input name="OptionTemplateUpdate" var="_ca_bspschema_optiontemplateupdate"/>
      <input name="OptionAdaptation" var="_ca_bspschema_optionadaptation"/>
      <input name="OptionBinning" var="_ca_bspschema_optionbinning"/>
      <input name="OptionSelfContainedDevice" var="_ca_bspschema_optionselfcontaineddevice"/>
      <input name="OptionMOC" var="_ca_bspschema_optionmoc"/>
      <input name="OptionSubtypeToCapture" var="_ca_bspschema_optionsubtypetocapture"/>
      <input name="OptionSensorBFP" var="_ca_bspschema_optionsensorbfp"/>
      <input name="OptionArchiveBFP" var="_ca_bspschema_optionarchivebfp"/>
      <input name="OptionMatchingBFP" var="_ca_bspschema_optionmatchingbfp"/>
      <input name="OptionProcessingBFP" var="_ca_bspschema_optionprocessingbfp"/>
      <input name="OptionCoarseScores" var="_ca_bspschema_optioncoarsescores"/>

      <input name="PayloadPolicy" var="_ca_bspschema_payloadpolicy"/>
      <input name="MaxPayloadSize" var="_ca_bspschema_maxpayloadsize"/>
      <input name="DefaultVerifyTimeout" var="_ca_bspschema_defaultverifytimeout"/>
      <input name="DefaultIdentifyTimeout" var="_ca_bspschema_defaultidentifytimeout"/>
      <input name="DefaultCaptureTimeout" var="_ca_bspschema_defaultcapturetimeout"/>
      <input name="DefaultEnrollTimeout" var="_ca_bspschema_defaultenrolltimeout"/>
      <input name="DefaultCalibrateTimeout" var="_ca_bspschema_defaultcalibratetimeout"/>
      <input name="MaxBSPDbSize" var="_ca_bspschema_maxbspdbsize"/>
      <input name="MaxIdentify" var="_ca_bspschema_maxidentify"/>
      <output name="ErrorCode" setvar="errorcode"/>
      <output name="ErrorString" setvar="errorstring"/>
      <return setvar="return"/>
    </invoke>
    <assert_condition response_if_false="undecided" break_if_false="true">
      <description>
        The function BioAPI_Util_InstallBSP has returned BioAPI_OK.
      </description>
      <equal_to var1="return" var2="__BioAPI_OK"/>
    </assert_condition>
  </activity>

  <!-- ********************************** -->
  <!-- Activity to invoke BioAPI_Terminate -->
  <!-- ********************************** -->
  <activity name="Termination">
    <!-- Invoke the function BioAPI_Terminate -->
    <invoke function="BioAPI_Terminate">
      <return setvar="return"/>
    </invoke>
    <assert_condition response_if_false="undecided" break_if_false="true">
      <description>
        The function BioAPI_Terminate has returned BioAPI_OK.
      </description>
      <equal_to var1="return" var2="__BioAPI_OK"/>
    </assert_condition>
  </activity>

  <!-- ***************************************************** -->
  <!-- Activity to extract the error code from the error value -->
  <!-- ***************************************************** -->
  <activity name="ExtractErrorCode">
    <output name="Returnvalue"/><!-- input/output -->

    <subtract name="Returnvalue" var="__BioAPI_BSP_ERROR">
      <only_if>
        <and>
          <not_equal_to var1="Returnvalue" var2="_indeterminate_error"/>
          <greater_than var1="Returnvalue" var2="__BioAPI_BSP_ERROR"/>
          <less_than var1="Returnvalue" var2="__BioAPI_UNIT_ERROR"/>
        </and>
      </only_if>
    </subtract>
    <subtract name="Returnvalue" var="__BioAPI_UNIT_ERROR">
      <only_if>
        <not_equal_to var1="Returnvalue" var2="_indeterminate_error"/>
        <greater_than var1="Returnvalue" var2="__BioAPI_UNIT_ERROR"/>
      </only_if>
    </subtract>
  </activity>

  <!-- ******************************** -->
  <!-- Activity to Attach(Some Options) -->
  <!-- ******************************** -->
  <activity name="AttachWithSomeOptions">
    <input name="Operationsmask1"/>
    <input name="Optionmask1"/>
    <input name="Operationsmask2"/>
    <input name="Optionmask2"/>
    <input name="Operationsmask3"/>
    <input name="Optionmask3"/>
    <input name="Operationsmask4"/>
    <input name="Optionmask4"/>
    <input name="Bspuuid"/>
    <output name="Bsphandle"/>

    <!-- Invoke the activity Set BSP Property. -->
    <invoke activity="ResetBSPProperty" break_on_break="true"/>
    <invoke activity="SetBSPProperty" break_on_break="true">
      <input name="Operationsmask" var="Operationsmask1"/>
      <input name="Optionsmask" var="Optionmask1"/>
    </invoke>
    <invoke activity="SetBSPProperty" break_on_break="true">
      <input name="Operationsmask" var="Operationsmask2"/>
      <input name="Optionsmask" var="Optionmask2"/>
    </invoke>
    <invoke activity="SetBSPProperty" break_on_break="true">
      <input name="Operationsmask" var="Operationsmask3"/>
      <input name="Optionsmask" var="Optionmask3"/>
    </invoke>
    <invoke activity="BSPAttach" break_on_break="true">
      <input name="Operationsmask" var="Operationsmask4"/>
      <input name="Optionsmask" var="Optionmask4"/>
      <input name="Bspuuid" var="Bspuuid"/>
      <output name="Newbsphandle" setvar="Bsphandle"/>
    </invoke>
  </activity>

  <!-- ************************************* -->
  <!-- Activity to Prepare ReferenceTemplate -->
  <!-- ************************************* -->
  <activity name="PrepareReferenceTemplate">
    <input name="Bsphandle"/>
    <input name="Outputformatowner"/>
    <input name="Outputformattype"/>
    <output name="Form"/>
    <output name="Birhandle"/>

    <invoke activity="Enroll" package="fb6ff5b0-7d5e-11de-8a39-0800200c9a66" break_on_break="true">
      <input name="Bsphandle" var="Bsphandle"/>
      <input name="Purpose" value="__BioAPI_PURPOSE_ENROLL"/>
      <input name="Outputformatowner" value="Outputformatowner"/>
      <input name="Outputformattype" value="Outputformattype"/>
      <output name="Newtemplate" setvar="Birhandle"/>
    </invoke>

    <set name="Form" value="2"/>
  </activity>

  <!-- ****************** -->
  <!-- Activity to Capture -->
  <!-- ****************** -->
  <activity name="PrepareCapturedBIR">
    <input name="Bsphandle"/>
    <input name="Purpose"/>
    <input name="Outputformatowner"/>
    <input name="Outputformattype"/>
    <output name="Form"/>
    <output name="Birhandle"/>

    <invoke activity="Capture" package="fb6ff5b0-7d5e-11de-8a39-0800200c9a66" break_on_break="true">
      <input name="Bsphandle" var="Bsphandle"/>
      <input name="Purpose" value="Purpose"/>
      <input name="Outputformatowner" var="Outputformatowner"/>
      <input name="Outputformattype" var="Outputformattype"/>
      <output name="Capturedbirhandle" setvar="Birhandle"/>
    </invoke>

    <set name="Form" value="2"/>
  </activity>

  <!-- ****************************** -->
  <!-- Activity to Capture and process -->
```

```
  <!-- ****************************** -->
  <activity name="PrepareProcessedBIR">
    <input name="Bsphandle"/>
    <input name="Purpose"/>
    <input name="Outputformatowner"/>
    <input name="Outputformattype"/>
    <output name="Form"/>
    <output name="Birhandle"/>

    <invoke activity="Capture" package="fb6ff5b0-7d5e-11de-8a39-0800200c9a66" break_on_break="true">
      <input name="Bsphandle" var="Bsphandle"/>
      <input name="Purpose" value="Purpose"/>
      <input name="Outputformatowner" var="Outputformatowner"/>
      <input name="Outputformattype" var="Outputformattype"/>
      <output name="Capturedbirhandle" setvar="capturedbir_birhandle"/>
    </invoke>
    <invoke activity="Process" package="fb6ff5b0-7d5e-11de-8a39-0800200c9a66" break_on_break="true">
      <input name="Bsphandle" var="Bsphandle"/>
      <input name="Capturedbir_form" value="2"/>
      <input name="Capturedbir_birhandle" var="capturedbir_birhandle"/>
      <input name="Outputformatowner" value="Outputformatowner"/>
      <input name="Outputformattype" value="Outputformattype"/>
      <output name="Processedbirhandle" setvar="Birhandle"/>
    </invoke>
    <set name="Form" value="2"/>
  </activity>
</package>
```

8.4 Assertion 1.1 – BioAPI_Init

Description: This assertion calls BioAPI_Init with the input parameters described in Default Input Table and Test Condition Table, and checks if the framework under test returns BioAPI_OK or an error value in accordance with the description in Expected Result Table. The assertion is according to the following statement in the subclauses from BioAPI 2.0 standard document (ISO/IEC19784-1:2006), which are described in References in this subclause in this part of ISO/IEC 24709.

Excerpts:

Subclause 8.1.1

BioAPI_Return BioAPI BioAPI_Init (BioAPI_Version Version);

This function initialises the BioAPI Framework and verifies that the version of the BioAPI Framework expected by the application is compatible with the version of the BioAPI on the system. This function should be called at least once by the application.

Any call to BioAPI_Init which occurs while there are previous calls to BioAPI_Init that have not been followed by a corresponding call to BioAPI_Terminate will be handled as follows: The BioAPI Framework will respond with either BioAPI_OK (if and only if the handling of the proposed version number by the framework is compatible with the handling of the version number that was proposed by previous BioAPI_Init call) or __BioAPIERR_INCOMPATIBLE_VERSION, but will not reinitialise. A count of the number of successful BioAPI_Init calls will be maintained by the Framework, which will not terminate until a corresponding number of BioAPI_Terminate calls have been issued.

This function is handled internally within the BioAPI Framework and is not passed through to a BSP.

A BioAPI_RETURN value indicates success or specifies a particular error condition. The value BioAPI_OK indicates success. All other values represent an error condition.

Subclause 11.2.3

#define __BioAPIERR_INCOMPATIBLE_Version (0x00010a)

References: 8.1.1, 11.2.3

Scenario:

The testing application does the followings:

a) Calls BioAPI_Init with the conditions given by the Default Input Table and by each row of the Test Condition Table.

b) Checks the return value of the BioAPI_Init and if it is not as described in the Expected Result Table, then issues a fail conformity response.

Parameters: The input parameters described below table are used for this assertion.

Table 13 — Default Input Table for BioAPI_Init

Number	Input parameter name	Input parameter value (underscore:invalid)
1	Version	32

Table 14 — Test Condition Table for BioAPI_Init

Test number	Input parameter name	Input parameter value (underscore:invalid)	Supported options in BSP Schema		Return value (from BioSPI)
			OPERATIONS_MASK	OPTIONS_MASK	
010101	Version	32	-	-	-
010102	Version	16	-	-	-

Table 15 — Expected Result Table for BioAPI_Init

Test number	BioSPI function (BioAPI/BioSPI parameter check)	BioAPI function			Other conditions
		Return value	Output parameter name	Output parameter value	
010101	-	*OK*	-	-	-
010102	-	*INCOMPATIBLE_VERSION*	-	-	-

Assertion language package

```
<?xml version="1.0" encoding="UTF-8"?>
<package name="4839c860-7929-11de-8a39-0800200c9a66">
  <author>ISO/IEC JTC1 SC37</author>
  <description>This package contains the assertion "BioAPI_Init".</description>

  <!-- ************** -->
  <!-- Test assertion -->
  <!-- ************** -->
  <assertion name="BioAPI_Init" model="frameworkTesting">
    <description>
      This assertion checks if BioAPI_Init is called with input parameters as described in Test Condition Table,
      the framework under test returns BioAPI_OK or error value in accordance with the description
      in Expected Result Table.
    </description>

    <!-- Parameter given by the CTS by referring to the Test Condition Table. -->
    <input name="_testnumber"/>
    <input name="_version"/>

    <!-- Parameter given by the CTS by referring to the Expected Result Table. -->
    <input name="_expected_return_value"/>

    <!-- Invocation of the primary activity of this assertion. -->
    <invoke activity="BioAPI_Init"/>
  </assertion>

  <!-- **************** -->
  <!-- Primary activity -->
  <!-- **************** -->
  <activity name="BioAPI_Init">

    <!-- Invoke the function BioAPI_Init. -->
    <invoke function="BioAPI_Init">
      <input name="Version" var="_version"/>
      <return setvar="return"/>
    </invoke>

    <!-- Assertion -->
    <assert_condition response_if_false="fail">
```

```
      <description>
        The function BioAPI_Init has returned the expected return value.
      </description>
      <or>
        <!-- If the "_expected_return_value" parameter is 0xFFFFFFFF,
             then check the "return" parameter is not __BioAPI_OK. (error check only) -->
        <and>
          <equal_to var1="_expected_return_value" var2="_indeterminate_error"/>
          <not_equal_to var1="return" var2="__BioAPI_OK"/>
        </and>
        <and>
          <not_equal_to var1="_expected_return_value" var2="_indeterminate_error"/>
          <equal_to var1="return" var2="_expected_return_value"/>
        </and>
      </or>
    </assert_condition>

    <!-- Invoke the functions Terminate -->
    <invoke activity="Termination" package="fb6ff5b0-7d5e-11de-8a39-0800200c9a66" break_on_break="true"/>
  </activity>
</package>
```

8.5 Assertion 1.2 – BioAPI_Terminate

Description: This assertion calls BioAPI_Terminate with the input parameters described in Default Input Table and Test Condition Table, and checks if the framework under test returns BioAPI_OK or an error value in accordance with the description in Expected Result Table. The assertion is according to the following statement in the subclauses from BioAPI 2.0 standard document (ISO/IEC19784-1:2006), which are described in References in this subclause in this part of ISO/IEC 24709.

Excerpts:

Subclause 8.1.2

BioAPI_RETURN BioAPI BioAPI_Terminate (void);

This function shall only be called if there is at least one successful call to BioAPI_Init for which a corresponding call to this function has not yet been made. BioAPI_Terminate shall not perform any actions if there are any outstanding calls to BioAPI_Init (i.e., calls to BioAPI_Init that have not been followed by corresponding calls to BioAPI_Terminate).

This function should not be called by the application if there is a call to BioAPI_BSPLoad for which a corresponding call to BioAPI_BSPUnload (for a given BSP UUID) has not yet been made. However, should this function be called while there are still BSPs loaded, then for each call to BioAPI_BSPLoad without a corresponding call to BioAPI_BSPUnload, the actions relative to the missing corresponding call to BioAPI_BSPUnload shall be implicitly performed by the Framework (as though the corresponding function had been called at that time), followed by the actions relative to BioAPI_Terminate (i.e, the Framework shall unload any BSPs prior to terminating).

This function is handled internally within the BioAPI Framework and is not passed through to a BSP, except where a BioAPI_BSPUnload is implied, as specified above.

A BioAPI_RETURN value indicates success or specifies a particular error condition. The value BioAPI_OK indicates success. All other values represent an error condition.

NOTE A BioAPI_Terminate operation is typically not expected to fail; however, should an anomaly condition occur, the application is not required to take any further action.

References: 8.1.2

Scenario:

The testing application does the followings:

a) Initialises the Framework under test.

b) If the Test Number is 010202, calls BioAPI_BSPLoad.

c) Calls BioAPI_Terminate with the conditions given by the Default Input Table and by each row of the Test Condition Table.

d) If the Test Number is 010202, check if the BioSPI_BSPUnload is called after the BioAPI_Terminate.

e) Checks the return value of the BioAPI_Terminate and if it is not as described in the Expected Result Table, then issues a fail conformity response.

Parameters: The input parameters described below table are used for this assertion.

Table 16 — Default Input Table for BioAPI_Terminate

Number	Input parameter name	Input parameter value (underscore:invalid)
1	-	-

Table 17 — Test Condition Table for BioAPI_Terminate

Test number	Input parameter name	Input parameter value (underscore:invalid)	Supported options in BSP Schema		Return value (from BioSPI)
			OPERATIONS_MASK	OPTIONS_MASK	
010201	-	-	-	-	-
010202	-	-	-	-	-

Table 18 — Expected Result Table for BioAPI_Terminate

Test number	BioSPI function (BioAPI/BioSPI parameter check)	BioAPI function			Other conditions
		Return value	Output parameter name	Output parameter value	
010201	-	*OK*	-	-	Check if the BioSPI_BSPUnload is called.
010202	-	*OK*	-	-	-

Assertion language package

```
<?xml version="1.0" encoding="UTF-8"?>
<package name="8782cd50-7929-11de-8a39-0800200c9a66">
  <author>ISO/IEC JTC1 SC37</author>
  <description>This package contains the assertion "BioAPI_Terminate".</description>

  <!-- ************** -->
  <!-- Test assertion -->
  <!-- ************** -->
  <assertion name="BioAPI_Terminate" model="frameworkTesting">
    <description>
      This assertion checks if BioAPI_Terminate is called with input parameters as described in Test Condition Table,
      the framework under test returns BioAPI_OK or error value in accordance with the description in Expected
      Result Table.
    </description>

    <!-- Parameter given by the CTS by referring to the Test Condition Table. -->
    <input name="_testnumber"/>

    <!-- Parameter given by the CTS by referring to the Expected Result Table. -->
    <input name="_expected_return_value"/>

    <!-- Parameter given by the CTS. -->
    <input name="_bspuuid"/>

    <!-- Invocation of the primary activity of this assertion. -->
    <invoke activity="BioAPI_Terminate"/>

    <!-- Bind activities to check the BioSPI function invoked by the framework under test -->
    <bind activity="SPI_BSPUnloadCheck" function="BioSPI_BSPUnload"/>
  </assertion>

  <!-- **************** -->
  <!-- Primary activity -->
  <!-- **************** -->
  <activity name="BioAPI_Terminate">

    <!-- Flag _bspunloadcheck that shows called BioSPI_BSPUnload. -->
    <set name="_bspunloadcheck" value="0"/>
    <!-- Invoke the activity Initialisation to initialise the framework under test -->
    <invoke activity="Initialisation" package="fb6ff5b0-7d5e-11de-8a39-0800200c9a66" break_on_break="true"/>

    <!-- If Test Number is 010202, Invoke the function BioAPI_BSPLoad. -->
    <set name="return" value="__BioAPI_OK"/>
    <invoke function="BioAPI_BSPLoad">
      <only_if>
        <not_equal_to var1="_testnumber" value2="010202"/>
      </only_if>
      <input name="BSPUuid" var="_bspuuid"/>
      <input name="AppNotifyCallback" value="*"/>
      <input name="AppNotifyCallbackCtx" value="*"/>
      <return setvar="return"/>
    </invoke>
    <assert_condition response_if_false="undecided" break_if_false="true">
      <description>
        The function BioAPI_BSPLoad has returned BioAPI_OK.
      </description>
      <equal_to var1="return" var2="__BioAPI_OK"/>
    </assert_condition>

    <!-- Invoke the function BioAPI_Terminate. -->
    <invoke function="BioAPI_Terminate">
      <return setvar="return"/>
    </invoke>

    <!-- Assertion -->
    <assert_condition response_if_false="fail">
      <description>
        The function BioAPI_Terminate has returned the expected return value.
      </description>
      <and>
        <or>
          <!-- If the "_expected_return_value" parameter is 0xFFFFFFFF,
               then check the "return" parameter is not __BioAPI_OK. (error check only) -->
          <and>
            <equal_to var1="_expected_return_value" var2="_indeterminate_error"/>
            <not_equal_to var1="return" var2="__BioAPI_OK"/>
          </and>
          <and>
            <not_equal_to var1="_expected_return_value" var2="_indeterminate_error"/>
            <equal_to var1="return" var2="_expected_return_value"/>
          </and>
        </or>
        <or>
          <not_equal_to var1="_testnumber" value2="010202"/>
          <and>
            <equal_to var1="_testnumber" value2="010202"/>
            <equal_to var1="_bspunloadcheck" value2="1"/>
          </and>
        </or>
      </and>
    </assert_condition>
  </activity>

  <!-- ******************************************* -->
  <!-- Activity bound to BioSPI_BSPUnload to check -->
  <!-- ******************************************* -->
  <activity name="SPI_BSPUnloadCheck">
    <input name="BSPUuid"/>
    <output name="return"/>

    <!-- Flag _bspunloadcheck that shows called BioSPI_BSPUnload. -->
    <assert_condition response_if_false="fail">
      <or>
        <not_equal_to var1="_testnumber" value2="010202"/>
        <and>
          <equal_to var1="_testnumber" value2="010202"/>
          <equal_to var1="_bspunloadcheck" value2="0"/>
        </and>
      </or>
    </assert_condition>
    <set name="_bspunloadcheck" value="1">
      <only_if>
        <equal_to var1="_testnumber" value2="010202"/>
      </only_if>
    </set>
  </activity>
</package>
```

8.6 Assertion 1.3 – BioAPI_GetFrameworkInfo

Description: This assertion calls BioAPI_GetFrameworkInfo with the input parameters described in Default Input Table and Test Condition Table, and checks if the framework under test returns BioAPI_OK or an error value in accordance with the description in Expected Result Table. The assertion is according to the following statement in the subclauses from BioAPI 2.0 standard document (ISO/IEC19784-1:2006), which are described in References in this subclause in this part of ISO/IEC 24709.

Excerpts:

Subclause 8.1.3

```
BioAPI_RETURN BioAPI BioAPI_GetFrameworkInfo
        (BioAPI_FRAMEWORK_SCHEMA      *FrameworkSchema);
```

This function returns information about the BioAPI Framework itself. Since multiple frameworks may exist on a computer, applications will need information about them in order to choose the one to use.
Return value

A BioAPI_RETURN value indicates success or specifies a particular error condition. The value BioAPI_OK indicates success. All other values represent an error condition.

References: 8.1.3

Scenario:
The testing application does the followings:

a) Initialises the Framework under test.

b) Calls BioAPI_GetFrameworkInfo with the conditions given by the Default Input Table and by each row of the Test Condition Table.

c) Checks the return value of the BioAPI_GetFrameworkInfo and if it is not as described in the Expected Result Table, then issues a fail conformity response.

d) Checks the output parameters and the contents of the pointer variables and if either of them is not as described in the column 'BioAPI function' in the Expected Result Table, then issues a fail conformity response.

e) Terminates the Framework under test.

Parameters: The input parameters described below table are used for this assertion.

Table 19 — Default Input Table for BioAPI_GetFrameworkInfo

Number	Input parameter name	Input parameter value (underscore:invalid)
1	no_FrameworkSchema	false

Table 20 — Test Condition Table for BioAPI_GetFrameworkInfo

Test number	Input parameter name	Input parameter value (underscore:invalid)	Supported options in BSP Schema		Return value (from BioSPI)
			OPERATIONS_MASK	OPTIONS_MASK	
010301	no_FrameworkSchema	false	-	-	-
010302	no_FrameworkSchema	true	-	-	-

Table 21 — Expected Result Table for BioAPI_GetFrameworkInfo

Test number	BioSPI function (BioAPI/BioSPI parameter check)	BioAPI function			Other conditions
		Return value	Output parameter name	Output parameter value	
010301	-	*OK*	Framework schema	Framework schema	-
010302	-	*indeterminate error*	-	-	-

NOTE1: For the detail of the cell of the test number row "010301" and the column "Output parameter names", see ISO/IEC 24709-1 9.2.9.
NOTE2: For the detail of the cell of the test number row "010301" and the column "Contents of pointer variables", see Table10.

Assertion language package

```
<?xml version="1.0" encoding="UTF-8"?>
<package name="b3a468d0-7929-11de-8a39-0800200c9a66">
  <author>ISO/IEC JTC1 SC37</author>
  <description>This package contains the assertion "BioAPI_GetFrameworkInfo".</description>

  <!-- ************** -->
  <!-- Test assertion -->
  <!-- ************** -->
  <assertion name="BioAPI_GetFrameworkInfo" model="frameworkTesting">
    <description>
      This assertion checks if BioAPI_GetFrameworkInfo is called with input parameters as described
      in Test Condition Table, the framework under test returns BioAPI_OK or error value in accordance
      with the description in Expected Result Table.
    </description>

    <!-- Parameter given by the CTS by referring to the Test Condition Table. -->
    <input name="_testnumber"/>
    <input name="_no_frameworkschema" />

    <!-- Parameter given by the CTS by referring to the Expected Result Table. -->
    <input name="_expected_return_value"/>
    <input name="_expected_Frameworkuuid"/>
    <input name="_expected_Description"/>
    <input name="_expected_Path"/>
    <input name="_expected_SpecVersion"/>
    <input name="_expected_ProductVersion"/>
    <input name="_expected_Vendor"/>
    <input name="_expected_FwPropertyId"/>
    <input name="_expected_FwProperty"/>

    <!-- Invocation of the primary activity of this assertion
         with input parameter values assigned from the assertion's parameters. -->
    <invoke activity="BioAPI_GetFrameworkInfo"/>

  </assertion>

  <!-- **************** -->
  <!-- Primary activity -->
  <!-- **************** -->
  <activity name="BioAPI_GetFrameworkInfo">

    <!-- Invoke the activity Initialisation to initialise the framework under test -->
    <invoke activity="Initialisation"
            package="fb6ff5b0-7d5e-11de-8a39-0800200c9a66" break_on_break="true"/>

    <!-- Invoke the function BioAPI_GetFrameworkInfo. -->
    <invoke function="BioAPI_GetFrameworkInfo">
      <input name="no_Frameworkschema" var="_no_frameworkschema"/>
      <output name="Frameworkuuid" setvar="frameworkuuid"/>
      <output name="Description" setvar="description"/>
      <output name="Path" setvar="Path"/>
      <output name="SpecVersion" setvar="specversion"/>
      <output name="ProductVersion" setvar="productversion"/>
      <output name="Vendor" setvar="Vendor"/>
      <output name="FwPropertyId" setvar="fwpropertyid"/>
      <output name="FwProperty" setvar="fwproperty"/>
      <return setvar="return"/>
    </invoke>

    <!-- Extract the error code from the error value. -->
    <invoke activity="ExtractErrorCode"
            package="fb6ff5b0-7d5e-11de-8a39-0800200c9a66" break_on_break="true">
      <output name="Returnvalue" setvar="return"/>
    </invoke>

    <!-- Assertion -->
    <assert_condition response_if_false="fail">
      <description>
        The function BioAPI_BSPUnload has returned the expected return value.
      </description>
      <and>
        <or>
```

```
          <!-- If the "_expected_return_value" parameter is 0xFFFFFFFF,
               then check the "return" parameter is not __BioAPI_OK. (error check only) -->
          <and>
            <equal_to var1="_expected_return_value" var2="_indeterminate_error"/>
            <not_equal_to var1="return" var2="__BioAPI_OK"/>
          </and>
          <and>
            <not_equal_to var1="_expected_return_value" var2="_indeterminate_error"/>
            <equal_to var1="return" var2="_expected_return_value"/>
          </and>
        </or>
        <equal_to var1="Frameworkuuid" var2="_expected_frameworkuuid"/>
        <equal_to var1="Description" var2="_expected_description"/>
        <equal_to var1="Path" var2="_expected_path"/>
        <equal_to var1="SpecVersion" var2="_expected_specversion"/>
        <equal_to var1="ProductVersion" var2="_expected_productversion"/>
        <equal_to var1="Vendor" var2="_expected_vendor"/>
        <equal_to var1="FwPropertyId" var2="_expected_fwpropertyid"/>
        <equal_to var1="FwProperty" var2="_expected_fwproperty"/>
      </and>
    </assert_condition>

    <!-- Invoke the activity Terminate to the framework under test -->
    <invoke activity="Termination"
            package="fb6ff5b0-7d5e-11de-8a39-0800200c9a66" break_on_break="true"/>

  </activity>
</package>
```

8.7 Assertion 1.4 – BioAPI_EnumBSPs

Description: This assertion calls BioAPI_EnumBSPs with the input parameters described in Default Input Table and Test Condition Table, and checks if the framework under test returns BioAPI_OK or an error value in accordance with the description in Expected Result Table. The assertion is according to the following statement in the subclauses from BioAPI 2.0 standard document (ISO/IEC19784-1:2006), which are described in References in this subclause in this part of ISO/IEC 24709.

Excerpts:

Subclause 8.1.4

```
BioAPI_RETURN BioAPI BioAPI_EnumBSPs
       (BioAPI_BSP_SCHEMA          **BSPSchemaArray,
       uint32_t                    *NumberOfElements);
```

This function provides information about all BSPs currently installed in the component registry. It performs the following actions (in order):

a) allocates a memory block large enough to contain an array of elements of type BioAPI_BSP_SCHEMA with as many elements as the number of installed BSPs;

b) fills the array with the BSP schemas of all installed BSPs; and

c) returns the address of the array in the BSPSchemaArray parameter and the number of elements of the array in the NumberOfElements parameter.

This function shall only be called if there is at least one call to BioAPI_Init for which a corresponding call to

BioAPI_Terminate has not yet been made. This function is handled internally within the BioAPI Framework and is not passed through to any BSP. The memory block containing the array shall be freed by the application via a call to BioAPI_Free (see clause 8.7.2) when it is no longer needed by the application. The memory block pointed to by the Path member within each element of the array shall also be freed by the application via a call to BioAPI_Free when it is no longer needed by the application.

A BioAPI_RETURN value indicates success or specifies a particular error condition. The value

BioAPI_OK indicates success. All other values represent an error condition.

References: 8.1.4

Scenario:

The testing application does the followings:

a) Initialises the Framework under test.

b) Calls BioAPI_EnumBSPs with the conditions given by the Default Input Table and by each row of the Test Condition Table.

c) Checks the return value of the BioAPI_EnumBSPs and if it is not as described in the Expected Result Table, then issues a fail conformity response.

d) Terminates the Framework under test.

Parameters: The input parameters described below table are used for this assertion.

Table 22 — Default Input Table for BioAPI_EnumBSPs

Number	Input parameter name	Input parameter value (underscore:invalid)
1	no_BSPSchemaArray	false
2	no_NumberOfElements	false

Table 23 — Test Condition Table for BioAPI_EnumBSPs

Test number	Input parameter name	Input parameter value (underscore:invalid)	Supported options in BSP Schema		Return value (from BioSPI)
			OPERATIONS_MASK	OPTIONS_MASK	
010401	no_BSPSchemaArray	false	-	-	-
010402	no_BSPSchemaArray	true	-	-	-
010403	no_NumberOfElements	true	-	-	-

Table 24 — Expected Result Table for BioAPI_EnumBSPs

Test number	BioSPI function (BioAPI/BioSPI parameter check)	BioAPI function			Other conditions
		Return value	Output parameter name	Output parameter value	
010401	-	*OK*	BSP schema	BSP schema	-
010402	-	*indeterminate error*	-	-	-
010403	-	*indeterminate error*	-	-	-

NOTE1: For the detail of the cell of the test number row "010401" and the column "Output parameter names", see ISO/IEC 24709-1 9.2.10.

NOTE2: For the detail of the cell of the test number row "010401" and the column "Contents of pointer variables", see Table1.

Assertion language package

```
<?xml version="1.0" encoding="UTF-8"?>
<package name="ce45e240-7929-11de-8a39-0800200c9a66">
  <author>ISO/IEC JTC1 SC37</author>
  <description>This package contains the assertion "BioAPI_EnumBSPs".</description>

  <!-- ************** -->
  <!-- Test assertion -->
  <!-- ************** -->
  <assertion name="BioAPI_EnumBSPs" model="frameworkTesting">
    <description>
      This assertion checks if BioAPI_EnumBSPs is called with input parameters as described in Test Condition Table,
      the framework under test returns BioAPI_OK or error value in accordance with the description in
      Expected Result Table.
    </description>

    <!-- Parameter given by the CTS by referring to the Test Condition Table. -->
    <input name="_testnumber"/>
    <input name="_no_bspschemaarray"/>
    <input name="_no_numberofelements"/>

    <!-- Parameter given by the CTS by referring to the Expected Result Table. -->
    <input name="_expected_return_value"/>
    <input name="_expected_bspuuid"/>
    <input name="_expected_description"/>
    <input name="_expected_path"/>
    <input name="_expected_specversion"/>
    <input name="_expected_productversion"/>
    <input name="_expected_vendor"/>
    <input name="_expected_format_1_formatowner"/>
    <input name="_expected_format_1_formattype"/>
    <input name="_expected_format_2_formatowner"/>
    <input name="_expected_format_2_formattype"/>
    <input name="_expected_format_3_formatowner"/>
    <input name="_expected_format_3_formattype"/>
    <input name="_expected_format_4_formatowner"/>
    <input name="_expected_format_4_formattype"/>
    <input name="_expected_numsupportedformats"/>
    <input name="_expected_typemultiple"/>
    <input name="_expected_typefacialfeatures"/>
    <input name="_expected_typevoice"/>
    <input name="_expected_typefingerprint"/>
    <input name="_expected_typeiris"/>
    <input name="_expected_typeretina"/>
    <input name="_expected_typehandgeometry"/>
    <input name="_expected_typesignaturedynamics"/>
    <input name="_expected_typekeystrokedynamics"/>
    <input name="_expected_typelipmovement"/>
    <input name="_expected_typethermalfaceimage"/>
    <input name="_expected_typethermalhandimage"/>
    <input name="_expected_typegait"/>
    <input name="_expected_typeother"/>
    <input name="_expected_typepassword"/>
    <input name="_expected_operationenableevents"/>
    <input name="_expected_operationsetguicallbacks"/>
    <input name="_expected_operationcapture"/>
    <input name="_expected_operationcreatetemplate"/>
    <input name="_expected_operationprocess"/>
    <input name="_expected_operationprocesswithauxbir"/>
    <input name="_expected_operationverifymatch"/>
    <input name="_expected_operationidentifymatch"/>
    <input name="_expected_operationenroll"/>
    <input name="_expected_operationverify"/>
    <input name="_expected_operationidentify"/>
    <input name="_expected_operationimport"/>
    <input name="_expected_operationpresetidentifypopulation"/>
    <input name="_expected_operationdatabaseoperations"/>
    <input name="_expected_operationsetpowermode"/>
    <input name="_expected_operationsetindicatorstatus"/>
    <input name="_expected_operationgetindicatorstatus"/>
    <input name="_expected_operationcalibratesensor"/>
    <input name="_expected_operationutilities"/>
    <input name="_expected_operationqueryunits"/>
    <input name="_expected_operationquerybfps"/>
    <input name="_expected_operationcontrolunit"/>
    <input name="_expected_optionraw"/>
    <input name="_expected_optionqualityraw"/>
    <input name="_expected_optionqualityintermediate"/>
    <input name="_expected_optionqualityprocessed"/>
    <input name="_expected_optionappgui"/>
    <input name="_expected_optionsourcepresent"/>
    <input name="_expected_optionpayload"/>
    <input name="_expected_optionbirsign"/>
    <input name="_expected_optionbirencrypt"/>
    <input name="_expected_optiontemplateupdate"/>
    <input name="_expected_optionadaptation"/>
    <input name="_expected_optionbinning"/>
    <input name="_expected_optionselfcontaineddevice"/>
    <input name="_expected_optionmoc"/>
    <input name="_expected_optionsubtypetocapture"/>
    <input name="_expected_optionsensorbfp"/>
    <input name="_expected_optionarchivebfp"/>
    <input name="_expected_optionmatchingbfp"/>
    <input name="_expected_optionprocessingbfp"/>
    <input name="_expected_optioncoarsescores"/>
    <input name="_expected_payloadpolicy"/>
    <input name="_expected_maxpayloadsize"/>
    <input name="_expected_defaultverifytimeout"/>
    <input name="_expected_defaultidentifytimeout"/>
    <input name="_expected_defaultcapturetimeout"/>
    <input name="_expected_defaultenrolltimeout"/>
    <input name="_expected_defaultcalibratetimeout"/>
    <input name="_expected_maxbspdbsize"/>
    <input name="_expected_maxidentify"/>

    <!-- Invocation of the primary activity of this assertion. -->
    <invoke activity="BioAPI_EnumBSPs"/>

  </assertion>

  <!-- **************** -->
  <!-- Primary activity -->
  <!-- **************** -->
  <activity name="BioAPI_EnumBSPs">

    <!-- Invoke the activity Initialisation to initialise the framework under test -->
    <invoke activity="Initialisation" package="fb6ff5b0-7d5e-11de-8a39-0800200c9a66" break_on_break="true"/>
```

```
<!-- Invoke the function BioAPI_EnumBSPs. -->
<invoke function="BioAPI_EnumBSPs">
  <input name="no_BSPSchemaArray" var="_no_bspschemaarray"/>
  <input name="no_NumberOfelements" var="_no_numberofelements"/>
  <output name="NumberOfelements" setvar="numberofelements"/>
  <output name="BSPSchema_1_BSPUuid" setvar="sa1_bspuuid"/>
  <output name="BSPSchema_1_Description" setvar="sa1_description"/>
  <output name="BSPSchema_1_Path" setvar="sa1_path"/>
  <output name="BSPSchema_1_SpecVersion" setvar="sa1_specversion"/>
  <output name="BSPSchema_1_ProductVersion" setvar="sa1_productversion"/>
  <output name="BSPSchema_1_Vendor" setvar="sa1_vendor"/>
  <output name="BSPSchema_1_Format_1_FormatOwner" setvar="sa1_format_1_formatowner"/>
  <output name="BSPSchema_1_Format_1_FormatType" setvar="sa1_format_1_formattype"/>
  <output name="BSPSchema_1_Format_2_FormatOwner" setvar="sa1_format_2_formatowner"/>
  <output name="BSPSchema_1_Format_2_FormatType" setvar="sa1_format_2_formattype"/>
  <output name="BSPSchema_1_Format_3_FormatOwner" setvar="sa1_format_3_formatowner"/>
  <output name="BSPSchema_1_Format_3_FormatType" setvar="sa1_format_3_formattype"/>
  <output name="BSPSchema_1_Format_4_FormatOwner" setvar="sa1_format_4_formatowner"/>
  <output name="BSPSchema_1_Format_4_FormatType" setvar="sa1_format_4_formattype"/>
  <output name="BSPSchema_1_NumSupportedFormats" setvar="sa1_numsupportedformats"/>
  <output name="BSPSchema_1_TypeMultiple" setvar="sa1_typemultiple"/>
  <output name="BSPSchema_1_TypeFacialFeatures" setvar="sa1_typefacialfeatures"/>
  <output name="BSPSchema_1_TypeVoice" setvar="sa1_typevoice"/>
  <output name="BSPSchema_1_TypeFingerprint" setvar="sa1_typefingerprint"/>
  <output name="BSPSchema_1_TypeIris" setvar="sa1_typeiris"/>
  <output name="BSPSchema_1_TypeRetina" setvar="sa1_typeretina"/>
  <output name="BSPSchema_1_TypeHandGeometry" setvar="sa1_typehandgeometry"/>
  <output name="BSPSchema_1_TypeSignatureDynamics" setvar="sa1_typesignaturedynamics"/>
  <output name="BSPSchema_1_TypeKeystrokeDynamics" setvar="sa1_typekeystrokedynamics"/>
  <output name="BSPSchema_1_TypeLipMovement" setvar="sa1_typelipmovement"/>
  <output name="BSPSchema_1_TypeThermalFaceImage" setvar="sa1_typethermalfaceimage"/>
  <output name="BSPSchema_1_TypeThermalHandImage" setvar="sa1_typethermalhandimage"/>
  <output name="BSPSchema_1_TypeGait" setvar="sa1_typegait"/>
  <output name="BSPSchema_1_TypeOther" setvar="sa1_typeother"/>
  <output name="BSPSchema_1_TypePassword" setvar="sa1_typepassword"/>
  <output name="BSPSchema_1_OperationEnableEvents" setvar="sa1_operationenableevents"/>
  <output name="BSPSchema_1_OperationSetGUICallbacks" setvar="sa1_operationsetguicallbacks"/>
  <output name="BSPSchema_1_OperationCapture" setvar="sa1_operationcapture"/>
  <output name="BSPSchema_1_OperationCreateTemplate" setvar="sa1_operationcreatetemplate"/>
  <output name="BSPSchema_1_OperationProcess" setvar="sa1_operationprocess"/>
  <output name="BSPSchema_1_OperationProcessWithAuxBIR" setvar="sa1_operationprocesswithauxbir"/>
  <output name="BSPSchema_1_OperationVerifyMatch" setvar="sa1_operationverifymatch"/>
  <output name="BSPSchema_1_OperationIdentifyMatch" setvar="sa1_operationidentifymatch"/>
  <output name="BSPSchema_1_OperationEnroll" setvar="sa1_operationenroll"/>
  <output name="BSPSchema_1_OperationVerify" setvar="sa1_operationverify"/>
  <output name="BSPSchema_1_OperationIdentify" setvar="sa1_operationidentify"/>
  <output name="BSPSchema_1_OperationImport" setvar="sa1_operationimport"/>
  <output name="BSPSchema_1_OperationPresetIdentifyPopulation" setvar="sa1_operationpresetidentifypopulation"/>
  <output name="BSPSchema_1_OperationDatabaseOperations" setvar="sa1_operationdatabaseoperations"/>
  <output name="BSPSchema_1_OperationSetPowerMode" setvar="sa1_operationsetpowermode"/>
  <output name="BSPSchema_1_OperationSetIndicatorStatus" setvar="sa1_operationsetindicatorstatus"/>
  <output name="BSPSchema_1_OperationGetIndicatorStatus" setvar="sa1_operationgetindicatorstatus"/>
  <output name="BSPSchema_1_OperationCalibrateSensor" setvar="sa1_operationcalibratesensor"/>
  <output name="BSPSchema_1_OperationUtilities" setvar="sa1_operationutilities"/>
  <output name="BSPSchema_1_OperationQueryUnits" setvar="sa1_operationqueryunits"/>
  <output name="BSPSchema_1_OperationQueryBFPs" setvar="sa1_operationquerybfps"/>
  <output name="BSPSchema_1_OperationControlUnit" setvar="sa1_operationcontrolunit"/>
  <output name="BSPSchema_1_OptionRaw" setvar="sa1_optionraw"/>
  <output name="BSPSchema_1_OptionQualityRaw" setvar="sa1_optionqualityraw"/>
  <output name="BSPSchema_1_OptionQualityIntermediate" setvar="sa1_optionqualityintermediate"/>
  <output name="BSPSchema_1_OptionQualityProcessed" setvar="sa1_optionqualityprocessed"/>
  <output name="BSPSchema_1_OptionAppGUI" setvar="sa1_optionappgui"/>
  <output name="BSPSchema_1_OptionSourcePresent" setvar="sa1_optionsourcepresent"/>
  <output name="BSPSchema_1_OptionPayload" setvar="sa1_optionpayload"/>
  <output name="BSPSchema_1_OptionBIRSign" setvar="sa1_optionbirsign"/>
  <output name="BSPSchema_1_OptionBIREncrypt" setvar="sa1_optionbirencrypt"/>
  <output name="BSPSchema_1_OptionTemplateUpdate" setvar="sa1_optiontemplateupdate"/>
  <output name="BSPSchema_1_OptionAdaptation" setvar="sa1_optionadaptation"/>
  <output name="BSPSchema_1_OptionBinning" setvar="sa1_optionbinning"/>
  <output name="BSPSchema_1_OptionSelfContainedDevice" setvar="sa1_optionselfcontaineddevice"/>
  <output name="BSPSchema_1_OptionMOC" setvar="sa1_optionmoc"/>
  <output name="BSPSchema_1_OptionSubtypeToCapture" setvar="sa1_optionsubtypetocapture"/>
  <output name="BSPSchema_1_OptionSensorBFP" setvar="sa1_optionsensorbfp"/>
  <output name="BSPSchema_1_OptionArchiveBFP" setvar="sa1_optionarchivebfp"/>
  <output name="BSPSchema_1_OptionMatchingBFP" setvar="sa1_optionmatchingbfp"/>
  <output name="BSPSchema_1_OptionProcessingBFP" setvar="sa1_optionprocessingbfp"/>
  <output name="BSPSchema_1_OptionCoarseScores" setvar="sa1_optioncoarsescores"/>
  <output name="BSPSchema_1_PayloadPolicy" setvar="sa1_payloadpolicy"/>
  <output name="BSPSchema_1_MaxPayloadSize" setvar="sa1_maxpayloadsize"/>
  <output name="BSPSchema_1_DefaultVerifyTimeout" setvar="sa1_defaultverifytimeout"/>
  <output name="BSPSchema_1_DefaultIdentifyTimeout" setvar="sa1_defaultidentifytimeout"/>
  <output name="BSPSchema_1_DefaultCaptureTimeout" setvar="sa1_defaultcapturetimeout"/>
  <output name="BSPSchema_1_DefaultEnrollTimeout" setvar="sa1_defaultenrolltimeout"/>
  <output name="BSPSchema_1_DefaultCalibrateTimeout" setvar="sa1_defaultcalibratetimeout"/>
  <output name="BSPSchema_1_MaxBSPDbSize" setvar="sa1_maxbspdbsize"/>
  <output name="BSPSchema_1_MaxIdentify" setvar="sa1_maxidentify"/>
  <return setvar="return"/>
</invoke>

<!-- Extract the error code from the error value. -->
<invoke activity="ExtractErrorCode" package="fb6ff5b0-7d5e-11de-8a39-0800200c9a66" break_on_break="true">
  <output name="Returnvalue" setvar="return"/>
</invoke>

<!-- Assertion -->
<assert_condition response_if_false="fail">
  <description>
    The function BioAPI_EnumBSPs has returned the expected return value.
  </description>
  <and>
    <or>
      <!-- If the "_expected_return_value" parameter is 0xFFFFFFFF,
           then check the "return" parameter is not __BioAPI_OK. (error check only) -->
      <and>
        <equal_to var1="_expected_return_value" var2="_indeterminate_error"/>
        <not_equal_to var1="return" var2="__BioAPI_OK"/>
      </and>
      <and>
        <not_equal_to var1="_expected_return_value" var2="_indeterminate_error"/>
        <equal_to var1="return" var2="_expected_return_value"/>
      </and>
    </or>
    <equal_to var1="sa1_bspuuid" var2="_expected_bspuuid"/>
    <equal_to var1="sa1_description" var2="_expected_description"/>
    <equal_to var1="sa1_path" var2="_expected_path"/>
    <equal_to var1="sa1_specversion" var2="_expected_specversion"/>
    <equal_to var1="sa1_productversion" var2="_expected_productversion"/>
    <equal_to var1="sa1_vendor" var2="_expected_vendor"/>
    <equal_to var1="sa1_format_1_formatowner" var2="_expected_format_1_formatowner"/>
    <equal_to var1="sa1_format_1_formattype" var2="_expected_format_1_formattype"/>
    <equal_to var1="sa1_format_2_formatowner" var2="_expected_format_2_formatowner"/>
    <equal_to var1="sa1_format_2_formattype" var2="_expected_format_2_formattype"/>
    <equal_to var1="sa1_format_3_formatowner" var2="_expected_format_3_formatowner"/>
    <equal_to var1="sa1_format_3_formattype" var2="_expected_format_3_formattype"/>
    <equal_to var1="sa1_format_4_formatowner" var2="_expected_format_4_formatowner"/>
    <equal_to var1="sa1_format_4_formattype" var2="_expected_format_4_formattype"/>
    <equal_to var1="sa1_numsupportedformats" var2="_expected_numsupportedformats"/>
    <equal_to var1="sa1_typemultiple" var2="_expected_typemultiple"/>
    <equal_to var1="sa1_typefacialfeatures" var2="_expected_typefacialfeatures"/>
    <equal_to var1="sa1_typevoice" var2="_expected_typevoice"/>
    <equal_to var1="sa1_typefingerprint" var2="_expected_typefingerprint"/>
    <equal_to var1="sa1_typeiris" var2="_expected_typeiris"/>
    <equal_to var1="sa1_typeretina" var2="_expected_typeretina"/>
    <equal_to var1="sa1_typehandgeometry" var2="_expected_typehandgeometry"/>
    <equal_to var1="sa1_typesignaturedynamics" var2="_expected_typesignaturedynamics"/>
    <equal_to var1="sa1_typekeystrokedynamics" var2="_expected_typekeystrokedynamics"/>
    <equal_to var1="sa1_typelipmovement" var2="_expected_typelipmovement"/>
    <equal_to var1="sa1_typethermalfaceimage" var2="_expected_typethermalfaceimage"/>
    <equal_to var1="sa1_typethermalhandimage" var2="_expected_typethermalhandimage"/>
    <equal_to var1="sa1_typegait" var2="_expected_typegait"/>
    <equal_to var1="sa1_typeother" var2="_expected_typeother"/>
    <equal_to var1="sa1_typepassword" var2="_expected_typepassword"/>
    <equal_to var1="sa1_operationenableevents" var2="_expected_operationenableevents"/>
    <equal_to var1="sa1_operationsetguicallbacks" var2="_expected_operationsetguicallbacks"/>
    <equal_to var1="sa1_operationcapture" var2="_expected_operationcapture"/>
    <equal_to var1="sa1_operationcreatetemplate" var2="_expected_operationcreatetemplate"/>
    <equal_to var1="sa1_operationprocess" var2="_expected_operationprocess"/>
    <equal_to var1="sa1_operationprocesswithauxbir" var2="_expected_operationprocesswithauxbir"/>
    <equal_to var1="sa1_operationverifymatch" var2="_expected_operationverifymatch"/>
    <equal_to var1="sa1_operationidentifymatch" var2="_expected_operationidentifymatch"/>
    <equal_to var1="sa1_operationenroll" var2="_expected_operationenroll"/>
    <equal_to var1="sa1_operationverify" var2="_expected_operationverify"/>
    <equal_to var1="sa1_operationidentify" var2="_expected_operationidentify"/>
    <equal_to var1="sa1_operationimport" var2="_expected_operationimport"/>
    <equal_to var1="sa1_operationpresetidentifypopulation" var2="_expected_operationpresetidentifypopulation"/>
    <equal_to var1="sa1_operationdatabaseoperations" var2="_expected_operationdatabaseoperations"/>
    <equal_to var1="sa1_operationsetpowermode" var2="_expected_operationsetpowermode"/>
    <equal_to var1="sa1_operationsetindicatorstatus" var2="_expected_operationsetindicatorstatus"/>
    <equal_to var1="sa1_operationgetindicatorstatus" var2="_expected_operationgetindicatorstatus"/>
    <equal_to var1="sa1_operationcalibratesensor" var2="_expected_operationcalibratesensor"/>
    <equal_to var1="sa1_operationutilities" var2="_expected_operationutilities"/>
    <equal_to var1="sa1_operationqueryunits" var2="_expected_operationqueryunits"/>
    <equal_to var1="sa1_operationquerybfps" var2="_expected_operationquerybfps"/>
    <equal_to var1="sa1_operationcontrolunit" var2="_expected_operationcontrolunit"/>
    <equal_to var1="sa1_optionraw" var2="_expected_optionraw"/>
    <equal_to var1="sa1_optionqualityraw" var2="_expected_optionqualityraw"/>
    <equal_to var1="sa1_optionqualityintermediate" var2="_expected_optionqualityintermediate"/>
    <equal_to var1="sa1_optionqualityprocessed" var2="_expected_optionqualityprocessed"/>
    <equal_to var1="sa1_optionappgui" var2="_expected_optionappgui"/>
    <equal_to var1="sa1_optionsourcepresent" var2="_expected_optionsourcepresent"/>
    <equal_to var1="sa1_optionpayload" var2="_expected_optionpayload"/>
    <equal_to var1="sa1_optionbirsign" var2="_expected_optionbirsign"/>
    <equal_to var1="sa1_optionbirencrypt" var2="_expected_optionbirencrypt"/>
```

```
        <equal_to var1="sa1_optiontemplateupdate" var2="_expected_optiontemplateupdate"/>
        <equal_to var1="sa1_optionadaptation" var2="_expected_optionadaptation"/>
        <equal_to var1="sa1_optionbinning" var2="_expected_optionbinning"/>
        <equal_to var1="sa1_optionselfcontaineddevice" var2="_expected_optionselfcontaineddevice"/>
        <equal_to var1="sa1_optionmoc" var2="_expected_optionmoc"/>
        <equal_to var1="sa1_optionsubtypetocapture" var2="_expected_optionsubtypetocapture"/>
        <equal_to var1="sa1_optionsensorbfp" var2="_expected_optionsensorbfp"/>
        <equal_to var1="sa1_optionarchivebfp" var2="_expected_optionarchivebfp"/>
        <equal_to var1="sa1_optionmatchingbfp" var2="_expected_optionmatchingbfp"/>
        <equal_to var1="sa1_optionprocessingbfp" var2="_expected_optionprocessingbfp"/>
        <equal_to var1="sa1_optioncoarsescores" var2="_expected_optioncoarsescores"/>
        <equal_to var1="sa1_payloadpolicy" var2="_expected_payloadpolicy"/>
        <equal_to var1="sa1_maxpayloadsize" var2="_expected_maxpayloadsize"/>
        <equal_to var1="sa1_defaultverifytimeout" var2="_expected_defaultverifytimeout"/>
        <equal_to var1="sa1_defaultidentifytimeout" var2="_expected_defaultidentifytimeout"/>
        <equal_to var1="sa1_defaultcapturetimeout" var2="_expected_defaultcapturetimeout"/>
        <equal_to var1="sa1_defaultenrolltimeout" var2="_expected_defaultenrolltimeout"/>
        <equal_to var1="sa1_defaultcalibratetimeout" var2="_expected_defaultcalibratetimeout"/>
        <equal_to var1="sa1_maxbspdbsize" var2="_expected_maxbspdbsize"/>
        <equal_to var1="sa1_maxidentify" var2="_expected_maxidentify"/>
      </and>
    </assert_condition>

    <!-- Invoke the activity to terminate the framework under test -->
    <invoke activity="Termination" package="fb6ff5b0-7d5e-11de-8a39-0800200c9a66" break_on_break="true"/>
  </activity>
</package>
```

8.8 Assertion 1.5 – BioAPI_BSPLoad_And_BioSPI_BSPLoad

Description: This assertion calls BioAPI_BSPLoad with the input parameters described in Default Input Table and Test Condition Table, and checks if the framework under test returns BioAPI_OK or an error value in accordance with the description in Expected Result Table. The assertion is according to the following statement in the subclauses from BioAPI 2.0 standard document (ISO/IEC19784-1:2006), which are described in References in this subclause in this part of ISO/IEC 24709.

Excerpts:

Subclause 8.1.5

```
BioAPI_RETURN BioAPI BioAPI_Load
      (const  BioAPI_UUID           *BSPUuid,
       BioAPI_EventHandler          AppNotifyCallback,
       Void                         *AppNotifyCallbackCtx);
```

This function initialises a BSP using the BioSPI_BSPLoad (see clause 9.3.1.1 in ISO/IEC 19784-1:2006). Initialisation includes registering the application's event handler for the specified BSP and enabling all events. The application can choose to provide an event handler function to receive notification of events. Many applications can independently and concurrently load the same BSP, and each application can establish its own event handler. They will all receive notification of an event. The same or different event handlers can be used if an application loads multiple BSPs.

An application may establish as many event handlers as it wishes, for a given BSP, by calling BioAPI_BSPLoad one or more times for that BSP. An event handler is identified by a combination of address and context.

When an event occurs in a BSP, the BSP may send an event notification to the Framework by calling the

Framework's event handler.

When the Framework receives an event notification from a BSP, it shall send one notification to each event handler established by each application for which that event notification is enabled for that BSP. Therefore, a single event notification callback made from a BSP to the Framework may result in zero or more callbacks made by the Framework to zero or more applications.

When the framework receives an event notification from a BSP, it shall call all the event handlers established by each application for that BSP. If an application has set up multiple event handlers, they shall be called one at a time (in any order chosen by the Framework) rather than concurrently.

Event notification may occur at any time, either during a BioAPI call (related or unrelated to the event) or while there is no BioAPI call in execution. Application writers should ensure that all callbacks are properly and safely handled by the application, no matter when the application receives them.

NOTE This usually requires the use of thread synchronization techniques and discipline in the actions performed by the application code placed in event handlers.

There is a "use count" on the establishment of event handlers; they have to be de-established (by BioAPI_BSPUnload) as many times as they were established. When a BSP is loaded (BioAPI_BSPLoad), it shall raise an "insert" event immediately for each present BioAPI Unit. This will indicate to the biometric application that it can go ahead and do a BioAPI_BSPAttach. If the hardware component for a specific functionality is not present, the "insert" event cannot be raised until the hardware component has been plugged in.

This function shall only be called if there is at least one call to BioAPI_Init for which a corresponding call to

BioAPI_Terminate has not yet been made. The function BioAPI_BSPAttach can be invoked multiple times

for each call to BioAPI_BSPLoad.

The BioAPI_BSPLoad function is not to be called unless the BSP has been installed using BioAPI_Util_InstallBSP. A determination of installed BSPs can be made through a call to BioAPI_EnumBSPs.

A BioAPI_RETURN value indicates success or specifies a particular error condition. The value BioAPI_OK indicates success. All other values represent an error condition.

NOTE A BioAPI_Terminate operation is typically not expected to fail; however, should an anomaly condition occur, the application is not required to take any further action.

Subclause 9.3.1

```
BioSPI_RETURN BioAPI BioSPI_Load
        (const  BioAPI_UUID                  *BSPUuid,
        BioSPI_EventHandler                  BioAPINotifyCallback,
        BioSPI_BFP_ENUMERATION_HANDLER       BFPEnumerationHandler,
        BioSPI_MEMORY_FREE_HANDLER           *MemoryFreeHandler);
```

This function completes the component initialisation process between BioAPI and the biometric service provider. The function BioSPI_BSPLoad shall not be called more than once without a corresponding call to BioSPI_BSPUnload.

The BSPUuid identifies the invoked BSP.

The BioAPINotifyCallback defines a callback used to notify the BioAPI Framework of events of type BioAPI_EVENT in any ongoing, attached sessions. The BSP shall retain this information for later use.

The BFPEnumerationHandler is the address of the BFP enumeration handler callback provided by the Framework to the BSP. The BSP shall retain this information for later use. The BSP can use the callback whenever it needs to obtain information about the BFPs installed in the Framework.

The MemoryFreeHandler is the address of the memory deallocation handler callback provided by the Framework to the BSP. The BSP shall retain this information for later use. The BSP shall use the callback whenever it needs to deallocate a memory block that was allocated by the Framework during a prior callback to the BFP enumeration handler.

NOTE This is a sister function to BioAPI_BSPLoad (see clause 8.1.5).

References: 8.1.5, 9.3.1

Scenario:

a) The testing application does the followings:

1) Initialises the Framework under test, then installs.
2) Calls BioAPI_BSPLoad with the conditions given by the Default Input Table and by each row of the Test Condition Table.

b) The testing BSP does the followings:

1) Checks the parameters and the contents of the pointer variables as described in the column 'BioSPI function' in the Expected Result Table.
2) Sets the return value of the BioSPI_BSPLoad given by each row of the Test Condition Table, then returns to the Framework.

c) The testing application does the followings:

1) Checks the return value of the BioAPI_BSPLoad and if it is not as described in the Expected Result Table, then issues a fail conformity response.
2) Unloads and uninstalls the testing BSP, then terminates the Framework under test.

Parameters: This assertion uses input parameters described as follows.

Table 25 — Default Input Table for BioAPI_BSPLoad_And_BioSPI_BSPLoad

Number	Input parameter name	Input parameter value (underscore:invalid)
1	BSPUuid	*Valid Uuid*
2	no_BSPUuid	false
3	AppNotifyCallback	*
4	AppNotifyCallbackCtx	*

NOTE: The parameter name "no_BSPUuid" is not defined in ISO/IEC 24709-1 so the implementation of this parameter depends on each CTS.

Table 26 — Test Condition Table for BioAPI_BSPLoad_And BioSPI_BSPLoad

Test number	Input parameter name	Input parameter value (underscore:invalid)	Supported options in BSP Schema		Return value (from BioSPI)
			OPERATIONS_MASK	OPTIONS_MASK	
010501	BSPUuid	*Valid Uuid*	-	-	*OK*
010502	BSPUuid	<u>Invalid Uuid</u>	-	-	-
010503	no_BSPUuid	<u>true</u>	-	-	-
010504	AppNotifyCallback	0	-	-	*OK*
010505	AppNotifyCallbackCtx	0	-	-	*OK*

NOTE: The parameter name "no_BSPUuid" is not defined in ISO/IEC 24709-1 so the implementation of this parameter depends on each CTS.

Table 27 — Expected Result Table for BioAPI_BSPLoad_And_BioSPI_BSPLoad

Test number	BioSPI function (BioAPI/BioSPI parameter check)	BioAPI function			Other conditions
		Return value	Output parameter name	Output parameter value	
010501	X	*OK*	-	-	-
010502	-	*indeterminate error*	-	-	-
010503	-	*indeterminate error*	-	-	-
010504	X	*OK*	-	-	-
010505	X	*OK*	-	-	-

Assertion language package

```
<?xml version="1.0" encoding="UTF-8"?>
<package name="f481f070-7929-11de-8a39-0800200c9a66">
  <author>ISO/IEC JTC1 SC37</author>
  <description>This package contains the assertion "BioAPI_BSPLoad_And_BioSPI_BSPLoad".</description>

  <!-- ************** -->
  <!-- Test assertion -->
  <!-- ************** -->
  <assertion name="BioAPI_BSPLoad_And_BioSPI_BSPLoad" model="frameworkTesting">
    <description>
```

```
      This assertion checks if BioAPI_BSPLoad is called with input parameters as described in Test Condition Table,
      the framework under test returns BioAPI_OK or error value in accordance with the description in
      Expected Result Table.
    </description>

    <!-- Parameter given by the CTS by referring to the Test Condition Table. -->
    <input name="_testnumber"/>
    <input name="_bspuuid"/>
    <input name="_appnotifycallback"/>
    <input name="_appnotifycallbackctx"/>
    <input name="_no_bspuuid"/>

    <!-- Parameter given by the CTS by referring to the Expected Result Table. -->
    <input name="_expected_return_value"/>

    <!-- Invocation of the primary activity of this assertion. -->
    <invoke activity="BioAPI_BSPLoad_And_BioSPI_BSPLoad"/>

    <!-- Bind activities to check the BioSPI function invoked by the framework under test -->
    <bind activity="SPI_BSPLoad" function="BioSPI_BSPLoad"/>
  </assertion>

  <!-- **************** -->
  <!-- Primary activity -->
  <!-- **************** -->
  <activity name="BioAPI_BSPLoad_And_BioSPI_BSPLoad">

    <!-- Invoke the activity Initialisation to initialise the framework under test -->
    <invoke activity="Initialisation" package="fb6ff5b0-7d5e-11de-8a39-0800200c9a66" break_on_break="true"/>

    <!-- Invoke the function BioAPI_BspLoad. -->
    <invoke function="BioAPI_BSPLoad">
      <input name="BSPUuid" var="_bspuuid"/>
      <input name="APINotifyCallback" var="_apinotifycallback"/>
      <input name="APINotifyCallbackCtx" var="_apinotifycallbackctx"/>
      <input name="no_BSPUuid" var="_no_bspuuid"/>
      <return setvar="return"/>
    </invoke>

    <!-- Extract the error code from the error value. -->
    <invoke activity="ExtractErrorCode" package="fb6ff5b0-7d5e-11de-8a39-0800200c9a66" break_on_break="true">
      <output name="Returnvalue" setvar="return"/>
    </invoke>

    <!-- Assertion -->
    <assert_condition response_if_false="fail">
      <description>
        The function BioAPI_BSPLoad has returned the expected return value.
      </description>
      <or>
        <!-- If the "_expected_return_value" parameter is 0xFFFFFFFF,
             then check the "return" parameter is not __BioAPI_OK. (error check only) -->
        <and>
          <equal_to var1="_expected_return_value" var2="_indeterminate_error"/>
          <not_equal_to var1="return" var2="__BioAPI_OK"/>
        </and>
        <and>
          <not_equal_to var1="_expected_return_value" var2="_indeterminate_error"/>
          <equal_to var1="return" var2="_expected_return_value"/>
        </and>
      </or>
    </assert_condition>

    <!-- Invoke the activity Unload to the framework under test -->
    <invoke activity="BSPUnload" package="fb6ff5b0-7d5e-11de-8a39-0800200c9a66" break_on_break="true">
      <input name="Bspuuid" var="_bspuuid"/>
    </invoke>
  </activity>

  <!-- ******************************** -->
  <!-- Activity bound to BioSPI_BSPLoad -->
  <!-- ******************************** -->
  <activity name="SPI_BSPLoad">
    <input name="BSPUuid"/>
    <input name="BioAPINotifyCallback"/>
    <input name="BFPEnumerationHandler"/>
    <input name="MemoryFreeHandler"/>
    <input name="no_BSPUuid"/>
    <output name="return"/>

    <!-- check API=SPI -->
    <assert_condition response_if_false="fail">
      <equal_to var1="BSPUuid" var2="_bspuuid"/>
      <equal_to var1="no_BSPUuid" var2="_no_bspuuid"/>
    </assert_condition>
  </activity>
</package>
```

8.9 Assertion 1.6 – BioAPI_BSPUnload_And_BioSPI_BSPUnload

Description: This assertion calls BioAPI_BSPUnload with the input parameters described in Default Input Table and Test Condition Table, and checks if the framework under test returns BioAPI_OK or an error value in accordance with the description in Expected Result Table. The assertion is according to the following statement in the subclauses from BioAPI 2.0 standard document (ISO/IEC19784-1:2006), which are described in References in this subclause in this part of ISO/IEC 24709.

Excerpts:

Subclause 8.1.6

```
BioAPI_RETURN BioAPI BioAPI_BSPUnload
          (const BioAPI_UUID           *BSPUuid,
          BioAPI_EventHandler          AppNotifyCallback,
          Void                         *AppNotifyCallbackCtx);
```

The function de-registers event notification callbacks for the caller identified by BSPUuid. BioAPI_BSPUnload is the analogue call to BioAPI_BSPLoad. If all callbacks registered with BioAPI are removed, then BioAPI unloads (for that biometric application) the BSP that was loaded by calls to BioAPI_BSPLoad.

The BioAPI Framework uses the three input parameters; BSPUuid, AppNotifyCallback, and AppNotifyCallback Ctx to uniquely identify registered callbacks.

This function shall only be called (for a given BSP UUID) if there is at least one call to BioAPI_BSPLoad (for that BSP UUID) for which a corresponding call to this function has not yet been made.

This function should not be called by the application if there is a call to BioAPI_BSPAttach for which a corresponding call to BioAPI_BSPDetach (for a given BSP handle) has not yet been made. However, should this function be called while the BSP is still attached, then for each call to BioAPI_BSPAttach without a corresponding call to BioAPI_BSPDetach, the actions relative to the missing corresponding call to BioAPI_BSPDetach shall be implicitly performed by the BioAPI Framework (as though the corresponding function had been called at that time), followed by the actions relative to BioAPI_BSPUnload, (i.e., the Framework will detach the BSP prior to unloading it).

This includes the case in which the actions relative to a missing call to BioAPI_BSPUnload are implicitly performed by the BioAPI Framework during a call to BioAPI_Terminate (see clause 8.1.2 in ISO/IEC 19784-1:2006).

A BioAPI_RETURN value indicates success or specifies a particular error condition. The value BioAPI_OK indicates success. All other values represent an error condition.

Subclause 9.3.1.2

```
BioAPI_RETURN BioAPI BioSPI_BSPUnload
          (const BioAPI_UUID       *BSPUuid);
```

This function disables events and de-registers the event-notification function. The biometric service provider may perform cleanup operations, reversing the initialisation performed in BioSPI_BSPLoad.

NOTE This is a sister function to BioAPI_BSPUnload (see clause 8.1.6).

Return value

A BioAPI_RETURN value indicates success or specifies a particular error condition. The value BioAPI_OK indicates success. All other values represent an error condition.

References: 8.1.6, 9.3.1.2

Scenario:

a) The testing application does the followings:

1) Initialises the Framework under test, then installs and loads the testing BSP.

2) If the Test Number is 010605, calls BioAPI_BSPAttach.

3) Calls BioAPI_BSPUnload with the conditions given by the Default Input Table and by each row of the Test Condition Table.

b) The testing BSP does the followings:

1) If the Test Number is 010605, checks if the BioSPI_BSPDetach is called after the BioAPI_BSPUnload.

2) Checks the parameters and the contents of the pointer variables as described in the column 'BioSPI function' in the Expected Result Table.

3) Sets the return value of the BioSPI_BSPUnload given by each row of the Test Condition Table, then returns to the Framework.

c) The testing application does the followings:

1) Checks the return value of the BioAPI_BSPUnload and if it is not as described in the Expected Result Table, then issues a fail conformity response.

2) Terminate the Framework under test.

Parameters: The input parameters described below table are used for this assertion.

Table 28 — Default Input Table for BioAPI_BSPUnload_And_BioSPI_BSPUnload

Number	Input parameter name	Input parameter value (underscore:invalid)
1	BSPUuid	*Valid Uuid*
2	no_BSPUuid	false
3	AppNotifyCallback	*
4	AppNotifyCallbackCtx	*

NOTE: The parameter name "no_BSPUuid" is not defined in ISO/IEC 24709-1 so the implementation of this parameter depends on each CTS.

Table 29 — Test Condition Table for BioAPI_BSPUnload_And_BioSPI_BSPUnload

Test number	Input parameter name	Input parameter value (underscore:invalid)	Supported options in BSP Schema		Return value (from BioSPI)
			OPERATIONS_MASK	OPTIONS_MASK	
010601	BSPUuid	*Valid Uuid*	-	-	*OK*
010602	BSPUuid	*Invalid Uuid*	-	-	-
010603	no_BSPUuid	true	-	-	-
010604	AppNotifyCallback	0	-	-	*OK*
010605	AppNotifyCallbackCtx	0	-	-	*OK*

NOTE: The parameter name "no_BSPUuid" is not defined in ISO/IEC 24709-1 so the implementation of this parameter depends on each CTS.

Table 30 — Expected Result Table for BioAPI_BSPUnload_And_BioSPI_BSPUnload

Test number	BioSPI function (BioAPI/BioSPI parameter check)	BioAPI function			Other conditions
		Return value	Output parameter name	Output parameter value	
010601	X	*OK*	-	-	-
010602	-	*INVALID_UUID*	-	-	-
010603	-	*INVALID_UUID*	-	-	-
010604	X	*OK*	-	-	-
010605	X	*OK*	-	-	Check if the BioSPI_BSPDetach is called.

Assertion language package

```
<?xml version="1.0" encoding="UTF-8"?>
<package name="1067a9b0-792a-11de-8a39-0800200c9a66">
  <author>ISO/IEC JTC1 SC37</author>
  <description>This package contains the assertion "BioAPI_BSPUnload_And_BioSPI_BSPUnload".</description>

  <!-- ************** -->
  <!-- Test assertion -->
  <!-- ************** -->
  <assertion name="BioAPI_BSPUnload_And_BioSPI_BSPUnload" model="frameworkTesting">
    <description>
      This assertion checks if BioAPI_BSPUnload is called with input parameters as described in Test Condition Table,
      the framework under test returns BioAPI_OK or error value in accordance with the description in Expected
      Result Table.
    </description>

    <!-- Parameter given by the CTS by referring to the Test Condition Table. -->
    <input name="_testnumber"/>
    <input name="_operation"/>
    <input name="_option"/>
    <input name="_bspuuid" />
    <input name="_appnotifycallback"/>
    <input name="_appnotifycallbackctx"/>
    <input name="_no_bspuuid"/>

    <!-- Parameter given by the CTS by referring to the Expected Result Table. -->
    <input name="_expected_return_value"/>

    <!-- Invocation of the primary activity of this assertion. -->
    <invoke activity="BioAPI_BSPUnload_And_BioSPI_BSPUnload"/>

    <!-- Bind activities to check the BioSPI function invoked by the framework under test -->
    <bind activity="SPI_BSPUnload" function="BioSPI_BSPUnload"/>
    <bind activity="SPI_BSPDetachCheck" function="BioSPI_BSPDetach"/>
  </assertion>

  <!-- **************** -->
  <!-- Primary activity -->
  <!-- **************** -->
  <activity name="BioAPI_BSPUnload_And_BioSPI_BSPUnload">

    <!-- Flag _bspdetachcheck that shows called BioSPI_BSPDetach. -->
    <set name="_bspdetachcheck" value="0"/>

    <!-- Invoke the activity Load. -->
    <invoke activity="BSPLoad" package="fb6ff5b0-7d5e-11de-8a39-0800200c9a66" break_on_break="true">
      <input name="Bspuuid" var="_bspuuid" />
      <input name="Operationsmask" var="_operation" />
      <input name="Optionsmask" var="_option" />
    </invoke>

    <!-- If Test Number is 010605, Invoke the function BioAPI_BSPAttach. -->
    <set name="return" value="__BioAPI_OK"/>
    <invoke function="BioAPI_BSPAttach">
      <only_if>
        <not_equal_to var1="_testnumber" value2="010605"/>
      </only_if>
      <input name="BSPUuid" var="_bspuuid"/>
      <input name="Version" value="32" />
      <input name="NumUnits" value="0"/>
      <output name="NewBSPHandle" setvar="newbsphandle"/>
      <return setvar="return"/>
    </invoke>
    <assert_condition response_if_false="undecided" break_if_false="true">
      <description>
        The function BioAPI_BSPAttach has returned BioAPI_OK.
      </description>
      <equal_to var1="return" var2="__BioAPI_OK"/>
    </assert_condition>

    <!-- Invoke the function BioAPI_BSPUnload. -->
    <invoke function="BioAPI_BSPUnload">
      <input name="AppNotifyCallback" var="_appnotifycallback"/>
      <input name="AppNotifyCallbackCtx" var="_appnotifycallbackctx"/>
      <input name="BSPUuid" var="_bspuuid"/>
      <input name="no_BSPUuid" var="_no_bspuuid"/>
      <return setvar="return"/>
    </invoke>

    <!-- Extract the error code from the error value. -->
    <invoke activity="ExtractErrorCode" package="fb6ff5b0-7d5e-11de-8a39-0800200c9a66" break_on_break="true">
      <output name="Returnvalue" setvar="return"/>
    </invoke>

    <!-- Assertion -->
    <assert_condition response_if_false="fail">
      <description>
        The function BioAPI_BSPUnload has returned the expected return value.
      </description>
      <and>
        <or>
          <!-- If the "_expected_return_value" parameter is 0xFFFFFFFF,
               then check the "return" parameter is not __BioAPI_OK. (error check only) -->
          <and>
            <equal_to var1="_expected_return_value" var2="_indeterminate_error"/>
            <not_equal_to var1="return" var2="__BioAPI_OK"/>
          </and>
          <and>
            <not_equal_to var1="_expected_return_value" var2="_indeterminate_error"/>
            <equal_to var1="return" var2="_expected_return_value"/>
          </and>
        </or>
        <or>
          <not_equal_to var1="_testnumber" value2="010605"/>
          <and>
            <equal_to var1="_testnumber" value2="010605"/>
            <equal_to var1="_bspdetachcheck" value2="1"/>
          </and>
        </or>
      </and>
    </assert_condition>

    <!-- Invoke the activity Terminate to the framework under test -->
    <invoke activity="Termination" package="fb6ff5b0-7d5e-11de-8a39-0800200c9a66" break_on_break="true"/>
  </activity>

  <!-- ********************************** -->
  <!-- Activity bound to BioSPI_BSPUnload -->
  <!-- ********************************** -->
  <activity name="SPI_BSPUnload">
    <input name="BSPUuid"/>
    <input name="no_BSPUuid"/>
    <output name="return"/>

    <!-- check API=SPI -->
    <assert_condition response_if_false="fail">
      <equal_to var1="BSPUuid" var2="_bspuuid"/>
      <equal_to var1="no_BSPUuid" var2="_no_bspuuid"/>
    </assert_condition>
  </activity>

  <!-- ******************************************** -->
  <!-- Activity bound to BioSPI_BSPDetach to Check -->
  <!-- ******************************************** -->
  <activity name="SPI_BSPDetachCheck">
    <input name="BSPHandle"/>
    <input name="no_BSPUuid"/>
    <output name="return"/>

    <!-- Flag _bspdetachcheck that shows called BioSPI_BSPDetach. -->
    <assert_condition response_if_false="fail">
      <or>
        <not_equal_to var1="_testnumber" value2="010605"/>
        <and>
          <equal_to var1="_testnumber" value2="010605"/>
          <equal_to var1="_bspdetachcheck" value2="0"/>
        </and>
      </or>
    </assert_condition>
    <set name="_bspdetachcheck" value="1">
      <only_if>
        <equal_to var1="_testnumber" value2="010605"/>
      </only_if>
    </set>
  </activity>
</package>
```

8.10 Assertion 1.7 – BioAPI_BSPAttach_And_BioSPI_BSPAttach

Description: This assertion calls BioAPI_BSPAttach with the input parameters described in Default Input Table and Test Condition Table, and checks if the framework under test returns BioAPI_OK or an error value in accordance with the description in Expected Result Table. The assertion is according to the following statement in the subclauses from BioAPI 2.0 standard document (ISO/IEC19784-1:2006), which are described in References in this subclause in this part of ISO/IEC 24709.

Excerpts:

Subclause 8.1.7

```
BioAPI_RETURN BioAPI BioAPI_BSPAttach
          (const BioAPI_UUID                          *BSPUuid,
          BioAPI_VERSION                              Version,
          const BioAPI_UNIT_LIST_ELEMENT              *UnitList,
          uint32_t                                    NumUnits,
          BioAPI_HANDLE                               *NewBSPHandle);
```

This function initiates a BSP attach session and verifies that the version of the BSP expected by the application is compatible with the version on the system. The caller shall specify a list of zero or more BioAPI Units that the BSP is to use in the attach session being created.

This function shall only be called (for a given BSP UUID) if there is at least one call to BioAPI_BSPLoad (for that BSP UUID) for which a corresponding call to BioAPI_BSPUnload has not yet been made. The function BioAPI_BSPAttach can be invoked multiple times for each call to BioAPI_BSPLoad (prior to a call to BioAPI_BSPUnload), for the same BSP, creating multiple invocations of that BSP.

Subclause 9.3.1.3

```
BioAPI_RETURN BioAPI BioSPI_BSPAttach
          (const BioAPI_UUID                          *BSPUuid,
          BioAPI_VERSION                              Version,
          const BioAPI_UNIT_LIST_ELEMENT              *UnitList,
          uint32_t                                    NumUnits,
          BioAPI_HANDLE                               BSPHandle);
```

This function is invoked by the Framework once for each invocation of BioAPI_BSPAttach specifying the BSP identified by BSPUuid.

The biometric service provider shall verify compatibility with the version level specified by Version. If the version is not compatible, then this function fails. The BSP should perform all initialisations required to support the new BSP invocation.

The BSP shall attach the specified BioAPI Units if they are supported.

NOTE This is a sister function to BioAPI_BSPAttach (see clause 8.1.7).

A BioAPI_RETURN value indicates success or specifies a particular error condition. The value BioAPI_OK indicates success. All other values represent an error condition.

References: 8.1.7, 9.3.1.3

Scenario:

a) The testing application does the followings:

1) Initialises the Framework under test, then installs, loads the testing BSP.
2) Calls BioAPI_BSPAttach with the conditions given by the Default Input Table and by each row of the Test Condition Table.

b) The testing BSP does the followings:

1) Checks the parameters and the contents of the pointer variables as described in the column 'BioSPI function' in the Expected Result Table.
2) Sets the return value of the BioSPI_BSPAttach given by each row of the Test Condition Table, then returns to the Framework.

c) The testing application does the followings:

1) Checks the return value of the BioAPI_BSPAttach and if it is not as described in the Expected Result Table, then issues a fail conformity response.
2) Checks the output parameters and the contents of the pointer variables and if either of them is not as described in the column 'BioAPI function' in the Expected Result Table, then issues a fail conformity response.
3) Detaches, unloads and uninstalls the testing BSP, then terminates the Framework under test.

Parameters: The input parameters described below table are used for this assertion.

Table 31 — Default Input Table for BioAPI_BSPAttach_And_BioSPI_BSPAttach

Number	Input parameter name	Input parameter value (underscore:invalid)
1	BSPUuid	*Valid Uuid*
2	no_BSPUuid	false
3	Version	32
4	UnitList_1_UnitCategory	N/A
5	UnitList_1_UnitID	N/A
6	no_UnitList	false
7	NumUnits	0
8	no_NewBSPHandle	false

NOTE: The parameter names "no_BSPUuid" and "no_UnitList" are not defined in ISO/IEC 24709-1 so the implementation of this parameter depends on each CTS.

Table 32 — Test Condition Table for BioAPI_BSPAttach_And_BioSPI_BSPAttach

Test number	Input parameter name	Input parameter value (underscore:invalid)	Supported options in BSP Schema		Return value (from BioSPI)
			OPERATIONS_MASK	OPTIONS_MASK	
010701	BSPUuid	*Valid Uuid*	-	-	*OK*
010702	BSPUuid	*Invalid Uuid*	-	-	-
010703	no_BSPUuid	true	-	-	-
010704	Version	16	-	-	INCOMPATIBLE_VERSION
010705	no_UnitList	true	-	-	INVALID_INPUT_POINTER
010706	NumUnits	100	-	-	*INVALID_DATA*
010707	no_NewBSPHandle	true	-	-	*INVALID_INPUT_POINTER*

Table 33 — Expected Result Table for BioAPI_BSPAttach_And_BioSPI_BSPAttach

Test number	BioSPI function (BioAPI/BioSPI parameter check)	BioAPI function			Other conditions
		Return value	Output parameter name	Output parameter value	
010701	X	*OK*	-	-	-
010702	-	*INVALID_UUID*	-	-	-
010703	-	*INVALID_UUID*	-	-	-
010704	X	*INCOMPATIBLE_VERSION*	-	-	-
010705	X	*indeterminate error*	-	-	-
010706	X	*indeterminate error*	-	-	-
010707	X	*indeterminate error*	-	-	-

Assertion language package

```
<?xml version="1.0" encoding="UTF-8"?>
<package name="2ae45d10-792a-11de-8a39-0800200c9a66">
  <author>ISO/IEC JTC1 SC37</author>
  <description>This package contains the assertion "BioAPI_BSPAttach_And_BioSPI_BSPAttach".</description>

  <!-- ************** -->
  <!-- Test assertion -->
  <!-- ************** -->
  <assertion name="BioAPI_BSPAttach_And_BioSPI_BSPAttach" model="frameworkTesting">
    <description>
      This assertion checks if BioAPI_BSPAttach is called with input parameters as described in Test Condition Table,
      the framework under test returns BioAPI_OK or error value in accordance with the description in Expected
      Result Table.
    </description>
    <!-- Parameter given by the CTS by referring to the Test Condition Table. -->
    <input name="_testnumber"/>
    <input name="_operationsmask"/>
    <input name="_optionsmask"/>
    <input name="_bspuuid"/>
    <input name="_version"/>
    <input name="_no_newbsphandle"/>
    <input name="_no_bspuuid"/>
    <input name="_no_unitlist"/>

    <!-- Parameter given by the CTS by referring to the Expected Result Table. -->
    <input name="_expected_return_value"/>

    <!-- Invocation of the primary activity of this assertion. -->
    <invoke activity="BioAPI_BSPAttach_And_BioSPI_BSPAttach"/>

    <!-- Bind activities to check the BioSPI function invoked by the framework under test -->
    <bind activity="SPI_BSPAttach" function="BioSPI_BSPAttach"/>
  </assertion>

  <!-- **************** -->
  <!-- Primary activity -->
  <!-- **************** -->
  <activity name="BioAPI_BSPAttach_And_BioSPI_BSPAttach">

    <!-- Invoke the activity BSPLoad the framework under test -->
    <invoke activity="BSPLoad" package="fb6ff5b0-7d5e-11de-8a39-0800200c9a66" break_on_break="true">
      <input name="Operationsmask" var="_operationsmask"/>
      <input name="Optionsmask" var="_optionsmask"/>
      <input name="Bspuuid" var="_bspuuid"/>
    </invoke>

    <!-- Invoke the function BioAPI_BSPAttach. -->
    <invoke function="BioAPI_BSPAttach">
      <input name="BSPUuid" var="_bspuuid"/>
      <input name="Version" var="_version"/>
      <input name="NumUnits" value="0"/>
      <input name="no_NewBSPHandle" var="_no_newbsphandle"/>
      <input name="no_BSPUuid" var="_no_bspuuid"/>
      <input name="no_UnitList" var="_no_unitlist"/>
      <output name="NewBSPHandle" setvar="bsphandle"/>
      <return setvar="return"/>
    </invoke>

    <!-- Extract the error code from the error value. -->
    <invoke activity="ExtractErrorCode" package="fb6ff5b0-7d5e-11de-8a39-0800200c9a66" break_on_break="true">
      <output name="Returnvalue" setvar="return"/>
    </invoke>

    <!-- Assertion -->
    <assert_condition response_if_false="fail">
      <description>
        The function BioAPI_BSPAttach has returned the expected return value.
      </description>
      <or>
        <!-- If the "_expected_return_value" parameter is 0xFFFFFFFF,
             then check the "return" parameter is not __BioAPI_OK. (error check only) -->
        <and>
          <equal_to var1="_expected_return_value" var2="_indeterminate_error"/>
          <not_equal_to var1="return" var2="__BioAPI_OK"/>
        </and>
        <and>
          <not_equal_to var1="_expected_return_value" var2="_indeterminate_error"/>
          <equal_to var1="return" var2="_expected_return_value"/>
        </and>
      </or>
    </assert_condition>

    <!-- Invoke the functions Detach -->
    <invoke activity="BSPDetach" package="fb6ff5b0-7d5e-11de-8a39-0800200c9a66" break_on_break="true">
      <input name="Bsphandle" var="bsphandle"/>
      <input name="Bspuuid" var="_bspuuid"/>
    </invoke>
  </activity>

  <!-- ********************************** -->
  <!-- Activity bound to BioSPI_BSPAttach -->
  <!-- ********************************** -->
  <activity name="SPI_BSPAttach">
    <input name="BSPUuid"/>
    <input name="Version"/>
    <input name="Unit_1_UnitCategory"/>
    <input name="Unit_1_UnitID"/>
    <input name="Unit_2_UnitCategory"/>
    <input name="Unit_2_UnitID"/>
    <input name="Unit_3_UnitCategory"/>
    <input name="Unit_3_UnitID"/>
    <input name="Unit_4_UnitCategory"/>
    <input name="Unit_4_UnitID"/>
    <input name="NumUnits"/>
    <input name="BSPHandle"/>
    <input name="no_BSPUuid"/>
    <input name="no_UnitList"/>
    <output name="return"/>

    <!-- check API=SPI -->
    <assert_condition response_if_false="fail">
      <equal_to var1="BSPUuid" var2="_bspuuid"/>
      <equal_to var1="Version" var2="_version"/>
      <equal_to var1="NumUnits" var2="_numunits"/>
      <equal_to var1="BSPHandle" var2="_bsphandle"/>
      <equal_to var1="no_BSPUuid" var2="_no_bspuuid"/>
      <equal_to var1="no_UnitList" var2="_no_unitlist"/>
    </assert_condition>
  </activity>
</package>
```

8.11 Assertion 1.8 – BioAPI_BSPDetach_And_BioSPI_BSPDetach

Description: This assertion calls BioAPI_BSPDetach with the input parameters described in Default Input Table and Test Condition Table, and checks if the framework under test returns BioAPI_OK or an error value in accordance with the description in Expected Result Table. The assertion is according to the following statement in the subclauses from BioAPI 2.0 standard document (ISO/IEC19784-1:2006), which are described in References in this subclause in this part of ISO/IEC 24709.

Excerpts:

Subclause 8.1.8

BioAPI_RETURN BioAPI BioAPI_BSPDetach
        (BioAPI_HANDLE   BSPHandle);

This function detaches the biometric application from the BSP invocation.

At this time, all BSP allocated resources associated with the BSP attach session shall be freed or released or invalidated. This is especially important for BIR, BSP, and database handles. At this time, all set callbacks associated with the BSP attach session shall become invalid.

This function shall only be called after BioAPI_BSPAttach has been called, and shall not be called more than once for the same BSP handle created by the call to BioAPI_BSPAttach.

Return value

A BioAPI_RETURN value indicates success or specifies a particular error condition. The value BioAPI_OK indicates success. All other values represent an error condition.

Subclause 9.3.1.4

BioAPI_RETURN BioAPI BioSPI_BSPDetach
        (BioAPI_HANDLE   BSPHandle);

References: 8.1.8, 9.3.1.4

Scenario:

a) The testing application does the followings:

1) Initialises the Framework under test, then installs, loads and attaches the testing BSP.

2) Calls BioAPI_BSPDetach with the conditions given by the Default Input Table and by each row of the Test Condition Table.

b) The testing BSP does the followings:

1) Checks the parameters and the contents of the pointer variables as described in the column 'BioSPI function' in the Expected Result Table.
2) Sets the return value of the BioSPI_BSPDetach given by each row of the Test Condition Table, then returns to the Framework.

c) The testing application does the followings:

1) Checks the return value of the BioAPI_BSPDetach and if it is not as described in the Expected Result Table, then issues a fail conformity response.
2) Unloads and uninstalls the testing BSP, then terminates the Framework under test.

Parameters: The input parameters described below table are used for this assertion.

Table 34 — Default Input Table for BioAPI_BSPDetach_And_BioSPI_BSPDetach

Number	Input parameter name	Input parameter value (underscore:invalid)
1	BSPHandle	*Valid BSPHandle*

Table 35 — Test Condition Table for BioAPI_BSPDetach_And_BioSPI_BSPDetach

Test number	Input parameter name	Input parameter value (underscore:invalid)	Supported options in BSP Schema		Return value (from BioSPI)
			OPERATIONS_MASK	OPTIONS_MASK	
010801	BSPHandle	*Valid BSPHandle*	-	-	*OK*
010802	BSPHandle	*<u>Invalid BSPHandle</u>*	-	-	-

Table 36 — Expected Result Table for BioAPI_BSPDetach_And_BioSPI_BSPDetach

Test number	BioSPI function (BioAPI/BioSPI parameter check)	BioAPI function			Other conditions
		Return value	Output parameter name	Output parameter value	
010801	X	*OK*	-	-	-
010802	-	*INVALID_BSP_HANDLE*	-	-	-

Assertion language package

```
<?xml version="1.0" encoding="UTF-8"?>
<package name="4149b370-792a-11de-8a39-0800200c9a66">
  <author>ISO/IEC JTC1 SC37</author>
  <description>This package contains the assertion "BioAPI_BSPDetach_And_BioSPI_BSPDetach".</description>

  <!-- ************** -->
  <!-- Test assertion -->
  <!-- ************** -->
  <assertion name="BioAPI_BSPDetach_And_BioSPI_BSPDetach" model="frameworkTesting">
    <description>
      This assertion checks if BioAPI_BSPDetach is called with input parameters as described in Test Condition Table,
      the framework under test returns BioAPI_OK or error value in accordance with the description in Expected
      Result Table.
    </description>

    <!-- Parameter given by the CTS by referring to the Test Condition Table. -->
    <input name="_testnumber"/>
    <input name="_operationsmask"/>
    <input name="_optionsmask"/>

    <!-- Parameter given by the CTS by referring to the Expected Result Table. -->
    <input name="_expected_return_value"/>

    <!-- Parameter given by the CTS. -->
    <input name="_bspuuid"/>

    <!-- Invocation of the primary activity of this assertion. -->
    <invoke activity="BioAPI_BSPDetach_And_BioSPI_BSPDetach"/>

    <!-- Bind activities to check the BioSPI function invoked by the framework under test -->
    <bind activity="SPI_BSPDetach" function="BioSPI_BSPDetach"/>
  </assertion>

  <!-- **************** -->
  <!-- Primary activity -->
  <!-- **************** -->
  <activity name="BioAPI_BSPDetach_And_BioSPI_BSPDetach">

    <!-- Invoke the activity BSP Attach the framework under test -->
    <invoke activity="BSPAttach" package="fb6ff5b0-7d5e-11de-8a39-0800200c9a66" break_on_break="true">
      <input name="Operationsmask" var="_operationsmask"/>
      <input name="Optionsmask" var="_optionsmask"/>
      <input name="Bspuuid" var="_bspuuid"/>
      <output name="Newbsphandle" setvar="_bsphandle"/>
    </invoke>

    <!-- If Test Number is 010802, make _bsphandle to an illegal value. -->
    <add name="_bsphandle" value="1">
      <only_if><equal_to var1="_testnumber" value2="010802"/></only_if>
    </add>

    <!-- Invoke the function BioAPI_BSPDetach. -->
    <invoke function="BioAPI_BSPDetach">
      <input name="BSPHandle" var="_bsphandle"/>
      <return setvar="return"/>
    </invoke>

    <!-- Extract the error code from the error value. -->
    <invoke activity="ExtractErrorCode" package="fb6ff5b0-7d5e-11de-8a39-0800200c9a66" break_on_break="true">
      <output name="Returnvalue" setvar="return"/>
    </invoke>

    <!-- Assertion -->
    <assert_condition response_if_false="fail">
      <description>
        The function BioAPI_BSPDetach has returned the expected return value.
      </description>
      <or>
        <!-- If the "_expected_return_value" parameter is 0xFFFFFFFF,
             then check the "return" parameter is not __BioAPI_OK. (error check only) -->
        <and>
          <equal_to var1="_expected_return_value" var2="_indeterminate_error"/>
          <not_equal_to var1="return" var2="__BioAPI_OK"/>
        </and>
        <and>
          <not_equal_to var1="_expected_return_value" var2="_indeterminate_error"/>
          <equal_to var1="return" var2="_expected_return_value"/>
        </and>
      </or>
    </assert_condition>

    <!-- Invoke the activity BSPUnload the framework under test -->
    <invoke activity="BSPUnload" package="fb6ff5b0-7d5e-11de-8a39-0800200c9a66" break_on_break="true">
      <input name="Bspuuid" var="_bspuuid"/>
    </invoke>
  </activity>

  <!-- ********************************** -->
  <!-- Activity bound to BioSPI_BSPDetach -->
  <!-- ********************************** -->
  <activity name="SPI_BSPDetach">
    <input name="BSPHandle"/>
    <output name="return"/>

    <!-- check API=SPI -->
    <assert_condition response_if_false="fail">
      <equal_to var1="BSPHandle" var2="_bsphandle"/>
    </assert_condition>
  </activity>
</package>
```

8.12 Assertion 1.9 – BioAPI_QueryUnits_And_BioSPI_QueryUnits

Description: This assertion calls BioAPI_QueryUnits with the input parameters described in Default Input Table and Test Condition Table, and checks if the framework under test returns BioAPI_OK or an error value in accordance with the description in Expected Result Table. The assertion is according to the following statement in the subclauses from BioAPI 2.0 standard document (ISO/IEC19784-1:2006), which are described in References in this subclause in this part of ISO/IEC 24709.

Excerpts:

Subclause 8.1.9

```
BioAPI_RETURN BioAPI BioAPI_QueryUnits
        (const BioAPI_UUID          *BSPUuid,
        BioAPI_UNIT_SCHEMA          **UnitSchemaArray,
        uint32_t                    *NumberOfElements);
```

This function provides information about all BioAPI Units that are managed directly or indirectly by the BSP identified by the given BSP UUID and are currently in the inserted state. It performs the following actions (in order):

a) determines the set of BioAPI Units that are managed directly or indirectly by the BSP and are currently in the inserted state;
b) allocates a memory block large enough to contain an array of elements of type BioAPI_UNIT_SCHEMA with as many elements as the number of BioAPI Units determined in (a);
c) fills the array with the unit schemas of all BioAPI Units determined in (a); and
d) returns the address of the array in the UnitSchemaArray parameter and the size of the array in the NumberOfElements parameter.

NOTE When the Framework calls the function BioSPI_QueryUnits of a BSP, the BSP allocates memory for the data to be returned to the Framework. In some implementation architectures, the Framework will simply pass to the application the data and the addresses exactly as returned by the BSP because the application will interpret the addresses in the same way as the BSP and will be able to access the data that the BSP has placed at those addresses. In other implementation architectures, the framework will need to move all the data returned by the BSP to newly allocated memory blocks accessible to the application, and will call BioSPI_Free after copying each memory block but before returning from the BioAPI_QueryUnits call. In the former case, when the application calls BioAPI_Free, the Framework will make a corresponding call to BioSPI_Free. In the latter case, the calls to BioAPI_Free will be handled internally by the framework. However, such differences in the behavior of the Framework are not visible to the application.

The memory block containing the array shall be freed by the application via a call to BioAPI_Free (see clause 8.7.2) when it is no longer needed by the application. The memory block pointed to by the UnitProperties member within each element of the array shall also be freed by the application via a call to BioAPI_Free when it is no longer needed by the application.

This function shall only be called after BioAPI_BSPLoad has been called for the specified BSP, and shall not be called after BioAPI_BSPUnload has been called for the BSP.

A BioAPI_RETURN value indicates success or specifies a particular error condition. The value BioAPI_OK indicates success. All other values represent an error condition.

Subclause 9.3.1.5

```
BioAPI_RETURN BioAPI BioSPI_QueryUnits
        (const BioAPI_UUID          *Uuid,
        BioAPI_UNIT_SCHEMA          **UnitSchemaArray,
        uint32_t                    *NumberOfElements);
```

References: 8.1.9, 9.3.1.5

Scenario:

a) The testing application does the followings:

1) Initialises the Framework under test, then installs and loads the testing BSP.
2) Calls BioAPI_QueryUnits with the conditions given by the Default Input Table and by each row of the Test Condition Table.

b) The testing BSP does the followings:

1) Checks the parameters and the contents of the pointer variables as described in the column 'BioSPI function' in the Expected Result Table.

2) Sets the return value of the BioSPI_QueryUnits given by each row of the Test Condition Table, then returns to the Framework.

c) The testing application does the followings:

1) Checks the return value of the BioAPI_QueryUnits and if it is not as described in the Expected Result Table, then issues a fail conformity response.

2) Calls BioAPI_Free.

d) If the Test Number is 010901, the testing BSP checks if the BioSPI_Free is called only once.

e) The testing application unloads and uninstalls the testing BSP, then terminates the Framework under test.

Parameters: The input parameters described below table are used for this assertion.

Table 37 — Default Input Table for BioAPI_QueryUnits_And_BioSPI_QueryUnits

Number	Input parameter name	Input parameter value (underscore:invalid)
1	BSPUuid	*Valid Uuid*
2	no_BSPUuid	false
3	no_UnitSchemaArray	false
4	no_NumberOfElements	false

NOTE: The parameter name "no_BSPUuid" is not defined in ISO/IEC 24709-1 so the implementation of this parameter depends on each CTS.

Table 38 — Test Condition Table for BioAPI_QueryUnits_And_BioSPI_QueryUnits

Test number	Input parameter name	Input parameter value (underscore:invalid)	Supported options in BSP Schema		Return value (from BioSPI)
			OPERATIONS_MASK	OPTIONS_MASK	
010901	BSPUuid	*Valid Uuid*	*QueryUnits*	-	*OK*
010902	BSPUuid	*Invalid Uuid*	*QueryUnits*	-	-
010903	no_BSPUuid	true	*QueryUnits*	-	-
010904	no_UnitSchemaArray	true	*QueryUnits*	-	*INVALID_OUTPUT_POINTER*
010905	no_NumberOfElements	true	*QueryUnits*	-	*INVALID_OUTPUT_POINTER*
010906	BSPUuid	*Valid Uuid*	-	-	-

NOTE: The parameter name "no_BSPUuid" is not defined in ISO/IEC 24709-1 so the implementation of this parameter depends on each CTS.

Table 39 — Expected Result Table for BioAPI_QueryUnits_And_BioSPI_QueryUnits

Test number	BioSPI function (BioAPI/BioSPI parameter check)	BioAPI function			Other conditions
		Return value	Output parameter name	Output parameter value	
010901	X	*OK*	-	-	Check if the BioSPI_Free is called.
010902	-	*INVALID_UUID*	-	-	-
010903	-	*INVALID_UUID*	-	-	-
010904	X	*indeterminate error*	-	-	-
010905	X	*indeterminate error*	-	-	-
010906	-	*indeterminate error*	-	-	-

Assertion language package

```
<?xml version="1.0" encoding="UTF-8"?>
<package name="507a4030-792a-11de-8a39-0800200c9a66">
  <author>ISO/IEC JTC1 SC37</author>
  <description>This package contains the assertion "BioAPI_QueryUnits_And_BioSPI_QueryUnits".</description>

  <!-- ************** -->
  <!-- Test assertion -->
  <!-- ************** -->
  <assertion name="BioAPI_QueryUnits_And_BioSPI_QueryUnits" model="frameworkTesting">
    <description>
      This assertion checks if BioAPI_QueryUnits is called with input parameters as described in
      Test Condition Table, the framework under test returns BioAPI_OK or error value in accordance with
      the description in Expected Result Table.
    </description>

    <!-- Parameter given by the CTS by referring to the Test Condition Table. -->
    <input name="_testnumber"/>
    <input name="_operation"/>
    <input name="_option"/>
    <input name="_bspuuid"/>
    <input name="_no_unitschemaarray"/>
    <input name="_no_numberofelements"/>
    <input name="_no_bspuuid"/>

    <!-- Parameter given by the CTS by referring to the Expected Result Table. -->
    <input name="_expected_return_value"/>

    <!-- Invocation of the primary activity of this assertion. -->
    <invoke activity="BioAPI_QueryUnits_And_BioSPI_QueryUnits"/>

    <!-- Bind activities to check the BioSPI function invoked by the framework under test -->
    <bind activity="SPI_QueryUnits" function="BioSPI_QueryUnits"/>
    <bind activity="SPI_FreeCheck" function="BioSPI_Free"/>
  </assertion>

  <!-- **************** -->
  <!-- Primary activity -->
  <!-- **************** -->
  <activity name="BioAPI_QueryUnits_And_BioSPI_QueryUnits">

    <!-- Flag _freecheck that shows called BioSPI_Free. -->
    <set name="_freecheck" value="0"/>

    <!-- Invoke the activity Load. -->
    <invoke activity="BSPLoad" package="fb6ff5b0-7d5e-11de-8a39-0800200c9a66" break_on_break="true">
      <input name="Bspuuid" var="_bspuuid" />
      <input name="Operationsmask" var="_operation" />
      <input name="Optionsmask" var="_option" />
    </invoke>

    <!-- Invoke the function BioAPI_QueryUnits. -->
    <invoke function="BioAPI_QueryUnits">
      <input name="BSPUuid" var="_bspuuid"/>
      <input name="no_UnitSchemaArray" var="_no_unitschemaarray"/>
      <input name="no_NumberOfElements" var="_no_numberofelements"/>
      <input name="no_BSPUuid" var="_no_bspuuid"/>
      <output name="UnitSchema_1_BSPUuid" setvar="us_1_bspuuid"/>
      <output name="UnitSchema_1_UnitManagerUuid" setvar="us_1_unitmanageruuid"/>
      <output name="UnitSchema_1_UnitID" setvar="us_1_unitid"/>
      <output name="UnitSchema_1_UnitCategory" setvar="us_1_unitcategory"/>
      <output name="UnitSchema_1_UnitProperties" setvar="us_1_unitproperties"/>
      <output name="UnitSchema_1_VendorInformation" setvar="us_1_vendorinformation"/>
      <output name="UnitSchema_1_EventNotifyInsert" setvar="us_1_eventnotifyinsert"/>
      <output name="UnitSchema_1_EventNotifyRemove" setvar="us_1_eventnotifyremove"/>
      <output name="UnitSchema_1_EventNotifyFault" setvar="us_1_eventnotifyfault"/>
      <output name="UnitSchema_1_EventNotifySourcePresent" setvar="us_1_eventnotifysourcepresent"/>
      <output name="UnitSchema_1_EventNotifySourceRemoved" setvar="us_1_eventnotifysourceremoved"/>
      <output name="UnitSchema_1_UnitPropertyID" setvar="us_1_unitpropertyid"/>
      <output name="UnitSchema_1_UnitProperty" setvar="us_1_unitproperty"/>
      <output name="UnitSchema_1_HardwareVersion" setvar="us_1_hardwareversion"/>
      <output name="UnitSchema_1_FirmwareVersion" setvar="us_1_firmwareversion"/>
      <output name="UnitSchema_1_SoftwareVersion" setvar="us_1_softwareversion"/>
      <output name="UnitSchema_1_HardwareSerialNumber" setvar="us_1_hardwareserialnumber"/>
      <output name="UnitSchema_1_AuthenticatedHardware" setvar="us_1_authenticatedhardware"/>
      <output name="UnitSchema_1_MaxBSPDbSize" setvar="us_1_maxbspdbsize"/>
      <output name="UnitSchema_1_MaxIdentify" setvar="us_1_maxidentify"/>
      <output name="NumberOfElements" setvar="numberofelements"/>
      <return setvar="return"/>
    </invoke>

    <!-- Extract the error code from the error value. -->
    <invoke activity="ExtractErrorCode" package="fb6ff5b0-7d5e-11de-8a39-0800200c9a66" break_on_break="true">
      <output name="Returnvalue" setvar="return"/>
    </invoke>

    <!-- Assertion -->
    <assert_condition response_if_false="fail">
      <description>
        The function BioAPI_QueryUnits has returned the expected return value.
      </description>
      <and>
        <or>
          <!-- If the "_expected_return_value" parameter is 0xFFFFFFFF,
               then check the "return" parameter is not __BioAPI_OK. (error check only) -->
          <and>
            <equal_to var1="_expected_return_value" var2="_indeterminate_error"/>
            <not_equal_to var1="return" var2="__BioAPI_OK"/>
          </and>
          <and>
            <not_equal_to var1="_expected_return_value" var2="_indeterminate_error"/>
            <equal_to var1="return" var2="_expected_return_value"/>
          </and>
        </or>
        <or>
          <not_equal_to var1="_testnumber" value2="010901"/>
          <and>
            <equal_to var1="_testnumber" value2="010901"/>
            <equal_to var1="_freecheck" value2="1"/>
          </and>
        </or>
      </and>
    </assert_condition>

    <!-- Invoke the activity BSPUnload the framework under test -->
    <invoke activity="BSPUnload" package="fb6ff5b0-7d5e-11de-8a39-0800200c9a66" break_on_break="true">
      <input name="Bspuuid" var="_bspuuid"/>
    </invoke>

 </activity>

  <!-- *********************************** -->
  <!-- Activity bound to BioSPI_QueryUnits -->
  <!-- *********************************** -->
  <activity name="SPI_QueryUnits">
    <input name="BSPUuid"/>
    <input name="no_UnitSchemaArray"/>
    <input name="no_NumberOfElements"/>
    <input name="no_BSPUuid"/>
    <output name="UnitSchema_1_BSPUuid"/>
    <output name="UnitSchema_1_UnitManagerUuid"/>
    <output name="UnitSchema_1_UnitID"/>
    <output name="UnitSchema_1_UnitCategory"/>
    <output name="UnitSchema_1_UnitProperties"/>
    <output name="UnitSchema_1_VendorInformation"/>
    <output name="UnitSchema_1_EventNotifyInsert"/>
    <output name="UnitSchema_1_EventNotifyRemove"/>
    <output name="UnitSchema_1_EventNotifyFault"/>
    <output name="UnitSchema_1_EventNotifySourcePresent"/>
    <output name="UnitSchema_1_EventNotifySourceRemoved"/>
    <output name="UnitSchema_1_UnitPropertyID"/>
    <output name="UnitSchema_1_UnitProperty"/>
    <output name="UnitSchema_1_HardwareVersion"/>
    <output name="UnitSchema_1_FirmwareVersion"/>
    <output name="UnitSchema_1_SoftwareVersion"/>
    <output name="UnitSchema_1_HardwareSerialNumber"/>
    <output name="UnitSchema_1_AuthenticatedHardware"/>
    <output name="UnitSchema_1_MaxBSPDbSize"/>
    <output name="UnitSchema_1_MaxIdentify"/>
    <output name="NumberOfElements"/>
    <output name="return"/>

    <!-- check API=SPI -->
    <assert_condition response_if_false="fail">
      <equal_to var1="BSPUuid" var2="_bspuuid"/>
      <equal_to var1="no_UnitSchemaArray" var2="_no_unitschemaarray"/>
      <equal_to var1="no_NumberOfElements" var2="_no_numberofelements"/>
      <equal_to var1="no_NumberOfElements" var2="_no_numberofelements"/>
      <equal_to var1="no_BSPUuid" var2="_no_bspuuid"/>
    </assert_condition>
  </activity>

  <!-- ************************************** -->
  <!-- Activity bound to BioSPI_Free to check -->
  <!-- ************************************** -->
  <activity name="SPI_FreeCheck">
    <input name="Ptr"/>
    <output name="return"/>

    <!-- Flag _freecheck that shows called BioSPI_BSPFree. -->
    <assert_condition response_if_false="fail">
      <or>
        <not_equal_to var1="_testnumber" value2="010901"/>
        <and>
          <equal_to var1="_testnumber" value2="010901"/>
          <equal_to var1="_freecheck" value2="0"/>
        </and>
      </or>
    </assert_condition>
    <set name="_freecheck" value="1">
      <only_if>
        <equal_to var1="_testnumber" value2="010901"/>
      </only_if>
    </set>
  </activity>

</package>
```

8.13 Assertion 1.10 – BioAPI_EnumBFPs

Description: This assertion calls BioAPI_EnumBFPs with the input parameters described in Default Input Table and Test Condition Table, and checks if the framework under test returns BioAPI_OK or an error value in accordance with the description in Expected Result Table. The assertion is according to the following statement in the subclauses from BioAPI 2.0 standard document (ISO/IEC19784-1:2006), which are described in References in this subclause in this part of ISO/IEC 24709.

Excerpts:

Subclause 8.1.10

```
BioAPI_RETURN BioAPI BioAPI_EnumBFPs
          (BioAPI_BFP_SCHEMA       **BFPSchemaArray,
          uint32_t                 *NumberOfElements);
```

This function provides information about all BSPs currently installed in the component registry. It performs the following actions (in order):

a) allocates a memory block large enough to contain an array of elements of type BioAPI_BSP_SCHEMA with as many elements as the number of installed BSPs; .
b) fills the array with the BSP schemas of all installed BSPs; and
c) returns the address of the array in the BSPSchemaArray parameter and the number of elements of the array in the NumberOfElements parameter.

This function shall only be called if there is at least one call to BioAPI_Init for which a corresponding call to BioAPI_Terminate has not yet been made.

This function is handled internally within the BioAPI Framework and is not passed through to any BSP.

The memory block containing the array shall be freed by the application via a call to BioAPI_Free (see clause 8.7.2) when it is no longer needed by the application.

The memory block pointed to by the Path member within each element of the array shall also be freed by the application via a call to BioAPI_Free when it is no longer needed by the application.

A BioAPI_RETURN value indicates success or specifies a particular error condition. The value BioAPI_OK indicates success. All other values represent an error condition.

References: 8.1.10

Scenario:
The testing application does the followings:

a) Initialises the Framework under test, then installs the testing BFP.
b) Calls BioAPI_EnumBFPs with the conditions given by the Default Input Table and by each row of the Test Condition Table.
c) Checks the return value of the BioAPI_EnumBFPs and if it is not as described in the Expected Result Table, then issues a fail conformity response.
d) Checks the output parameters and the contents of the pointer variables and if either of them is not as described in the column 'BioAPI function' in the Expected Result Table, then issues a fail conformity response.
e) Uninstalls the testing BFP, then terminates the Framework under test.

Parameters: The input parameters described below table are used for this assertion.

Table 40 — Default Input Table for BioAPI_EnumBFPs

Number	Input parameter name	Input parameter value (underscore:invalid)
1	no_BFPSchemaArray	false
2	no_NumberOfElements	false

Table 41 — Test Condition Table for BioAPI_EnumBFPs

Test number	Input parameter name	Input parameter value (underscore:invalid)	Supported options in BSP Schema		Return value (from BioSPI)
			OPERATIONS_MASK	OPTIONS_MASK	
011001	no_BFPSchemaArray	false	-	-	-
011002	no_BFPSchemaArray	true	-	-	-
011003	no_NumberOfElements	true	-	-	-

Table 42 — Expected Result Table for BioAPI_EnumBFPs

Test number	BioSPI function (BioAPI/BioSPI parameter check)	BioAPI function			Other conditions
		Return value	Output parameter name	Output parameter value	
011001	-	*OK*	-	-	-
011002	-	*indeterminate error*	-	-	-
011003	-	*indeterminate error*	-	-	-

Assertion language package

```
<?xml version="1.0" encoding="UTF-8"?>
<package name="62eb03d0-792a-11de-8a39-0800200c9a66">
  <author>ISO/IEC JTC1 SC37</author>
  <description>This package contains the assertion "BioAPI_EnumBFPs".</description>

  <!-- ************** -->
  <!-- Test assertion -->
  <!-- ************** -->
  <assertion name="BioAPI_EnumBFPs" model="frameworkTesting">
    <description>
      This assertion checks if BioAPI_EnumBFPs is called with input parameters as described in Test Condition Table,
      the framework under test returns BioAPI_OK or error value in accordance with the description in Expected
      Result Table.
    </description>

    <!-- Parameter given by the CTS by referring to the Test Condition Table. -->
    <input name="_testnumber"/>
    <input name="_no_bfpschemaarray"/>
    <input name="_no_numberofelements"/>

    <!-- Parameter given by the CTS by referring to the Expected Result Table. -->
    <input name="_expected_return_value"/>

    <!-- Invocation of the primary activity of this assertion. -->
    <invoke activity="BioAPI_EnumBFPs"/>
  </assertion>

  <!-- **************** -->
  <!-- Primary activity -->
  <!-- **************** -->
  <activity name="BioAPI_EnumBFPs">

    <!-- Invoke the activity Initialisation to initialise the framework under test -->
    <invoke activity="Initialisation" package="fb6ff5b0-7d5e-11de-8a39-0800200c9a66" break_on_break="true"/>

    <!-- Invoke the function BioAPI_EnumBFPs. -->
    <invoke function="BioAPI_EnumBFPs">
      <input name="no_BFPSchemaArray" var="_no_bfpschemaarray"/>
      <input name="no_NumberOfElements" var="_no_numberofelements"/>
      <output name="BFPSchema_1_BFPUuid" setvar="bfps_1_bfpschema_1_bfpuuid"/>
      <output name="BFPSchema_1_BFPCategory" setvar="bfps_1_bfpschema_1_bfpcategory"/>
      <output name="BFPSchema_1_BFPDescription" setvar="bfps_1_bfpschema_1_bfpdescription"/>
      <output name="BFPSchema_1_Path" setvar="bfps_1_bfpschema_1_path"/>
      <output name="BFPSchema_1_SpecVersion" setvar="bfps_1_bfpschema_1_specversion"/>
      <output name="BFPSchema_1_ProductVersion" setvar="bfps_1_bfpschema_1_productversion"/>
      <output name="BFPSchema_1_Vendor" setvar="bfps_1_bfpschema_1_vendor"/>
      <output name="BFPSchema_1_Format_1_FormatOwner" setvar="bfps_1_bfpschema_1_format_1_formatowner"/>
      <output name="BFPSchema_1_Format_1_FormatType" setvar="bfps_1_bfpschema_1_format_1_formattype"/>
      <output name="BFPSchema_1_Format_2_FormatOwner" setvar="bfps_1_bfpschema_1_format_2_formatowner"/>
      <output name="BFPSchema_1_Format_2_FormatType" setvar="bfps_1_bfpschema_1_format_2_formattype"/>
      <output name="BFPSchema_1_Format_3_FormatOwner" setvar="bfps_1_bfpschema_1_format_3_formatowner"/>
      <output name="BFPSchema_1_Format_3_FormatType" setvar="bfps_1_bfpschema_1_format_3_formattype"/>
      <output name="BFPSchema_1_Format_4_FormatOwner" setvar="bfps_1_bfpschema_1_format_4_formatowner"/>
      <output name="BFPSchema_1_Format_4_FormatType" setvar="bfps_1_bfpschema_1_format_4_formattype"/>
      <output name="BFPSchema_1_NumSupportedFormats" setvar="bfps_1_bfpschema_1_numsupportedformats"/>
      <output name="BFPSchema_1_TypeMultiple" setvar="bfps_1_bfpschema_1_typemultiple"/>
      <output name="BFPSchema_1_TypeFacialFeatures" setvar="bfps_1_bfpschema_1_typefacialfeatures"/>
      <output name="BFPSchema_1_TypeVoice" setvar="bfps_1_bfpschema_1_typevoice"/>
      <output name="BFPSchema_1_TypeFingerprint" setvar="bfps_1_bfpschema_1_typefingerprint"/>
      <output name="BFPSchema_1_TypeIris" setvar="bfps_1_bfpschema_1_typeiris"/>
      <output name="BFPSchema_1_TypeRetina" setvar="bfps_1_bfpschema_1_typeretina"/>
      <output name="BFPSchema_1_TypeHandGeometry" setvar="bfps_1_bfpschema_1_typehandgeometry"/>
      <output name="BFPSchema_1_TypeSignatureDynamics" setvar="bfps_1_bfpschema_1_typesignaturedynamics"/>
      <output name="BFPSchema_1_TypeKeystrokeDynamics" setvar="bfps_1_bfpschema_1_typekeystrokedynamics"/>
      <output name="BFPSchema_1_TypeLipMovement" setvar="bfps_1_bfpschema_1_typelipmovement"/>
      <output name="BFPSchema_1_TypeThermalFaceImage" setvar="bfps_1_bfpschema_1_typethermalfaceimage"/>
      <output name="BFPSchema_1_TypeThermalHandImage" setvar="bfps_1_bfpschema_1_typethermalhandimage"/>
      <output name="BFPSchema_1_TypeGait" setvar="bfps_1_bfpschema_1_typegait"/>
      <output name="BFPSchema_1_TypeOther" setvar="bfps_1_bfpschema_1_typeother"/>
      <output name="BFPSchema_1_TypePassword" setvar="bfps_1_bfpschema_1_typepassword"/>
      <output name="BFPSchema_1_BFPPropertyID" setvar="bfps_1_bfpschema_1_bfppropertyid"/>
      <output name="BFPSchema_1_BFPProperty" setvar="bfps_1_bfpschema_1_bfpproperty"/>
      <return setvar="return"/>
    </invoke>

    <!-- Extract the error code from the error value. -->
    <invoke activity="ExtractErrorCode" package="fb6ff5b0-7d5e-11de-8a39-0800200c9a66" break_on_break="true">
      <output name="Returnvalue" setvar="return"/>
    </invoke>

    <!-- Assertion -->
    <assert_condition response_if_false="fail">
      <description>
        The function BioAPI_Capture has returned the expected return value.
      </description>
      <or>
        <!-- If the "_expected_return_value" parameter is 0xFFFFFFFF,
             then check the "return" parameter is not __BioAPI_OK. (error check only) -->
        <and>
          <equal_to var1="_expected_return_value" var2="_indeterminate_error"/>
          <not_equal_to var1="return" var2="__BioAPI_OK"/>
        </and>
        <and>
          <not_equal_to var1="_expected_return_value" var2="_indeterminate_error"/>
          <equal_to var1="return" var2="_expected_return_value"/>
        </and>
      </or>
    </assert_condition>

    <!-- Invoke the activity to terminate the framework under test -->
    <invoke activity="Termination" package="fb6ff5b0-7d5e-11de-8a39-0800200c9a66" break_on_break="true"/>
  </activity>
</package>
```

8.14 Assertion 1.11 – BioAPI_QueryBFPs_And_BioSPI_QueryBFPs

Description: This assertion calls BioAPI_QueryBFPs with the input parameters described in Default Input Table and Test Condition Table, and checks if the framework under test returns BioAPI_OK or an error value in accordance with the description in Expected Result Table. The assertion is according to the following statement in the subclauses from BioAPI 2.0 standard document (ISO/IEC19784-1:2006), which are described in References in this subclause in this part of ISO/IEC 24709.

Excerpts:

Subclause 8.1.11

```
BioAPI_RETURN BioAPI BioAPI_QueryBFPs
       (const BioAPI_UUID               *BSPUuid,
       BioAPI_BFP_LIST_ELEMENT          **BFPList,
       uint32_t                         *NumberOfElements);
```

This function returns a list of BFPs which are currently installed in the component registry and supported by the BSP identified by the given BSP UUID. It performs the following actions (in order):

a) determines which among all currently installed BFPs are supported by the BSP;

b) allocates a memory block large enough to contain an array of elements of type BioAPI_BFP_LIST_ELEMENT with as many elements as the number of BFPs determined in (a);

c) fills the array with identification information (category and UUID) about the BFPs determined in (a); and

d) returns the address of the array in the BFPList parameter and the number of elements of the array in the NumberOfElements parameter.

NOTE When the Framework calls the function BioSPI_QueryBFPs of a BSP, the BSP allocates memory for the data to be returned to the Framework. In some implementation architectures, the Framework will simply pass to the application the data and the addresses exactly as returned by the BSP because the application will interpret the addresses in the same way as the BSP and will be able to access the data that the BSP has placed at those addresses. In other implementation architectures, the framework will need to move all the data returned by the BSP to newly allocated memory blocks accessible to the application, and call BioSPI_Free after copying each memory block but before returning from the BioAPI_QueryBFPs call. In the former case, when the application calls BioAPI_Free, the Framework will make a corresponding call to BioSPI_Free. In the latter case, the calls to BioAPI_Free will be handled internally by the framework. However, such differences in the behavior of the Framework are not visible to the application.

Additional information about the supported BFPs can be retrieved by calling BioAPI_EnumBFPs and analyzing the BFPSchemaArray at the matching BFPUuids.

This function shall only be called after BioAPI_Load has been called for the specified BSP, and shall not be called after BioAPI_Unload has been called for the BSP.

The memory block containing the array shall be freed by the application via a call to BioAPI_Free (see clause 8.7.2) when it is no longer needed by the application.

A BioAPI_RETURN value indicates success or specifies a particular error condition. The value BioAPI_OK indicates success. All other values represent an error condition.

Subclause 9.3.1.6

```
BioAPI_RETURN BioAPI   BioSPI_QueryBFPs
       (const BioAPI_UUID                *BSPUuid,
        BioAPI_BFP_LIST_ELEMENT          **BFPList,
        uint32_t                         *NumberOfElements);
```

References: 8.1.11, 9.3.1.6

Scenario:

a) The testing application does the followings:

1) Initialises the Framework under test, then installs and loads the testing BSP.

2) Calls BioAPI_QueryBFPs with the conditions given by the Default Input Table and by each row of the Test Condition Table.

b) The testing BSP does the followings:

1) Checks the parameters and the contents of the pointer variables as described in the column 'BioSPI function' in the Expected Result Table.

2) Sets the return value of the BioSPI_QueryBFPs given by each row of the Test Condition Table, then returns to the Framework.

c) The testing application does the followings:

1) Checks the return value of the BioSPI_QueryBFPs and if it is not as described in the Expected Result Table, then issues a fail conformity response.

2) Unloads and uninstalls the testing BSP, then terminates the Framework under test.

Parameters: The input parameters described below table are used for this assertion.

Table 43 — Default Input Table for BioAPI_QueryBFPs_And_BioSPI_QueryBFPs

Number	Input parameter name	Input parameter value (underscore:invalid)
1	BSPUuid	*Valid Uuid*
2	no_BSPUuid	false
3	no_BFPList	false
4	no_NumberOfElements	false

NOTE: The parameter names "BSPUuid" and "no_BSPUuid" are not defined in ISO/IEC 24709-1 so the implementation of this parameter depends on each CTS.

Table 44 — Test Condition Table for BioAPI_QueryBFPs_And_BioSPI_QueryBFPs

Test number	Input parameter name	Input parameter value (underscore:invalid)	Supported options in BSP Schema		Return value (from BioSPI)
			OPERATIONS_MASK	OPTIONS_MASK	
011101	BSPUuid	*Valid Uuid*	*QueryBFPs*	-	*OK*
011102	BSPUuid	*Invalid Uuid*	*QueryBFPs*	-	-
011103	no_BSPUuid	true	*QueryBFPs*	-	-
011104	no_BFPList	true	*QueryBFPs*	-	*NVALID_OUTPUT_POINTER*
011105	no_NumberOfElements	true	*QueryBFPs*	-	*INVALID_OUTPUT_ POINTER*
011106	BSPUuid	*Valid Uuid*	-	-	-

NOTE: The parameter names "BSPUuid" and "no_BSPUuid" are not defined in ISO/IEC 24709-1 so the implementation of this parameter depends on each CTS.

Table 45 — Expected Result Table for BioAPI_QueryBFPS_And_BioSPI_QueryBFPs

Test number	BioSPI function (BioAPI/BioSPI parameter check)	BioAPI function			Other conditions
		Return value	Output parameter name	Output parameter value	
011101	X	*OK*	-	-	-
011102	-	*indeterminate error*	-	-	-
011103	-	*INVALID_UUID*	-	-	-
011104	X	*indeterminate error*	-	-	-
011105	X	*indeterminate error*	-	-	-
011106	-	*indeterminate error*	-	-	-

Assertion language package

```
<?xml version="1.0" encoding="UTF-8"?>
<package name="70d92580-792a-11de-8a39-0800200c9a66">
  <author>ISO/IEC JTC1 SC37</author>
  <description>This package contains the assertion "BioAPI_QueryBFPs_And_BioSPI_QueryBFPs".</description>

  <!-- ************** -->
  <!-- Test assertion -->
  <!-- ************** -->
  <assertion name="BioAPI_QueryBFPs_And_BioSPI_QueryBFPs" model="frameworkTesting">
    <description>
      This assertion checks if BioAPI_QueryBFPs is called with input parameters as described in Test Condition Table,
      the framework under test returns BioAPI_OK or error value in accordance with the description in Expected
      Result Table.
    </description>

    <!-- Parameter given by the CTS by referring to the Test Condition Table. -->
    <input name="_testnumber"/>
    <input name="_operationsmask"/>
    <input name="_optionsmask"/>
    <input name="_bspuuid"/>
    <input name="_no_bspuuid"/>
    <input name="_no_bfplist"/>
    <input name="_no_numberofelements"/>

    <!-- Parameter given by the CTS by referring to the Expected Result Table. -->
    <input name="_expected_return_value"/>

    <!-- Invocation of the primary activity of this assertion. -->
    <invoke activity="BioAPI_QueryBFPs_And_BioSPI_QueryBFPs"/>

    <!-- Bind activities to check the BioSPI function invoked by the framework under test -->
    <bind activity="SPI_QueryBFPs" function="BioSPI_QueryBFPs"/>
  </assertion>

  <!-- **************** -->
  <!-- Primary activity -->
  <!-- **************** -->
  <activity name="BioAPI_QueryBFPs_And_BioSPI_QueryBFPs">

    <!-- Invoke the activity BSPLoad the framework under test -->
    <invoke activity="BSPLoad" package="fb6ff5b0-7d5e-11de-8a39-0800200c9a66" break_on_break="true">
      <input name="Operationsmask" var="_operationsmask"/>
      <input name="Optionsmask" var="_optionsmask"/>
      <input name="Bspuuid" var="_bspuuid"/>
    </invoke>

    <!-- Invoke the function BioAPI_QueryBFPs. -->
    <invoke function="BioAPI_QueryBFPs">
      <input name="BSPUuid" var="_bspuuid"/>
      <input name="no_BSPUuid" var="_no_bspuuid"/>
      <input name="no_BFPList" var="_no_bfplist"/>
      <input name="no_NumberOfElements" var="_no_numberofelements"/>
      <output name="BFP_1_BFPCategory" setvar="bfp_1_bfpcategory"/>
      <output name="BFP_1_BFPUuid" setvar="bfp_1_bfpuuid"/>
      <output name="NumberOfElements" setvar="numberofelements"/>
     <return setvar="return"/>
    </invoke>

    <!-- Extract the error code from the error value. -->
    <invoke activity="ExtractErrorCode" package="fb6ff5b0-7d5e-11de-8a39-0800200c9a66" break_on_break="true">
      <output name="Returnvalue" setvar="return"/>
    </invoke>

    <!-- Assertion -->
    <assert_condition response_if_false="fail">
      <description>
        The function BioAPI_QueryBFPs has returned the expected return value.
      </description>
      <or>
        <!-- If the "_expected_return_value" parameter is 0xFFFFFFFF,
             then check the "return" parameter is not __BioAPI_OK. (error check only) -->
        <and>
          <equal_to var1="_expected_return_value" var2="_indeterminate_error"/>
          <not_equal_to var1="return" var2="__BioAPI_OK"/>
        </and>
        <and>
          <not_equal_to var1="_expected_return_value" var2="_indeterminate_error"/>
          <equal_to var1="return" var2="_expected_return_value"/>
        </and>
      </or>
    </assert_condition>

    <!-- Invoke the activity BSPUnload the framework under test -->
    <invoke activity="BSPUnload" package="fb6ff5b0-7d5e-11de-8a39-0800200c9a66" break_on_break="true">
      <input name="Bspuuid" var="_bspuuid"/>
    </invoke>
  </activity>

  <!-- ********************************** -->
  <!-- Activity bound to BioSPI_QueryBFPs -->
  <!-- ********************************** -->
  <activity name="SPI_QueryBFPs">
    <input name="BSPUuid"/>
    <input name="no_BSPUuid"/>
    <input name="no_BFPList"/>
    <input name="no_NumberOfElements"/>
    <output name="BFP_1_BFPCategory"/>
    <output name="BFP_1_BFPUuid"/>
    <output name="NumberOfElements"/>
    <output name="return"/>

    <!-- check API=SPI -->
    <assert_condition response_if_false="fail">
      <equal_to var1="BSPUuid" var2="_bspuuid"/>
      <equal_to var1="no_BSPUuid" var2="_no_bspuuid"/>
      <equal_to var1="no_BFPList" var2="_no_bfplist"/>
      <equal_to var1="no_NumberOfElements" var2="_no_numberofelements"/>
    </assert_condition>
  </activity>
</package>
```

8.15 Assertion 1.12 – BioAPI_ControlUnit_And_BioSPI_ControlUnit

Description: This assertion calls BioAPI_ControlUnit with the input parameters described in Default Input Table and Test Condition Table, and checks if the framework under test returns BioAPI_OK or an error value in accordance with the description in Expected Result Table. The assertion is according to the following statement in the subclauses from BioAPI 2.0 standard document (ISO/IEC19784-1:2006), which are described in References in this subclause in this part of ISO/IEC 24709.

Excerpts:

Subclause 8.1.12

```
BioAPI_RETURN BioAPI BioAPI_ControlUnit
        (BioAPI_HANDLE         BSPHandle,
        BioAPI_UNIT_ID         UnitID,
        uint32_t               ControlCode,
        const BioAPI_DATA      *InputData,
        BioAPI_DATA            *OutputData);
```

This function sends control data from the application to the BioAPI Unit and receives status or operation data from that unit. The content of the ControlCode, the send (input) data, and the receive (output) data will be specified in the related interface specification for this BioAPI Unit (or associated FPI, if present).

The function allocates a memory block large enough to contain the output data that are to be returned to the application, fills the memory block with the data, and sets the fields Length and Data of the OutputData structure to the size and address of the memory block (respectively).

The memory block returned by the BioAPI function call shall be freed by the application using BioAPI_Free (see clause 8.7.2 in ISO/IEC 19784-1:2006).

A BioAPI_RETURN value indicates success or specifies a particular error condition. The value BioAPI_OK indicates success. All other values represent an error condition.

Subclause 9.3.1.7

```
BioAPI_RETURN BioAPI BioSPI_ControlUnit
      (BioAPI_HANDLE           BSPHandle,
      BioAPI_UNIT_ID           UnitID,
      uint32_t                 ControlCode,
      const BioAPI_DATA        *InputData,
      BioAPI_DATA              *OutputData);
```

References: 8.1.12, 9.3.1.7

Scenario:

a) The testing application does the followings:

1) Initialises the Framework under test, then installs, loads and attaches the testing BSP.

2) Calls BioAPI_ControlUnit with the conditions given by the Default Input Table and by each row of the Test Condition Table.

b) The testing BSP does the followings:

1) Checks the parameters and the contents of the pointer variables as described in the column 'BioSPI function' in the Expected Result Table.

2) Sets the return value of the BioSPI_ControlUnit given by each row of the Test Condition Table, then returns to the Framework.

c) The testing application does the followings:

1) Checks the return value of the BioAPI_ControlUnit and if it is not as described in the Expected Result Table, then issues a fail conformity response.

2) Detaches, unloads and uninstalls the testing BSP, then terminates the Framework under test.

Parameters: The input parameters described below table are used for this assertion.

Table 46 — Default Input Table for BioAPI_ControlUnit_And_BioSPI_ControlUnit

Number	Input parameter name	Input parameter value (underscore:invalid)
1	BSPHandle	*Valid BSPHandle*
2	UnitID	N/A
3	ControlCode	N/A
4	InputData	N/A
5	no_OutputData	false

Table 47 — Test Condition Table for BioAPI_ControlUnit_And_BioSPI_ControlUnit

Test number	Input parameter name	Input parameter value (underscore:invalid)	Supported options in BSP Schema		Return value (from BioSPI)
			OPERATIONS_MASK	OPTIONS_MASK	
011201	BSPHandle	*Valid BSPHandle*	*ControlUnits*	-	*OK*
011202	BSPHandle	*Invalid BSPHandle*	*ControlUnits*	-	-
011203	no_OutputData	true	*ControlUnits*	-	*INVALID_OUTPUT_POINTER*
011204	BSPHandle	*Valid BSPHandle*	-	-	-

Table 48 — Expected Result Table for BioAPI_ControlUnit_And_BioSPI_ControlUnit

Test number	BioSPI function (BioAPI/BioSPI parameter check)	BioAPI function			Other conditions
		Return value	Output parameter name	Output parameter value	
011201	X	*OK*	-	-	-
011202	-	*INVALID_BSP_HANDLE*	-	-	-
011203	X	*indeterminate error*	-	-	-
011204	-	*indeterminate error*	-	-	-

Assertion language package

```
<?xml version="1.0" encoding="UTF-8"?>
<package name="819d98b0-792a-11de-8a39-0800200c9a66">
  <author>ISO/IEC JTC1 SC37</author>
  <description>This package contains the assertion "BioAPI_ControlUnit_And_BioSPI_ControlUnit".</description>

  <!-- ************** -->
  <!-- Test assertion -->
  <!-- ************** -->
  <assertion name="BioAPI_ControlUnit_And_BioSPI_ControlUnit" model="frameworkTesting">
    <description>
      This assertion checks if BioAPI_ControlUnit is called with input parameters as described in
      Test Condition Table, the framework under test returns BioAPI_OK or error value in accordance with
      the description in Expected Result Table.
    </description>

    <!-- Parameter given by the CTS by referring to the Test Condition Table. -->
    <input name="_testnumber"/>
    <input name="_operationsmask"/>
    <input name="_optionsmask"/>
    <input name="_unitid"/>
    <input name="_controlcode"/>
    <input name="_inputdata"/>
    <input name="_no_outputdata"/>

    <!-- Parameter given by the CTS by referring to the Expected Result Table. -->
    <input name="_expected_return_value"/>

    <!-- Parameter given by the CTS. -->
    <input name="_bspuuid"/>

    <!-- Invocation of the primary activity of this assertion. -->
    <invoke activity="BioAPI_ControlUnit_And_BioSPI_ControlUnit"/>

    <!-- Bind activities to check the BioSPI function invoked by the framework under test -->
    <bind activity="SPI_ControlUnit" function="BioSPI_ControlUnit"/>
 </assertion>

  <!-- **************** -->
  <!-- Primary activity -->
  <!-- **************** -->
  <activity name="BioAPI_ControlUnit_And_BioSPI_ControlUnit">

    <!-- Invoke the activity BSP Attach the framework under test -->
    <invoke activity="BSPAttach" package="fb6ff5b0-7d5e-11de-8a39-0800200c9a66" break_on_break="true">
      <input name="Operationsmask" var="_operationsmask"/>
      <input name="Optionsmask" var="_optionsmask"/>
      <input name="Bspuuid" var="_bspuuid"/>
      <output name="Newbsphandle" setvar="_bsphandle"/>
    </invoke>

    <!-- If Test Number is 011202, make _bsphandle to an illegal value. -->
    <add name="_bsphandle" value="1">
      <only_if><equal_to var1="_testnumber" value2="011202"/></only_if>
    </add>

    <!-- Invoke the function BioAPI_ControlUnit. -->
    <invoke function="BioAPI_ControlUnit">
      <input name="BSPHandle" var="_bsphandle"/>
      <input name="UnitID" var="_unitid"/>
      <input name="ControlCode" var="_controlcode"/>
      <input name="InputData" var="_inputdata"/>
      <input name="no_Outputdata" var="_no_outputdata"/>
      <output name="OutputData" setvar="outputdata"/>
      <return setvar="return"/>
    </invoke>

    <!-- Extract the error code from the error value. -->
    <invoke activity="ExtractErrorCode" package="fb6ff5b0-7d5e-11de-8a39-0800200c9a66" break_on_break="true">
      <output name="Returnvalue" setvar="return"/>
    </invoke>

    <!-- Assertion -->
    <assert_condition response_if_false="fail">
      <description>
        The function BioAPI_ControlUnit has returned the expected return value.
      </description>
      <or>
        <!-- If the "_expected_return_value" parameter is 0xFFFFFFFF,
             then check the "return" parameter is not __BioAPI_OK. (error check only) -->
        <and>
          <equal_to var1="_expected_return_value" var2="_indeterminate_error"/>
          <not_equal_to var1="return" var2="__BioAPI_OK"/>
        </and>
        <and>
          <not_equal_to var1="_expected_return_value" var2="_indeterminate_error"/>
          <equal_to var1="return" var2="_expected_return_value"/>
        </and>
      </or>
    </assert_condition>

    <!-- If Test Number is 011202, make _bsphandle to an valid value. -->
    <subtract name="_bsphandle" value="1">
      <only_if><equal_to var1="_testnumber" value2="011202"/></only_if>
    </subtract>

    <!-- Invoke the functions Detach -->
    <invoke activity="BSPDetach" package="fb6ff5b0-7d5e-11de-8a39-0800200c9a66" break_on_break="true">
      <input name="Bsphandle" var="_bsphandle"/>
      <input name="Bspuuid" var="_bspuuid"/>
    </invoke>
  </activity>

  <!-- ************************************ -->
  <!-- Activity bound to BioSPI_ControlUnit -->
  <!-- ************************************ -->
  <activity name="SPI_ControlUnit">
    <input name="BSPHandle"/>
    <input name="UnitID"/>
    <input name="ControlCode"/>
    <input name="InputData"/>
    <input name="no_Outputdata"/>
    <output name="OutputData"/>
    <output name="return"/>

    <!-- check API=SPI -->
    <assert_condition response_if_false="fail">
      <and>
        <equal_to var1="BSPHandle" var2="_bsphandle"/>
        <equal_to var1="UnitID" var2="_unitid"/>
        <equal_to var1="ControlCode" var2="_controlcode"/>
        <equal_to var1="InputData" var2="_controlcode"/>
        <equal_to var1="no_Outputdata" var2="_no_outputdata"/>
      </and>
    </assert_condition>
  </activity>
</package>
```

8.16 Assertion 2.1 – BioAPI_FreeBIRHandle_And_BioSPI_FreeBIRHandle

Description: This assertion calls BioAPI_FreeBIRHandle with the input parameters described in Default Input Table and Test Condition Table, and checks if the framework under test returns BioAPI_OK or an error value in accordance with the description in Expected Result Table. The assertion is according to the following statement in the subclauses from BioAPI 2.0 standard document (ISO/IEC19784-1:2006), which are described in References in this subclause in this part of ISO/IEC 24709.

Excerpts:

Subclause 8.2.1

```
BioAPI_RETURN BioAPI BioAPI_FreeBIRHandle
        (BioAPI_HANDLE           BSPHandle,
         BioAPI_BIR_HANDLE    Handle);
```

Subclause 9.3.2.1

```
BioAPI_RETURN BioAPI BioSPI_FreeBIRHandle
        (BioAPI_HANDLE           BSPHandle,
         BioAPI_BIR_HANDLE    Handle);
```

This function frees the memory and resources associated with the specified BIR Handle. The associated BIR is no longer referenceable through that handle. If necessary, the biometric application can store the BIR into a BSP-controlled BIR database (using BioAPI_DbStoreBIR) before calling BioAPI_FreeBIRHandle. Alternatively, the application can call BioAPI_GetBIRFromHandle (which will retrieve the BIR and free the handle) instead of calling BioAPI_FreeBIRHandle.

This function shall only be called after BioAPI_BSPAttach has been called, and shall not be called after BioAPI_BSPDetach has been called for the BSP handle created by the call to BioAPI_BSPAttach.

A BioAPI_RETURN value indicates success or specifies a particular error condition. The value BioAPI_OK indicates success. All other values represent an error condition.

References: 8.2.1, 9.3.2.1

Scenario:

a) The testing application does the followings:
   1) Initialises the Framework under test, then installs, loads and attaches the testing BSP.
   2) Captures a biometric sample to obtain a BIR handle.
   3) Calls BioAPI_FreeBIRHandle with the conditions given by the Default Input Table and by each row of the Test Condition Table.

b) The testing BSP does the followings:
   1) Checks the parameters and the contents of the pointer variables as described in the column 'BioSPI function' in the Expected Result Table.
   2) Sets the return value of the BioSPI_BSPUnload given by each row of the Test Condition Table, then returns to the Framework.

c) The testing application does the followings:
   1) Checks the return value of the BioAPI_FreeBIRHandle and if it is not as described in the Expected Result Table, then issues a fail conformity response.
   2) Detaches, unloads and uninstalls the testing BSP, then terminates the Framework under test.

Parameters: The input parameters described below table are used for this assertion.

Table 49 — Default Input Table for BioAPI_FreeBIRHandle_And_BioSPI_FreeBIRHandle

Number	Input parameter name	Input parameter value (underscore:invalid)
1	BSPHandle	*Valid BSPHandle*
2	Handle	*Valid BirHandle*

Table 50 — Test Condition Table for BioAPI_FreeBIRHandle_And_BioSPI_FreeBIRHandle

Test number	Input parameter name	Input parameter value (underscore:invalid)	Supported options in BSP Schema		Return value (from BioSPI)
			OPERATIONS_MASK	OPTIONS_MASK	
020101	BSPHandle	*Valid BSPHandle*	*Capture*	-	*OK*
020102	BSPHandle	*Invalid BSPHandle*	*Capture*	-	-
020103	Handle	*Invalid BIRHandle*	*Capture*	-	*INVALID_BIR_HANDLE*

Table 51 — Expected Result Table for BioAPI_FreeBIRHandle_And_BioSPI_FreeBIRHandle

Test number	BioSPI function (BioAPI/BioSPI parameter check)	BioAPI function			Other conditions
		Return value	Output parameter name	Output parameter value	
020101	X	*OK*	-	-	-
020102	-	*INVALID_BSP_HANDLE*	-	-	-
020103	X	*INVALID_BIR_HANDLE*	-	-	-

Assertion language package

```
<?xml version="1.0" encoding="UTF-8"?>
<package name="94a32240-792a-11de-8a39-0800200c9a66">
  <author>ISO/IEC JTC1 SC37</author>
  <description>This package contains the assertion "BioAPI_FreeBIRHandle_And_BioSPI_FreeBIRHandle".</description>

  <!-- ************** -->
  <!-- Test assertion -->
  <!-- ************** -->
  <assertion name="BioAPI_FreeBIRHandle_And_BioSPI_FreeBIRHandle" model="frameworkTesting">
    <description>
      This assertion checks if BioAPI_FreeBIRHandle is called with input parameters as described in Test Condition
      Table, the framework under test returns BioAPI_OK or error value in accordance with the description in
      Expected Result Table.
    </description>

    <!-- Parameter given by the CTS by referring to the Test Condition Table. -->
    <input name="_testnumber"/>
    <input name="_operation"/>
    <input name="_option"/>

    <!-- Parameter given by the CTS by referring to the Expected Result Table. -->
    <input name="_expected_return_value"/>

    <!-- Parameter given by the CTS. -->
    <input name="_bspuuid"/>

    <!-- Invocation of the primary activity of this assertion. -->
    <invoke activity="BioAPI_FreeBIRHandle_And_BioSPI_FreeBIRHandle"/>

    <!-- Bind activities to check the BioSPI function invoked by the framework under test -->
    <bind activity="SPI_FreeBIRHandle" function="BioSPI_FreeBIRHandle"/>
  </assertion>

  <!-- **************** -->
  <!-- Primary activity -->
  <!-- **************** -->
  <activity name="BioAPI_FreeBIRHandle_And_BioSPI_FreeBIRHandle">

    <!-- Invoke the activity BSP Attach the framework under test -->
    <invoke activity="BSPAttach" package="fb6ff5b0-7d5e-11de-8a39-0800200c9a66" break_on_break="true">
      <input name="Operationsmask" var="_operation"/>
      <input name="Optionsmask" var="_option"/>
      <input name="Bspuuid" var="_bspuuid"/>
      <output name="Newbsphandle" setvar="_bsphandle"/>
    </invoke>

    <!-- Invoke the activity Capture the framework under test -->
    <invoke activity="Capture" package="fb6ff5b0-7d5e-11de-8a39-0800200c9a66" break_on_break="true">
      <input name="Bsphandle" var="_bsphandle"/>
      <output name="Capturedbir" setvar="_handle"/>
    </invoke>
    <!-- Set an illegal value to the parameter. -->
    <add name="_bsphandle" value="1">
      <only_if><equal_to var1="_testnumber" value2="020102"/></only_if>
    </add>
    <add name="_handle" value="1">
      <only_if><equal_to var1="_testnumber" value2="020103"/></only_if>
    </add>

    <!-- Invoke the function BioAPI_FreeBIRHandle. -->
    <invoke function="BioAPI_FreeBIRHandle">
      <input name="BSPHandle" var="_bsphandle"/>
      <input name="Handle" var="_handle"/>
      <return setvar="return"/>
    </invoke>

    <!-- Extract the error code from the error value. -->
    <invoke activity="ExtractErrorCode" package="fb6ff5b0-7d5e-11de-8a39-0800200c9a66" break_on_break="true">
      <output name="Returnvalue" setvar="return"/>
    </invoke>

    <!-- Assertion -->
    <assert_condition response_if_false="fail">
      <description>
        The function BioAPI_FreeBIRHandle has returned the expected return value.
      </description>
      <or>
        <!-- If the "_expected_return_value" parameter is 0xFFFFFFFF,
             then check the "return" parameter is not __BioAPI_OK. (error check only) -->
        <and>
          <equal_to var1="_expected_return_value" var2="_indeterminate_error"/>
          <not_equal_to var1="return" var2="__BioAPI_OK"/>
        </and>
        <and>
          <not_equal_to var1="_expected_return_value" var2="_indeterminate_error"/>
          <equal_to var1="return" var2="_expected_return_value"/>
        </and>
      </or>
    </assert_condition>

    <!-- If Test Number is 020102, make _bsphandle to an valid value. -->
    <subtract name="_bsphandle" value="1">
      <only_if><equal_to var1="_testnumber" value2="020102"/></only_if>
    </subtract>

    <!-- Invoke the functions Detach -->
    <invoke activity="BSPDetach" package="fb6ff5b0-7d5e-11de-8a39-0800200c9a66" break_on_break="true">
      <input name="Bsphandle" var="_bsphandle"/>
      <input name="Bspuuid" var="_bspuuid"/>
    </invoke>
  </activity>

  <!-- ************************************** -->
  <!-- Activity bound to BioSPI_FreeBIRHandle -->
  <!-- ************************************** -->
  <activity name="SPI_FreeBIRHandle">
    <input name="BSPHandle"/>
    <input name="Handle"/>
    <output name="return"/>

    <!-- check API=SPI -->
    <assert_condition response_if_false="fail">
      <and>
        <equal_to var1="BSPHandle" var2="_bsphandle"/>
        <equal_to var1="Handle" var2="_handle"/>
      </and>
    </assert_condition>
  </activity>
</package>
```

8.17 Assertion 2.2 – BioAPI_GetBIRFromHandle_And_BioSPI_GetBIRFromHandle

Description: This assertion calls BioAPI_GetBIRFromHandle with the input parameters described in Default Input Table and Test Condition Table, and checks if the framework under test returns BioAPI_OK or an error value in accordance with the description in Expected Result Table. The assertion is according to the following statement in the subclauses from BioAPI 2.0 standard document (ISO/IEC19784-1:2006), which are described in References in this subclause in this part of ISO/IEC 24709.

Excerpts:

Subclause 8.2.2

```
BioAPI_RETURN BioAPI BioAPI_GetBIRFromHandle
          (BioAPI_HANDLE         BSPHandle,
          BioAPI_BIR_HANDLE   Handle,
          BioAPI_BIR                *BIR);
```

This function returns the BIR associated with a BIR handle returned by a BSP. The BIR handle is freed by the BSP before the function returns.

A BioAPI_RETURN value indicates success or specifies a particular error condition. The value BioAPI_OK indicates success. All other values represent an error condition.

Subclause 9.3.2.2

```
BioAPI_RETURN BioAPI BioSPI_GetBIRFromHandle
          (BioAPI_HANDLE         BSPHandle,
          BioAPI_BIR_HANDLE   Handle,
          BioAPI_BIR                *BIR);
```

References: 8.2.2, 9.3.2.2

Scenario:

a) The testing application does the followings:

1) Initialises the Framework under test, then installs, loads and attaches the testing BSP.
2) Enrolls to obtain a BIR handle.
3) Calls BioAPI_GetBIRFromHandle with the conditions given by the Default Input Table and by each row of the Test Condition Table.

b) The testing BSP does the followings:

1) Checks the parameters and the contents of the pointer variables as described in the column 'BioSPI function' in the Expected Result Table.
2) Sets the return value of the BioSPI_GetBIRFromHandle given by each row of the Test Condition Table, then returns to the Framework.

c) The testing application does the followings:

1) Checks the return value of the BioAPI_GetBIRFromHandle and if it is not as described in the Expected Result Table, then issues a fail conformity response.
2) Detaches, unloads and uninstalls the testing BSP, then terminates the Framework under test.

Parameters: The input parameters described below table are used for this assertion.

Table 52 — Default Input Table for BioAPI_GetBIRFromHandle_And_BioSPI_GetBIRFromHandle

Number	Input parameter name	Input parameter value (underscore:invalid)
1	BSPHandle	*Valid BSPHandle*
2	Handle	*Valid BirHandle*
3	no_BIR	false

Table 53 — Test Condition Table for BioAPI_GetBIRFromHandle_BioSPI_GetBIRFromHandle

Test number	Input parameter name	Input parameter value (underscore:invalid)	Supported options in BSP Schema		Return value (from BioSPI)
			OPERATIONS_MASK	OPTIONS_MASK	
020201	BSPHandle	*Valid BSPHandle*	*Enroll*	-	*OK*
020202	BSPHandle	*Invalid BSPHandle*	*Enroll*	-	-
020203	Handle	*Invalid BIRHandle*	*Enroll*	-	*INVALID_BIR_HANDLE*
020204	no_BIR	true	*Enroll*	-	*INVALID_OUTPUT_POINTER*

Table 54 — Expected Result Table for BioAPI_GetBIRFromHandle_And_BioSPI_GetBIRFromHandle

Test number	BioSPI function (BioAPI/BioSPI parameter check)	BioAPI function			Other conditions
		Return value	Output parameter name	Output parameter value	
020201	X	*OK*	-	-	-
020202	-	*INVALID_BSP_HANDLE*	-	-	-
020203	X	*INVALID_BIR_HANDLE*	-	-	-
020204	X	*indeterminate error*	-	-	-

Assertion language package

```
<?xml version="1.0" encoding="UTF-8"?>
<package name="ca10cea0-792a-11de-8a39-0800200c9a66">
  <author>ISO/IEC JTC1 SC37</author>
  <description>This package contains the assertion "BioAPI_Enroll_And_BioSPI_Enroll".</description>

  <!-- ************** -->
  <!-- Test assertion -->
  <!-- ************** -->
  <assertion name="BioAPI_GetBIRFromHandle_And_BioSPI_GetBIRFromHandle" model="frameworkTesting">
    <description>
      This assertion checks if BioAPI_GetBIRFromHandle is called with input parameters as described in Test
      Condition Table, the framework under test returns BioAPI_OK or error value in accordance with the description
      in Expected Result Table.
    </description>

    <!-- Parameter given by the CTS by referring to the Test Condition Table. -->
    <input name="_testnumber"/>
    <input name="_operationsmask"/>
    <input name="_optionsmask"/>
    <input name="_no_bir"/>

    <!-- Parameter given by the CTS by referring to the Expected Result Table. -->
    <input name="_expected_return_value"/>

    <!-- Parameter given by the CTS. -->
    <input name="_bspuuid"/>
    <input name="_outputformatowner"/>
    <input name="_outputformattype"/>

    <!-- Invocation of the primary activity of this assertion. -->
    <invoke activity="BioAPI_GetBIRFromHandle_And_BioSPI_GetBIRFromHandle"/>

    <!-- Bind activities to check the BioSPI function invoked by the framework under test -->
    <bind activity="SPI_GetBIRFromHandle" function="BioSPI_GetBIRFromHandle"/>
  </assertion>

  <!-- **************** -->
  <!-- Primary activity -->
  <!-- **************** -->
  <activity name="BioAPI_GetBIRFromHandle_And_BioSPI_GetBIRFromHandle">

    <!-- Invoke the activity BSP Attach the framework under test -->
    <invoke activity="BSPAttach" package="fb6ff5b0-7d5e-11de-8a39-0800200c9a66" break_on_break="true">
      <input name="Operationsmask" var="_operationsmask"/>
      <input name="Optionsmask" var="_optionsmask"/>
      <input name="Bspuuid" var="_bspuuid"/>
      <output name="Newbsphandle" setvar="_bsphandle"/>
    </invoke>

    <!-- Enrolls to obtain a BIR handle -->
    <invoke activity="Enroll" package="fb6ff5b0-7d5e-11de-8a39-0800200c9a66" break_on_break="true">
      <input name="Bsphandle" var="_bsphandle"/>
      <input name="Purpose" value="__BioAPI_PURPOSE_ENROLL"/>
      <input name="Outputformatowner" value="_outputformatowner"/>
      <input name="Outputformattype" value="_outputformattype"/>
      <output name="Newtemplate" setvar="_birhandle"/>
    </invoke>

    <!-- Set an illegal value to the parameter. -->
    <add name="_bsphandle" value="1">
      <only_if><equal_to var1="_testnumber" value2="020202"/></only_if>
    </add>
    <add name="_birhandle" value="1">
      <only_if><equal_to var1="_testnumber" value2="020203"/></only_if>
    </add>

    <!-- Invoke the function BioAPI_GetBIRFromHandle. -->
    <invoke function="BioAPI_GetBIRFromHandle">
      <input name="BSPHandle" var="_bsphandle"/>
      <input name="Handle" var="_birhandle"/>
      <input name="no_BIR" var="_no_bir"/>
      <return setvar="return"/>
    </invoke>

    <!-- Extract the error code from the error value. -->
    <invoke activity="ExtractErrorCode" package="fb6ff5b0-7d5e-11de-8a39-0800200c9a66" break_on_break="true">
      <output name="Returnvalue" setvar="return"/>
    </invoke>

    <!-- Assertion -->
    <assert_condition response_if_false="fail">
      <description>
        The function BioAPI_GetBIRFromHandle has returned the expected return value.
      </description>
      <or>
        <!-- If the "_expected_return_value" parameter is 0xFFFFFFFF,
             then check the "return" parameter is not __BioAPI_OK. (error check only) -->
        <and>
          <equal_to var1="_expected_return_value" var2="_indeterminate_error"/>
          <not_equal_to var1="return" var2="__BioAPI_OK"/>
        </and>
        <and>
          <not_equal_to var1="_expected_return_value" var2="_indeterminate_error"/>
          <equal_to var1="return" var2="_expected_return_value"/>
        </and>
      </or>
    </assert_condition>

    <!-- If Test Number is 020202, make _bsphandle to an valid value. -->
    <subtract name="_bsphandle" value="1">
      <only_if><equal_to var1="_testnumber" value2="020202"/></only_if>
    </subtract>

    <!-- Invoke the functions Detach -->
    <invoke activity="BSPDetach" package="fb6ff5b0-7d5e-11de-8a39-0800200c9a66" break_on_break="true">
      <input name="Bsphandle" var="_bsphandle"/>
      <input name="Bspuuid" var="_bspuuid"/>
    </invoke>
  </activity>

  <!-- ***************************************** -->
  <!-- Activity bound to BioSPI_GetBIRFromHandle -->
  <!-- ***************************************** -->
  <activity name="SPI_GetBIRFromHandle">
    <input name="BSPHandle"/>
    <input name="Handle"/>
    <input name="no_BIR"/>
    <output name="return"/>

    <!-- check API=SPI -->
    <assert_condition response_if_false="fail">
      <and>
        <equal_to var1="BSPHandle" var2="_bsphandle"/>
        <equal_to var1="Handle" var2="_bsphandle"/>
        <equal_to var1="no_BIR" var2="_no_bir"/>
      </and>
    </assert_condition>
  </activity>
</package>
```

8.18 Assertion 2.3 – BioAPI_GetHeaderFromHandle_And_BioSPI_GetHeaderFromHandle

Description: This assertion calls BioAPI_GetHeaderFromHandle with the input parameters described in Default Input Table and Test Condition Table, and checks if the framework under test returns BioAPI_OK or an error value in accordance with the description in Expected Result Table. The assertion is according to the following statement in the subclauses from BioAPI 2.0 standard document (ISO/IEC19784-1:2006), which are described in References in this subclause in this part of ISO/IEC 24709.

Excerpts:

Subclause 8.2.3

```
BioAPI_RETURN BioAPI BioAPI_GetHeaderFromHandle
        (BioAPI_HANDLE              BSPHandle,
        BioAPI_BIR_HANDLE           Handle,
        BioAPI_BIR_HEADER           *Header);
```

This function retrieves the BIR header of the BIR identified by Handle. The BIR handle is not freed by the BSP.

A BioAPI_RETURN value indicates success or specifies a particular error condition. The value BioAPI_OK indicates success. All other values represent an error condition.

Subclause 9.3.2.3

```
BioAPI_RETURN BioAPI BioSPI_GetHeaderFromHandle
        (BioAPI_HANDLE              BSPHandle,
        BioAPI_BIR_HANDLE           Handle,
        BioAPI_BIR_HEADER           *Header);
```

References: 8.2.3, 9.3.2.3

Scenario:

a) The testing application does the followings:
   1) Initialises the Framework under test, then installs, loads and attaches the testing BSP.
   2) Enrolls to obtain a BIR handle.
   3) Calls BioAPI_GetHeaderFromHandle with the conditions given by the Default Input Table and by each row of the Test Condition Table.

b) The testing BSP does the followings:
   1) Checks the parameters and the contents of the pointer variables as described in the column 'BioSPI function' in the Expected Result Table.
   2) Sets the return value of the BioSPI_GetHeaderFromHandle given by each row of the Test Condition Table, then returns to the Framework.

c) The testing application does the followings:
   1) Checks the return value of the BioAPI_GetHeaderFromHandle and if it is not as described in the Expected Result Table, then issues a fail conformity response.
   2) Detaches, unloads and uninstalls the testing BSP, then terminates the Framework under test.

Parameters: The input parameters described below table are used for this assertion.

Table 55 — Default Input Table for BioAPI_GetHeaderFromHandle_And_BioSPI_GetHeaderFromHandle

Number	Input parameter name	Input parameter value (underscore:invalid)
1	BSPHandle	*Valid BSPHandle*
2	Handle	*Valid BIRHandle*
3	no_Header	false

Table 56 — Test Condition Table for BioAPI_GetHeaderFromHandle_And_BioSPI_GetHeaderFromHanlde

Test number	Input parameter name	Input parameter value (underscore:invalid)	Supported options in BSP Schema		Return value (from BioSPI)
			OPERATIONS_MASK	OPTIONS_MASK	
020301	BSPHandle	*Valid BSPHandle*	*Capture*	-	*OK*
020302	BSPHandle	*Invalid BSPHandle*	*Capture*	-	-
020303	Handle	*Invalid BIRHandle*	*Capture*	-	*INVALID_BIR_HANDLE*
020304	no_Header	true	*Capture*	-	*INVALID_OUTPUT_POINTER*

Table 57 — Expected Result Table for BioAPI_GetHeaderFromHandle_And_BioSPI_GetHeaderFromHandle

Test number	BioSPI function (BioAPI/BioSPI parameter check)	BioAPI function			Other conditions
		Return value	Output parameter name	Output parameter value	
020301	X	*OK*	-	-	-
020302	-	*INVALID_BSP_HANDLE*	-	-	-
020303	X	*INVALID_BIR_HANDLE*	-	-	-
020304	X	*indeterminate error*	-	-	-

Assertion language package

```
<?xml version="1.0" encoding="UTF-8"?>
<package name="d9332a90-792a-11de-8a39-0800200c9a66">
  <author>ISO/IEC JTC1 SC37</author>
  <description>
   This package contains the assertion "BioAPI_GetHeaderFromHandle_And_BioSPI_GetHeaderFromHandle".
  </description>

  <!-- ************** -->
  <!-- Test assertion -->
  <!-- ************** -->
  <assertion name="BioAPI_GetHeaderFromHandle_And_BioSPI_GetHeaderFromHandle" model="frameworkTesting">
    <description>
      This assertion checks if BioAPI_GetHeaderFromHandle is called with input parameters as described in
      Test Condition Table, the framework under test returns BioAPI_OK or error value in accordance with
      the description in Expected Result Table.
    </description>

    <!-- Parameter given by the CTS by referring to the Test Condition Table. -->
    <input name="_testnumber"/>
    <input name="_operationsmask"/>
    <input name="_optionsmask"/>
    <input name="_no_header"/>

    <!-- Parameter given by the CTS by referring to the Expected Result Table. -->
    <input name="_expected_return_value"/>

    <!-- Parameter given by the CTS. -->
    <input name="_bspuuid"/>
    <input name="_outputformatowner"/>
    <input name="_outputformattype"/>

    <!-- Invocation of the primary activity of this assertion. -->
    <invoke activity="BioAPI_GetHeaderFromHandle_And_BioSPI_GetHeaderFromHandle"/>

    <!-- Bind activities to check the BioSPI function invoked by the framework under test -->
    <bind activity="SPI_GetHeaderFromHandle" function="BioSPI_GetHeaderFromHandle"/>
  </assertion>

  <!-- **************** -->
  <!-- Primary activity -->
  <!-- **************** -->
  <activity name="BioAPI_GetHeaderFromHandle_And_BioSPI_GetHeaderFromHandle">

    <!-- Invoke the activity BSP Attach the framework under test -->
    <invoke activity="BSPAttach" package="fb6ff5b0-7d5e-11de-8a39-0800200c9a66" break_on_break="true">
      <input name="Operationsmask" var="_operationsmask"/>
      <input name="Optionsmask" var="_optionsmask"/>
      <input name="Bspuuid" var="_bspuuid"/>
      <output name="Newbsphandle" setvar="_bsphandle"/>
    </invoke>

    <!-- Enrolls to obtain a BIR handle -->
    <invoke activity="Enroll" package="fb6ff5b0-7d5e-11de-8a39-0800200c9a66" break_on_break="true">
      <input name="Bsphandle" var="_bsphandle"/>
      <input name="Purpose" value="__BioAPI_PURPOSE_ENROLL"/>
      <input name="Outputformatowner" value="_outputformatowner"/>
      <input name="Outputformattype" value="_outputformattype"/>
      <output name="Newtemplate" setvar="_birhandle"/>
    </invoke>

    <!-- Set an illegal value to the parameter. -->
    <add name="_bsphandle" value="1">
      <only_if><equal_to var1="_testnumber" value2="020302"/></only_if>
    </add>
    <add name="_birhandle" value="1">
      <only_if><equal_to var1="_testnumber" value2="020303"/></only_if>
    </add>

    <!-- Invoke the function BioAPI_GetBIRFromHandle. -->
    <invoke function="BioAPI_GetHeaderFromHandle">
      <input name="BSPHandle" var="_bsphandle"/>
      <input name="Handle" var="_birhandle"/>
      <input name="no_Header" var="_no_header"/>
      <return setvar="return"/>
    </invoke>

    <!-- Extract the error code from the error value. -->
    <invoke activity="ExtractErrorCode" package="fb6ff5b0-7d5e-11de-8a39-0800200c9a66" break_on_break="true">
      <output name="Returnvalue" setvar="return"/>
    </invoke>

    <!-- Assertion -->
    <assert_condition response_if_false="fail">
      <description>
        The function BioAPI_GetHeaderFromHandle has returned the expected return value.
      </description>
      <or>
        <!-- If the "_expected_return_value" parameter is 0xFFFFFFFF,
             then check the "return" parameter is not __BioAPI_OK. (error check only) -->
        <and>
          <equal_to var1="_expected_return_value" var2="_indeterminate_error"/>
          <not_equal_to var1="return" var2="__BioAPI_OK"/>
        </and>
        <and>
          <not_equal_to var1="_expected_return_value" var2="_indeterminate_error"/>
          <equal_to var1="return" var2="_expected_return_value"/>
        </and>
      </or>
    </assert_condition>

    <!-- If Test Number is 020302, make _bsphandle to an valid value. -->
    <subtract name="_bsphandle" value="1">
      <only_if><equal_to var1="_testnumber" value2="020302"/></only_if>
    </subtract>

    <!-- Invoke the functions Detach -->
    <invoke activity="BSPDetach" package="fb6ff5b0-7d5e-11de-8a39-0800200c9a66" break_on_break="true">
      <input name="Bsphandle" var="_bsphandle"/>
      <input name="Bspuuid" var="_bspuuid"/>
    </invoke>

  </activity>

  <!-- ********************************************* -->
  <!-- Activity bound to BioSPI_GetHeaderFromHandle -->
  <!-- ********************************************* -->
  <activity name="SPI_GetHeaderFromHandle">
    <input name="BSPHandle"/>
    <input name="Handle"/>
    <input name="no_Header"/>
    <output name="return"/>

    <!-- check API=SPI -->
    <assert_condition response_if_false="fail">
      <and>
        <equal_to var1="BSPHandle" var2="_bsphandle"/>
        <equal_to var1="no_Header" var2="_no_header"/>
        <equal_to var1="no_BIR" var2="_no_bir"/>
      </and>
    </assert_condition>
  </activity>

</package>
```

8.19 Assertion 3.1 – BioAPI_EnableEvents_And_BioSPI_EnableEvents

Description: This assertion calls BioAPI_EnableEvents with the input parameters described in Default Input Table and Test Condition Table, and checks if the framework under test returns BioAPI_OK or an error value in accordance with the description in Expected Result Table. The assertion is according to the following statement in the subclauses from BioAPI 2.0 standard document (ISO/IEC19784-1:2006), which are described in References in this subclause in this part of ISO/IEC 24709.

Excerpts:

Subclause 8.3.1

```
BioAPI_RETURN BioAPI BioAPI_EnableEvents
          (BioAPI_HANDLE              BSPHandle,
          BioAPI_EVENT_MASK           Events);
```

This function enables the events specified by the Event Mask coming from all the BioAPI Units selected in the BSP attach session identified by the BSP Handle, and disables all other events from those BioAPI Units. Events from other BioAPI Units directly or indirectly managed by the same BSP (possibly selected in other attach sessions but not selected in the specified attach session) are not affected.

A BioAPI_RETURN value indicates success or specifies a particular error condition. The value BioAPI_OK indicates success. All other values represent an error condition.

Subclause 9.3.3.1

```
BioAPI_RETURN BioAPI BioSPI_EnableEvents
          (BioAPI_HANDLE              BSPHandle,
          BioAPI_EVENT_MASK           Events);
```

References: 8.3.1, 9.3.3.1

Scenario:

a) The testing application does the followings:
   1) Initialises the Framework under test, then installs, loads and attaches the testing BSP.
   2) Calls BioAPI_EnableEvents with the conditions given by the Default Input Table and by each row of the Test Condition Table.

b) The testing BSP does the followings:
   1) Checks the parameters and the contents of the pointer variables as described in the column 'BioSPI function' in the Expected Result Table.
   2) Sets the return value of the BioSPI_EnableEvents given by each row of the Test Condition Table, then returns to the Framework.

c) The testing application does the followings:
   1) Checks the return value of the BioAPI_EnableEvents and if it is not as described in the Expected Result Table, then issues a fail conformity response.
   2) Detaches, unloads and uninstalls the testing BSP, then terminates the Framework under test.

Parameters: This assertion uses input parameters described in Test Condition Table below.

Table 58 — Default Input Table for BioAPI_EnableEvent_And_BioSPI_EnableEvent

Number	Input parameter name	Input parameter value (underscore:invalid)
1	BSPHandle	*Valid BSPHandle*
2	EventNotifyInsert	false
3	EventNotifyRemove	false
4	EventNotifyFault	false
5	EventNotifySourcePresent	false
6	EventNotifySourceRemoved	false

Table 59 — Test Condition Table for BioAPI_EnableEvent_And_BioSPI_EnableEvent

Test number	Input parameter name	Input parameter value (underscore:invalid)	Supported options in BSP Schema		Return value (from BioSPI)
			OPERATIONS_MASK	OPTIONS_MASK	
030101	BSPHandle	*Valid BSPHandle*	*EnableEvents*	-	*OK*
030102	BSPHandle	*Invalid BSPHandle*	*EnableEvents*	-	-
030103	EventNotifyInsert	true	*EnableEvents*	-	*OK*
030104	EventNotifyRemove	true	*EnableEvents*	-	*OK*
030105	EventNotifyFault	true	*EnableEvents*	-	*OK*
030106	EventNotifySourcePresent	true	*EnableEvents*	-	*OK*
030107	EventNotifySourceRemoved	true	*EnableEvents*	-	*OK*
030108	(EventNotifyInsert, EventNotifyRemove, EventNotifyFault, EventNotifySourcePresent, EventNotifySourceRemoved)	(true, true, true, true, true)	*EnableEvents*	-	*OK*
030109	BSPHandle	*Valid BSPHandle*	-	-	-

Table 60 — Expected Result Table for BioAPI_EnableEvent_And_BioSPI_EnableEvent

Test number	BioSPI function (BioAPI/BioSPI parameter check)	BioAPI function			Other conditions
		Return value	Output parameter name	Output parameter value	
030101	X	*OK*	-	-	-
030102	-	*indeterminate error*	-	-	-
030103	X	*OK*	-	-	-
030104	X	*OK*	-	-	-
030105	X	*OK*	-	-	-
030106	X	*OK*	-	-	-
030107	X	*OK*	-	-	-
030108	X	*OK*	-	-	-
030109	-	*indeterminate error*	-	-	-

Assertion language package

```
<?xml version="1.0" encoding="UTF-8"?>
<package name="c0c4abd0-792c-11de-8a39-0800200c9a66">
  <author>ISO/IEC JTC1 SC37</author>
  <description>This package contains the assertion "BioAPI_BSPLoad_And_BioSPI_EnableEvents".</description>

  <!-- ************** -->
  <!-- Test assertion -->
  <!-- ************** -->
  <assertion name="BioAPI_BSPLoad_And_BioSPI_EnableEvents" model="frameworkTesting">
    <description>
      This assertion checks if BioAPI_BSPLoad is called with input parameters as described in Test Condition Table,
      the framework under test returns BioAPI_OK or error value in accordance with the description in
      Expected Result Table.
    </description>

    <!-- Parameter given by the CTS by referring to the Test Condition Table. -->
    <input name="_testnumber"/>
    <input name="_operationsmask"/>
    <input name="_optionsmask"/>
    <input name="_eventnotifyinsert"/>
    <input name="_eventnotifyremove"/>
    <input name="_eventnotifyfault"/>
    <input name="_eventnotifysourcepresent"/>
    <input name="_eventnotifysourceremoved"/>

    <!-- Parameter given by the CTS by referring to the Expected Result Table. -->
    <input name="_expected_return_value"/>

    <!-- Parameter given by the CTS. -->
    <input name="_bspuuid"/>

    <!-- Invocation of the primary activity of this assertion. -->
    <invoke activity="BioAPI_EnableEvents_And_BioSPI_EnableEvents"/>

    <!-- Bind activities to check the BioSPI function invoked by the framework under test -->
    <bind activity="SPI_EnableEvents" function="BioSPI_EnableEvents"/>
  </assertion>

  <!-- **************** -->
  <!-- Primary activity -->
  <!-- **************** -->
  <activity name="BioAPI_EnableEvents_And_BioSPI_EnableEvents">
```

```
    <!-- Invoke the activity BSP Attach the framework under test -->
    <invoke activity="BSPAttach" package="fb6ff5b0-7d5e-11de-8a39-0800200c9a66" break_on_break="true">
      <input name="Operationsmask" var="_operationsmask"/>
      <input name="Optionsmask" var="_optionsmask"/>
      <input name="Bspuuid" var="_bspuuid"/>
      <output name="Newbsphandle" setvar="_bsphandle"/>
    </invoke>

    <!-- If Test Number is 020302, make _bsphandle to an illegal value. -->
    <add name="_bsphandle" value="1">
      <only_if><equal_to var1="_testnumber" value2="020302"/></only_if>
    </add>

    <!-- Invoke the function BioSPI_EnableEvents. -->
    <invoke function="BioAPI_EnableEvents">
      <input name="BSPhandle" var="_bsphandle"/>
      <input name="EventNotifyInsert" var="_eventnotifyinsert"/>
      <input name="EventNotifyRemove" var="_eventnotifyremove"/>
      <input name="EventNotifyFault" var="_eventnotifyfault"/>
      <input name="EventNotifySourcePresent" var="_eventnotifysourcepresent"/>
      <input name="EventNotifySourceRemoved" var="_eventnotifysourceremoved"/>
      <return setvar="return"/>
    </invoke>

    <!-- Extract the error code from the error value. -->
    <invoke activity="ExtractErrorCode" package="fb6ff5b0-7d5e-11de-8a39-0800200c9a66" break_on_break="true">
      <output name="Returnvalue" setvar="return"/>
    </invoke>

    <!-- Assertion -->
    <assert_condition response_if_false="fail">
      <description>
        The function BioAPI_EnableEvents has returned the expected return value.
      </description>
      <or>
        <!-- If the "_expected_return_value" parameter is 0xFFFFFFFF,
             then check the "return" parameter is not __BioAPI_OK. (error check only) -->
        <and>
          <equal_to var1="_expected_return_value" var2="_indeterminate_error"/>
          <not_equal_to var1="return" var2="__BioAPI_OK"/>
        </and>
        <and>
          <not_equal_to var1="_expected_return_value" var2="_indeterminate_error"/>
          <equal_to var1="return" var2="_expected_return_value"/>
        </and>
      </or>
    </assert_condition>

    <!-- If Test Number is 020302, make _bsphandle to an valid value. -->
    <subtract name="_bsphandle" value="1">
      <only_if><equal_to var1="_testnumber" value2="020302"/></only_if>
    </subtract>

    <!-- Invoke the functions Detach -->
    <invoke activity="BSPDetach" package="fb6ff5b0-7d5e-11de-8a39-0800200c9a66" break_on_break="true">
      <input name="Bsphandle" var="_bsphandle"/>
      <input name="Bspuuid" var="_bspuuid"/>
    </invoke>

  </activity>

  <!-- ************************************** -->
  <!-- Activity bound to BioSPI_EnableEvents -->
  <!-- ************************************** -->
  <activity name="SPI_EnableEvents">
    <input name="BSPHandle"/>
    <input name="EventNotifyInsert"/>
    <input name="EventNotifyRemove"/>
    <input name="EventNotifyFault"/>
    <input name="EventNotifySourcePresent"/>
    <input name="EventNotifySourceRemoved"/>
    <output name="return"/>

    <!-- check API=SPI -->
    <assert_condition response_if_false="fail">
      <and>
        <equal_to var1="BSPHandle" var2="_bsphandle"/>
        <equal_to var1="EventNotifyInsert" var2="_eventnotifyinsert"/>
        <equal_to var1="EventNotifyRemove" var2="_eventnotifyremove"/>
        <equal_to var1="EventNotifyFault" var2="_eventnotifyfault"/>
        <equal_to var1="EventNotifySourcePresent" var2="_eventnotifysourcepresent"/>
        <equal_to var1="EventNotifySourceRemoved" var2="_eventnotifysourceremoved"/>
      </and>
    </assert_condition>
  </activity>
</package>
```

8.20 Assertion 3.2 – BioAPI_SetGUICallbacks_And_BioSPI_SetGUICallbacks

Description: This assertion calls BioAPI_SetGUICallbacks with the input parameters described in Default Input Table and Test Condition Table, and checks if the framework under test returns BioAPI_OK or an error value in accordance with the description in Expected Result Table. The assertion is according to the following statement in the subclauses from BioAPI 2.0 standard document (ISO/IEC19784-1:2006), which are described in References in this subclause in this part of ISO/IEC 24709.

Excerpts:

Subclause 8.3.2

```
BioAPI_RETURN BioAPI BioAPI_SetGUICallbacks
        (BioAPI_HANDLE                          BSPHandle,
        BioAPI_GUI_STREAMING_CALLBACK           GuiStreamingCallback,
        void                                    *GuiStreamingCallbackCtx,
        BioAPI_GUI_STATE_CALLBACK               GuiStateCallback,
        void                                    *GuiStateCallbackCtx);
```

This function allows the application to establish callbacks so that the application may control the "look-and-feel" of the biometric user interface by receiving from the BSP a sequence of bit-map images, called streaming data, for display by the biometric application as well as state information.

NOTE Not all BSPs support the provision of streaming data.

A BioAPI_RETURN value indicates success or specifies a particular error condition. The value BioAPI_OK indicates success. All other values represent an error condition.

Subclause 9.3.3.2

```
BioAPI_RETURN BioAPI BioSPI_SetGUICallbacks
        (BioAPI_HANDLE                          BSPHandle,
        BioAPI_GUI_STREAMING_CALLBACK           GuiStreamingCallback,
        void                                    *GuiStreamingCallbackCtx,
        BioAPI_GUI_STATE_CALLBACK               GuiStateCallback,
        void                                    *GuiStateCallbackCtx);
```

References: 8.3.2, 9.3.3.2

Scenario:

a) The testing application does the followings:
   1) Initialises the Framework under test, then installs, loads and attaches the testing BSP.
   2) Calls BioAPI_SetGUICallbacks with the conditions given by the Default Input Table and by each row of the Test Condition Table.

b) The testing BSP does the followings:
   1) Checks the parameters and the contents of the pointer variables as described in the column 'BioSPI function' in the Expected Result Table.
   2) Sets the return value of the BioSPI_SetGUICallbacks given by each row of the Test Condition Table, then returns to the Framework.

c) The testing application does the followings:
   1) Checks the return value of the BioAPI_SetGUICallbacks and if it is not as described in the Expected Result Table, then issues a fail conformity response.
   2) Detaches, unloads and uninstalls the testing BSP, then terminates the Framework under test.

Parameters: The input parameters described below table are used for this assertion.

Table 61 — Default Input Table for BioAPI_SetGUICallbacks_And_BioSPI_SetGUICallbacks

Number	Input parameter name	Input parameter value (underscore:invalid)
1	BSPHandle	*Valid BSPHandle*
2	GuiStreamingCallback	*
3	GuiStreamingCallbackCtx	*
4	GuiStateCallback	*
5	GuiStateCallbackCtx	*

Table 62 — Test Condition Table for BioAPI_SetGUICallback_And_BioSPI_SetGUICallbacks

Test number	Input parameter name	Input parameter value (underscore:invalid)	Supported options in BSP Schema		Return value (from BioSPI)
			OPERATIONS_MASK	OPTIONS_MASK	
030201	BSPHandle	*Valid BSPHandle*	*SetGUICallbacks*	-	*OK*
030202	BSPHandle	*Invalid BSPHandle*	*SetGUICallbacks*	-	-
030203	GuiStreamingCallback	0	*SetGUICallbacks*	-	*FUNCTION_NOT_SUPPORTED*
030204	GuiStreamingCallbackCtx	0	*SetGUICallbacks*	-	*INVALID_DATA*
030205	GuiStateCallback	0	*SetGUICallbacks*	-	*INVALID_POINTER*
030206	GuiStateCallbackCtx	0	*SetGUICallbacks*	-	*INVALID_POINTER*
030207	BSPHandle	*Valid BSPHandle*	-	-	-

Table 63 — Expected Result Table for BioAPI_SetGUICallback_And_BioSPI_SetGUICallbacks

Test number	BioSPI function (BioAPI/BioSPI parameter check)	BioAPI function			Other conditions
		Return value	Output parameter name	Output parameter value	
030201	X	*OK*	-	-	-
030202	-	*INVALID_BSP_HANDLE*	-	-	-
030203	X	*indeterminate error*	-	-	-
030204	X	*indeterminate error*	-	-	-
030205	X	*INVALID_POINTER*	-	-	-
030206	X	*indeterminate error*	-	-	-
030207	-	*indeterminate error*	-	-	-

Assertion language package

```
<?xml version="1.0" encoding="UTF-8"?>
<package name="f0a2b310-792c-11de-8a39-0800200c9a66">
  <author>ISO/IEC JTC1 SC37</author>
  <description>
    This package contains the assertion "BioAPI_SetGUICallbacks_And_BioSPI_SetGUICallbacks".
  </description>

  <!-- ************** -->
  <!-- Test assertion -->
  <!-- ************** -->
  <assertion name="BioAPI_SetGUICallbacks_And_BioSPI_SetGUICallbacks" model="frameworkTesting">
    <description>
      This assertion checks if BioAPI_SetGUICallbacks is called with input parameters as described in Test
      Condition Table, the framework under test returns BioAPI_OK or error value in accordance with the description
      in Expected Result Table.
    </description>

    <!-- Parameter given by the CTS by referring to the Test Condition Table. -->
    <input name="_testnumber"/>
    <input name="_operationsmask"/>
    <input name="_optionsmask"/>
    <input name="_guistreamingcallback"/>
    <input name="_guistreamingcallbackctx"/>
    <input name="_guistatecallback"/>
    <input name="_guistatecallbackctx"/>

    <!-- Parameter given by the CTS by referring to the Expected Result Table. -->
    <input name="_expected_return_value"/>

    <!-- Parameter given by the CTS. -->
    <input name="_bspuuid"/>

    <!-- Invocation of the primary activity of this assertion. -->
    <invoke activity="BioAPI_SetGUICallbacks_And_BioSPI_SetGUICallbacks"/>
    <!-- Bind activities to check the BioSPI function invoked by the framework under test -->
    <bind activity="SPI_SetGUICallbacks" function="BioSPI_SetGUICallbacks"/>
  </assertion>

  <!-- **************** -->
  <!-- Primary activity -->
  <!-- **************** -->
  <activity name="BioAPI_SetGUICallbacks_And_BioSPI_SetGUICallbacks">

    <!-- Invoke the activity BSP Attach the framework under test -->
    <invoke activity="BSPAttach" package="fb6ff5b0-7d5e-11de-8a39-0800200c9a66" break_on_break="true">
      <input name="Operationsmask" var="_operationsmask"/>
      <input name="Optionsmask" var="_optionsmask"/>
      <input name="Bspuuid" var="_bspuuid"/>
      <output name="Newbsphandle" setvar="_bsphandle"/>
    </invoke>

    <!-- If Test Number is 030202, make _bsphandle to an illegal value. -->
    <add name="_bsphandle" value="1">
      <only_if><equal_to var1="_testnumber" value2="030202"/></only_if>
    </add>

    <!-- Invoke the function BioAPI_SetGUICallbacks. -->
    <invoke function="BioAPI_SetGUICallbacks">
      <input name="BSPhandle" var="_bsphandle"/>
      <input name="GuiStreamingCallback" var="_guistreamingcallback"/>
      <input name="GuiStreamingCallbackCtx" var="_guistreamingcallbackctx"/>
      <input name="GuiStateCallback" var="_guistatecallback"/>
      <input name="GuiStateCallbackCtx" var="_guistatecallbackctx"/>
      <return setvar="return"/>
    </invoke>

    <!-- Extract the error code from the error value. -->
    <invoke activity="ExtractErrorCode" package="fb6ff5b0-7d5e-11de-8a39-0800200c9a66" break_on_break="true">
      <output name="Returnvalue" setvar="return"/>
    </invoke>

    <!-- Assertion -->
    <assert_condition response_if_false="fail">
      <description>
        The function BioAPI_SetGUICallbacks has returned the expected return value.
      </description>
      <or>
        <!-- If the "_expected_return_value" parameter is 0xFFFFFFFF,
             then check the "return" parameter is not __BioAPI_OK. (error check only) -->
        <and>
          <equal_to var1="_expected_return_value" var2="_indeterminate_error"/>
          <not_equal_to var1="return" var2="__BioAPI_OK"/>
        </and>
        <and>
          <not_equal_to var1="_expected_return_value" var2="_indeterminate_error"/>
          <equal_to var1="return" var2="_expected_return_value"/>
        </and>
      </or>
    </assert_condition>

    <!-- If Test Number is 030202, make _bsphandle to an valid value. -->
    <subtract name="_bsphandle" value="1">
      <only_if><equal_to var1="_testnumber" value2="030202"/></only_if>
    </subtract>

    <!-- Invoke the functions Detach -->
    <invoke activity="BSPDetach" package="fb6ff5b0-7d5e-11de-8a39-0800200c9a66" break_on_break="true">
      <input name="Bsphandle" var="_bsphandle"/>
      <input name="Bspuuid" var="_bspuuid"/>
    </invoke>
  </activity>

  <!-- **************************************** -->
  <!-- Activity bound to BioSPI_SetGUICallbacks -->
  <!-- **************************************** -->
  <activity name="SPI_SetGUICallbacks">
    <input name="BSPhandle"/>
    <input name="GuiStreamingCallback"/>
    <input name="GuiStreamingCallbackCtx"/>
    <input name="GuiStateCallback"/>
    <input name="GuiStateCallbackCtx"/>
    <output name="return"/>

    <!-- check API=SPI -->
    <assert_condition response_if_false="fail">
      <and>
        <equal_to var1="BSPHandle" var2="_bsphandle"/>
        <equal_to var1="GuiStreamingCallback" var2="_guistreamingcallback"/>
        <equal_to var1="GuiStreamingCallbackCtx" var2="_guistreamingcallbackctx"/>
        <equal_to var1="GuiStateCallback" var2="_guistatecallback"/>
        <equal_to var1="GuiStateCallbackCtx" var2="_guistatecallbackctx"/>
      </and>
    </assert_condition>
  </activity>
</package>
```

8.21 Assertion 4.1 – BioAPI_Capture_And_BioSPI_Capture

Description: This assertion calls BioAPI_Capture with the input parameters described in Default Input Table and Test Condition Table, and checks if the framework under test returns BioAPI_OK or an error value in accordance with the description in Expected Result Table. The assertion is according to the following statement in the subclauses from BioAPI 2.0 standard document (ISO/IEC19784-1:2006), which are described in References in this subclause in this part of ISO/IEC 24709.

Excerpts:

Subclause 8.4.1

```
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);
```

This function captures samples for the purpose specified, and the BSP returns either an "intermediate" type BIR (if the BioAPI_Process function needs to be called), or a "processed" BIR (if not). The Purpose is recorded in the header of the CapturedBIR. If AuditData is non-NULL, a BIR of type "raw" may be returned. The function returns handles to whatever data is collected, and all local operations can be completed through use of the handles. If the application needs to acquire the data either to store it in a BIR database or to send it to a server, the application can retrieve the data with the BioAPI_GetBIRFromHandle function.

By default, the BSP is responsible for providing the user interface associated with the capture operation. The application may, however, request control of the GUI "look-and-feel" by providing a GUI callback pointer in BioAPI_SetGUICallbacks. See clause C.7 for additional explanation of user interface features.

Capture serializes use of the sensor device. If two or more biometric applications are racing for the sensor, the losers will wait until the operation completes or the timeout expires. This serialization takes place in all functions that capture data. The BSP is responsible for serializing. It may do this by either returning 'busy' (BioAPI_UNIT_IN_USE) or by queuing requests.

The BIR Handle returned by the function shall be released by the application (via BioAPI_FreeBIRHandle) when it is no longer needed.

A BioAPI_RETURN value indicates success or specifies a particular error condition. The value BioAPI_OK indicates success. All other values represent an error condition.

Subclause 9.3.4.1

```
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);
```

References: 8.4.1, 9.3.4.1

Scenario:

a) The testing application does the followings:

1) Initialises the Framework under test, then installs, loads and attaches the testing BSP.

2) Calls BioAPI_Capture with the conditions given by the Default Input Table and by each row of the Test Condition Table.

b) The testing BSP does the followings:

1) Checks the parameters and the contents of the pointer variables as described in the column 'BioSPI function' in the Expected Result Table.

2) Sets the return value of the BioSPI_Capture given by each row of the Test Condition Table, then returns to the Framework.

c) The testing application does the followings:

1) Checks the return value of the BioAPI_Capture and if it is not as described in the Expected Result Table, then issues a fail conformity response.

2) Detaches, unloads and uninstalls the testing BSP, then terminates the Framework under test.

Parameters: The input parameters described below table are used for this assertion.

Table 64 — Default Input Table for BioAPI_Capture_And_BioSPI_Capture

Number	Input parameter name	Input parameter value (underscore:invalid)
1	BSPHandle	*Valid BSPHandle*
2	Purpose	__BioAPI_NO_PURPOSE_AVAILABLE
3	Left	false
4	Right	false
5	Thumb	false
6	PointerFinger	false
7	MiddleFinger	false
8	RingFinger	false
9	LittleFinger	false
10	Multiple	false
11	OutputFormatOwner	N/A
12	OutputFormatType	N/A
13	no_CapturedBIR	false
14	TimeOut	-1
15	no_AuditData	true

NOTE: These parameter names "Left","Right" ,"Thumb","PointerFinger","MiddleFinger","RingFinger","LittleFinger" and "Multiple" are based on the definitions in ISO/IEC 24709-1 9.2.6.

Table 65 — Test Condition Table for BioAPI_Capture_And_BioSPI_Capture

Test number	Input parameter name	Input parameter value (underscore:invalid)	Supported options in BSP Schema		Return value (from BioSPI)
			OPERATIONS_MASK	OPTIONS_MASK	
040101	BSPHandle	*Valid BSPHandle*	*Capture*	-	*OK*
040102	BSPHandle	*<u>Invalid BSPHandle</u>*	*Capture*	-	-
040103	Purpose	__BioAPI_PURPOSE _VERIFY	*Capture*	-	*OK*
040104	Purpose	__BioAPI_PURPOSE _IDENTIFY	*Capture*	-	*OK*
040105	Purpose	__BioAPI_PURPOSE _ENROLL	*Capture*	-	*OK*
040106	Purpose	__BioAPI_PURPOSE _ENROLL_FOR_VER IFICATION_ONLY	*Capture*	-	*OK*
040107	Purpose	__BioAPI_PURPOSE _ENROLL_FOR_IDE NTIFICATION_ONLY	*Capture*	-	*OK*
040108	Purpose	__BioAPI_PURPOSE _AUDIT	*Capture*	-	*OK*
040109	Purpose	*<u>Invalid Purpose</u>*	*Capture*	-	*INCONSISTENT_P URPOSE*
040110	Left	true	*Capture*	*SubtypeToCapture*	*OK*

040111	Left	true	*Capture*	-	*INVALID_DATA*
040112	Right	true	*Capture*	*SubtypeToCapture*	*OK*
040113	Right	true	*Capture*	-	*INVALID_DATA*
040114	(Left, Thumb)	(true, true)	*Capture*	*SubtypeToCapture*	*OK*
040115	(Left, Thumb)	(true, true)	*Capture*	-	*INVALID_DATA*
040116	(Left, PointerFinger)	(true, true)	*Capture*	*SubtypeToCapture*	*OK*
040117	(Left, PointerFinger)	(true, true)	*Capture*	-	*INVALID_DATA*
040118	(Left, MiddleFinger)	(true, true)	*Capture*	*SubtypeToCapture*	*OK*
040119	(Left, MiddleFinger)	(true, true)	*Capture*	-	*INVALID_DATA*
040120	(Left, RingFinger)	(true, true)	*Capture*	*SubtypeToCapture*	*OK*
040121	(Left, RingFinger)	(true, true)	*Capture*	-	*INVALID_DATA*
040122	(Left, LittleFinger)	(true, true)	*Capture*	*SubtypeToCapture*	*OK*
040123	(Left, LittleFinger)	(true, true)	*Capture*	-	*INVALID_DATA*
040124	(Right, Thumb)	(true, true)	*Capture*	*SubtypeToCapture*	*OK*
040125	(Right, Thumb)	(true, true)	*Capture*	-	*INVALID_DATA*
040126	(Right, PointerFinger)	(true, true)	*Capture*	*SubtypeToCapture*	*OK*
040127	(Right, PointerFinger)	(true, true)	*Capture*	-	*INVALID_DATA*
040128	(Right, MiddleFinger)	(true, true)	*Capture*	*SubtypeToCapture*	*OK*
040129	(Right, MiddleFinger)	(true, true)	*Capture*	-	*INVALID_DATA*
040130	(Right, RingFinger)	(true, true)	*Capture*	*SubtypeToCapture*	*OK*
040131	(Right, RingFinger)	(true, true)	*Capture*	-	*INVALID_DATA*
040132	(Right, LittleFinger)	(true, true)	*Capture*	*SubtypeToCapture*	*OK*
040133	(Right, LittleFinger)	(true, true)	*Capture*	-	*INVALID_DATA*
040134	Multiple	true	*Capture*	*SubtypeToCapture*	*OK*
040135	Multiple	true	*Capture*	-	*INVALID_DATA*
040136	Thumb	true	*Capture*	*SubtypeToCapture*	*OK*
040137	Thumb	true	*Capture*	-	*INVALID_DATA*
040138	PointerFinger	true	*Capture*	*SubtypeToCapture*	*OK*
040139	PointerFinger	true	*Capture*	-	*INVALID_DATA*
040140	MiddleFinger	true	*Capture*	*SubtypeToCapture*	*OK*
040141	MiddleFinger	true	*Capture*	-	*INVALID_DATA*
040142	RingFinger	true	*Capture*	*SubtypeToCapture*	*OK*
040143	RingFinger	true	*Capture*	-	*INVALID_DATA*
040144	LittleFinger	true	*Capture*	*SubtypeToCapture*	*OK*
040145	LittleFinger	true	*Capture*	-	*INVALID_DATA*
040146	(Left, Right, Thumb, PointerFinger, MiddleFinger, RingFinger, LittleFinger, Multiple)	(true, true, true, true, true, true, true, true)	*Capture*	-	*INVALID_DATA*
040147	no_CapturedBIR	true	*Capture*	-	*INVALID_OUTPUT_POINTER*
040148	Timeout	0	*Capture*	-	*TIMEOUT_EXPIRED*
040149	Timeout	DefaultCaptureTimeout	*Capture*	-	*OK*
040150	Timeout	-2	*Capture*	-	*INVALID_DATA*
040151	no_AuditData	false	*Capture*	*Raw QualityRaw*	*OK*
040152	no_AuditData	false	*Capture*	-	*FUNCTION_NOT_SUPPORTED*
040153	BSPHandle	*Valid BSPHandle*	-	-	-

NOTE1: For the detail of the cell of the test number row "040151" and the column "Parameter values(underscore:invalid)", see Table1.
NOTE2: These parameter names "Left","Right" ,"Thumb","PointerFinger","MiddleFinger","RingFinger","LittleFinger" and "Multiple" are based on the definitions in ISO/IEC 24709-1 9.2.6.

Table 66 — Expected Result Table for BioAPI_Capture_And_BioSPI_Capture

Test number	BioSPI function (BioAPI/BioSPI parameter check)	BioAPI function			Other conditions
		Return value	Output parameter name	Output parameter value	
040101	X	*OK*	-	-	-
040102	-	*indeterminate error*	-	-	-
040103	X	*OK*	-	-	-
040104	X	*OK*	-	-	-
040105	X	*OK*	-	-	-
040106	X	*OK*	-	-	-
040107	X	*OK*	-	-	-
040108	X	*OK*	-	-	-
040109	X	*INCONSISTENT_ PURPOSE*	-	-	-
040110	X	*OK*	-	-	-
040111	X	*indeterminate error*	-	-	-
040112	X	*OK*	-	-	-
040113	X	*indeterminate error*	-	-	-
040114	X	*OK*	-	-	-
040115	X	*indeterminate error*	-	-	-
040116	X	*OK*	-	-	-
040117	X	*indeterminate error*	-	-	-
040118	X	*OK*	-	-	-
040119	X	*indeterminate error*	-	-	-
040120	X	*OK*	-	-	-
040121	X	*indeterminate error*	-	-	-
040122	X	*OK*	-	-	-
040123	X	*indeterminate error*	-	-	-
040124	X	*OK*	-	-	-
040125	X	*indeterminate error*	-	-	-
040126	X	*OK*	-	-	-
040127	X	*indeterminate error*	-	-	-
040128	X	*OK*	-	-	-
040129	X	*indeterminate error*	-	-	-
040130	X	*OK*	-	-	-
040131	X	*indeterminate error*	-	-	-
040132	X	*OK*	-	-	-
040133	X	*indeterminate error*	-	-	-
040134	X	*OK*	-	-	-
040135	X	*indeterminate error*	-	-	-
040136	X	*OK*	-	-	-
040137	X	*indeterminate error*	-	-	-
040138	X	*OK*	-	-	-
040139	X	*indeterminate error*	-	-	-
040140	X	*OK*	-	-	-
040141	X	*indeterminate error*	-	-	-
040142	X	*OK*	-	-	-
040143	X	*indeterminate error*	-	-	-
040144	X	*OK*	-	-	-
040145	X	*indeterminate error*	-	-	-
040146	X	*indeterminate error*	-	-	-
040147	X	*indeterminate error*	-	-	-
040148	X	*TIMEOUT_EXPIRED*	-	-	-
040149	X	*OK*	-	-	-
040150	X	*indeterminate error*	-	-	-
040151	X	*OK*	-	-	-
040152	X	*indeterminate error*	-	-	-
040153	-	*indeterminate error*	-	-	-

Assertion language package

```
<?xml version="1.0" encoding="UTF-8"?>
<package name="aaec0fa0-792d-11de-8a39-0800200c9a66">
  <author>ISO/IEC JTC1 SC37</author>
  <description>This package contains the assertion "BioAPI_Capture_And_BioSPI_Capture".</description>

  <!-- ************** -->
  <!-- Test assertion -->
  <!-- ************** -->
  <assertion name="BioAPI_Capture_And_BioSPI_Capture" model="frameworkTesting">
    <description>
      This assertion checks if BioAPI_Capture is called with input parameters as described in Test Condition Table,
      the framework under test returns BioAPI_OK or error value in accordance with the description in Expected
      Result Table.
    </description>

    <!-- Parameter given by the CTS by referring to the Test Condition Table. -->
    <input name="_testnumber"/>
    <input name="_operationsmask"/>
    <input name="_optionsmask"/>
    <input name="_purpose"/>
    <input name="_left"/>
    <input name="_right"/>
    <input name="_thumb"/>
    <input name="_pointerfinger"/>
    <input name="_middlefinger"/>
    <input name="_ringfinger"/>
    <input name="_littlefinger"/>
    <input name="_multiple"/>
    <input name="_outputformatowner"/>
    <input name="_outputformattype"/>
    <input name="_no_capturedbir"/>
    <input name="_no_auditdata"/>

    <!-- Parameter given by the CTS by referring to the Expected Result Table. -->
    <input name="_expected_return_value"/>

    <!-- Parameter given by the CTS. -->
    <input name="_bspuuid"/>

    <!-- Invocation of the primary activity of this assertion. -->
    <invoke activity="BioAPI_Capture_And_BioSPI_Capture"/>

    <!-- Bind activities to check the BioSPI function invoked by the framework under test -->
    <bind activity="SPI_Capture" function="BioSPI_Capture"/>
  </assertion>

  <!-- **************** -->
  <!-- Primary activity -->
  <!-- **************** -->
  <activity name="BioAPI_Capture_And_BioSPI_Capture">

    <!-- Invoke the activity BSP Attach the framework under test -->
    <invoke activity="BSPAttach" package="fb6ff5b0-7d5e-11de-8a39-0800200c9a66" break_on_break="true">
      <input name="Operationsmask" var="_operationsmask"/>
      <input name="Optionsmask" var="_optionsmask"/>
      <input name="Bspuuid" var="_bspuuid"/>
      <output name="Newbsphandle" setvar="_bsphandle"/>
    </invoke>

    <!-- If Test Number is 040102, make _bsphandle to an illegal value. -->
    <add name="_bsphandle" value="1">
      <only_if><equal_to var1="_testnumber" value2="040102"/></only_if>
    </add>

    <!-- Invoke the function BioAPI_Capture. -->
    <invoke function="BioAPI_Capture">
      <input name="BSPHandle" var="_bsphandle"/>
      <input name="Purpose" var="_purpose"/>
      <input name="Left" var="_left"/>
      <input name="Right" var="_right"/>
      <input name="Thumb" var="_thumb"/>
      <input name="PointerFinger" var="_pointerfinger"/>
      <input name="MiddleFinger" var="_middlefinger"/>
      <input name="RingFinger" var="_ringfinger"/>
      <input name="LittleFinger" var="_littlefinger"/>
      <input name="Multiple" var="_multiple"/>
      <input name="OutputFormatOwner" var="_outputformatowner"/>
      <input name="OutputFormatType" var="_outputformattype"/>
      <input name="no_CapturedBIR" var="_no_capturedbir"/>
      <input name="no_AuditData" var="_no_auditdata"/>
      <output name="CapturedBIR" setvar="capturedbir"/>
      <output name="AuditData" setvar="auditdata"/>
      <return setvar="return"/>
    </invoke>

    <!-- Extract the error code from the error value. -->
    <invoke activity="ExtractErrorCode" package="fb6ff5b0-7d5e-11de-8a39-0800200c9a66" break_on_break="true">
      <output name="Returnvalue" setvar="return"/>
    </invoke>

    <!-- Assertion -->
    <assert_condition response_if_false="fail">
      <description>
        The function BioAPI_Capture has returned the expected return value.
      </description>
      <or>
        <!-- If the "_expected_return_value" parameter is 0xFFFFFFFF,
             then check the "return" parameter is not __BioAPI_OK. (error check only) -->
        <and>
```

```
          <equal_to var1="_expected_return_value" var2="_indeterminate_error"/>
          <not_equal_to var1="return" var2="__BioAPI_OK"/>
        </and>
        <and>
          <not_equal_to var1="_expected_return_value" var2="_indeterminate_error"/>
          <equal_to var1="return" var2="_expected_return_value"/>
        </and>
      </or>
    </assert_condition>

    <!-- If Test Number is 040102, make _bsphandle to an valid value. -->
    <subtract name="_bsphandle" value="1">
      <only_if><equal_to var1="_testnumber" value2="040102"/></only_if>
    </subtract>

    <!-- Invoke the functions Detach -->
    <invoke activity="BSPDetach" package="fb6ff5b0-7d5e-11de-8a39-0800200c9a66" break_on_break="true">
      <input name="Bsphandle" var="_bsphandle"/>
      <input name="Bspuuid" var="_bspuuid"/>
    </invoke>
  </activity>

  <!-- ******************************** -->
  <!-- Activity bound to BioSPI_Capture -->
  <!-- ******************************** -->
  <activity name="SPI_Capture">
    <input name="BSPHandle"/>
    <input name="Purpose"/>
    <input name="Left"/>
    <input name="Right"/>
    <input name="Thumb"/>
    <input name="PointerFinger"/>
    <input name="MiddleFinger"/>
    <input name="RingFinger"/>
    <input name="LittleFinger"/>
    <input name="Multiple"/>
    <input name="OutputFormatOwner"/>
    <input name="OutputFormatType"/>
    <input name="no_CapturedBIR"/>
    <input name="TimeOut"/>
    <input name="no_Auditdata"/>
    <output name="CapturedBIR"/>
    <output name="AuditData"/>
    <output name="return"/>

    <!-- check API=SPI -->
    <assert_condition response_if_false="fail">
      <and>
        <equal_to var1="BSPHandle" var2="_bsphandle"/>
        <equal_to var1="Purpose" var2="_purpose"/>
        <equal_to var1="Left" var2="_left"/>
        <equal_to var1="Right" var2="_right"/>
        <equal_to var1="Thumb" var2="_thumb"/>
        <equal_to var1="PointerFinger" var2="_pointerfinger"/>
        <equal_to var1="MiddleFinger" var2="_middlefinger"/>
        <equal_to var1="RingFinger" var2="_ringfinger"/>
        <equal_to var1="LittleFinger" var2="_littlefinger"/>
        <equal_to var1="Multiple" var2="_multiple"/>
        <equal_to var1="OutputFormatOwner" var2="_outputformatowner"/>
        <equal_to var1="OutputFormatType" var2="_outputformattype"/>
        <equal_to var1="no_CapturedBIR" var2="_no_capturedbir"/>
        <equal_to var1="TimeOut" var2="_timeout"/>
        <equal_to var1="no_Auditdata" var2="_no_auditdata"/>
      </and>
    </assert_condition>
  </activity>
</package>
```

8.22 Assertion 4.2 – BioAPI_CreateTemplate_And_BioSPI_CreateTemplate

Description: This assertion calls BioAPI_CreateTemplate with the input parameters described in Default Input Table and Test Condition Table, and checks if the framework under test returns BioAPI_OK or an error value in accordance with the description in Expected Result Table. The assertion is according to the following statement in the subclauses from BioAPI 2.0 standard document (ISO/IEC19784-1:2006), which are described in References in this subclause in this part of ISO/IEC 24709.

Excerpts:

Subclause 8.4.2

```
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);
```

This function takes a BIR containing biometric data in intermediate form for the purpose of creating a new enrollment template. A new BIR is constructed from the CapturedBIR, and (optionally) it may perform an update based on an existing ReferenceTemplate. The old ReferenceTemplate remains unchanged.

The optional input ReferenceTemplate is provided for use in creating the NewTemplate, if the BSP supports this capability. When present, use of the input ReferenceTemplate by the BSP, to create the output NewTemplate is optional.

If the BSP supports an internal or BSP-controlled BIR database (e.g., smartcard or identification engine), it may optionally return the UUID assigned to the newly created NewTemplate as stored within that BSP-controlled BIR database. The UUID value shall be the same as that included in the BIR header, if present.

The BIR handle returned by the function shall be released by the application (via BioAPI_FreeBIRHandle) when it is no longer needed. The BIR may be retrieved by calling BioAPI_GetBIRFromHandle, which also releases the handle.

A BioAPI_RETURN value indicates success or specifies a particular error condition. The value BioAPI_OK indicates success. All other values represent an error condition.

Subclause 9.3.4.2

```
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);
```

References: 8.4.2, 9.3.4.2

Scenario:

a) The testing application does the followings:

1) Initialises the Framework under test, then installs, loads and attaches the testing BSP.
2) Captures a biometric sample to obtain a BIR handle.
3) If the Test Number is 040206, 040207, 040208 or 040209, Enrolls to obtain a BIR handle.
4) Calls BioAPI_CreateTemplate with the conditions given by the Default Input Table and by each row of the Test Condition Table.

b) The testing BSP does the followings:

1) Checks the parameters and the contents of the pointer variables as described in the column 'BioSPI function' in the Expected Result Table.
2) Sets the return value of the BioSPI_CreateTemplate given by each row of the Test Condition Table, then returns to the Framework.

c) The testing application does the followings:

1) Checks the return value of the BioAPI_CreateTemplate and if it is not as described in the Expected Result Table, then issues a fail conformity response.
2) Detaches, unloads and uninstalls the testing BSP, then terminates the Framework under test.

Parameters: The input parameters described below table are used for this assertion.

Table 67 — Default Input Table for BioAPI_CreateTemplate_And_BioSPI_CreateTemplate

Number	Input parameter name	Input parameter value (underscore:invalid)
1	BSPHandle	*Valid BSPHandle*
2	CapturedBIR_Form	2
3	CapturedBIR_BIRHandle	*Valid BIRHandle*
4	ReferenceTemplate_Form	0
5	OutputFormatOwner	N/A
6	OutputFormatType	N/A
7	no_NewTemplate	false
8	Payload	N/A
7	no_TemplateUUID	true

Table 68 — Test Condition Table for BioAPI_CreateTemplate_And_BioSPI_CreateTemplate

Test number	Input parameter name	Input parameter value (underscore:invalid)	Supported options in BSP Schema		Return value (from BioSPI)
			OPERATIONS_MASK	OPTIONS_MASK	
040201	BSPHandle	*Valid BSPHandle*	*CreateTemplate Capture*	-	*OK*
040202	BSPHandle	*Invalid BSPHandle*	*CreateTemplate Capture*	-	-
040203	CapturedBIR_Form	0	*CreateTemplate Capture*	-	*INVALID_INPUT_POINTER*
040204	CapturedBIR_BIRHandle	*Valid BIRHandle*	*CreateTemplate Capture*	-	*OK*
040205	CapturedBIR_BIRHandle	*Invalid BIRHandle*	*CreateTemplate Capture*	-	*INVALID_BIR_HANDLE*
040206	(ReferenceTemplate_Form, ReferenceTemplate_BIRHandle)	(2, *Valid BIRHandle)*	*CreateTemplate Capture DatabaseOperation*	*TemplateUpdate*	*OK*
040207	(ReferenceTemplate_Form, ReferenceTemplate_BIRHandle)	(2, *Valid BIRHandle)*	*CreateTemplate Capture DatabaseOperation*	-	*INVALID_DATA*
040208	(ReferenceTemplate_Form, ReferenceTemplate_BIRHandle)	(2, *Invalid BIRHandle)*	*CreateTemplate Capture DatabaseOperation*	*TemplateUpdate*	*INVALID_DATA*
040209	(ReferenceTemplate_Form, ReferenceTemplate_BIRHandle)	(2, *Invalid BIRHandle)*	*CreateTemplate Capture DatabaseOperation*	-	*INVALID_DATA*
040210	no_NewTemplate	true	*CreateTemplate Capture*	-	*INVALID_OUTPUT_POINTER*
040211	no_TemplateUUID	false	*CreateTemplate Capture*	-	*OK*
040212	BSPHandle	*Valid BSPHandle*	*Capture*	-	-

Table 69 — Expected Result Table for BioAPI_CreateTemplate_And_BioSPI_CreateTemplate

Test number	BioSPI function (BioAPI/BioSPI parameter check)	BioAPI function			Other conditions
		Return value	Output parameter name	Output parameter value	
040201	X	*OK*	-	-	-
040202	-	*indeterminate error*	-	-	-
040203	X	*indeterminate error*	-	-	-
040204	X	*OK*	-	-	-
040205	X	*INVALID_BIR_HANDLE*	-	-	-
040206	X	*OK*	-	-	-
040207	X	*indeterminate error*	-	-	-
040208	X	*INVALID_BIR_HANDLE*	-	-	-
040209	X	*INVALID_BIR_HANDLE*	-	-	-
040210	X	*indeterminate error*	-	-	-
040211	X	*OK*	-	-	-
040212	-	*indeterminate error*	-	-	-

Assertion language package

```
<?xml version="1.0" encoding="UTF-8"?>
<package name="b851a060-792d-11de-8a39-0800200c9a66">
  <author>ISO/IEC JTC1 SC37</author>
  <description>This package contains the assertion "BioAPI_CreateTemplate_And_BioSPI_CreateTemplate".</description>

  <!-- ************** -->
  <!-- Test assertion -->
  <!-- ************** -->
  <assertion name="BioAPI_CreateTemplate_And_BioSPI_CreateTemplate" model="frameworkTesting">
    <description>
      This assertion checks if BioAPI_CreateTemplate is called with input parameters as described in Test Condition
      Table, the framework under test returns BioAPI_OK or error value in accordance with the description in
      Expected Result Table.
    </description>

    <!-- Parameter given by the CTS by referring to the Test Condition Table. -->
    <input name="_testnumber"/>
    <input name="_operationsmask1"/>
    <input name="_optionsmask1"/>
    <input name="_operationsmask2"/>
    <input name="_optionsmask2"/>
    <input name="_operationsmask3"/>
    <input name="_optionsmask3"/>
    <input name="_operationsmask4"/>
    <input name="_optionsmask4"/>
    <input name="_outputformatowner"/>
    <input name="_outputformattype"/>
    <input name="_capturedbir_keyvalue"/>
    <input name="_no_newtemplate"/>
    <input name="_payload"/>
    <input name="_no_templateuuid"/>

    <!-- Parameter given by the CTS by referring to the Expected Result Table. -->
    <input name="_expected_return_value"/>

    <!-- Parameter given by the CTS. -->
    <input name="_bspuuid"/>
    <input name="_dbuuid"/>

    <!-- Invocation of the primary activity of this assertion. -->
    <invoke activity="BioAPI_Createtemplate_And_BioSPI_Createtemplate"/>

    <!-- Bind activities to check the BioSPI function invoked by the framework under test -->
    <bind activity="SPI_CreateTemplate" function="SPI_CreateTemplate"/>
  </assertion>

  <!-- **************** -->
  <!-- Primary activity -->
  <!-- **************** -->
  <activity name="BioAPI_CreateTemplate_And_BioSPI_CreateTemplate">

    <!-- Attach -->
    <invoke activity="AttachWithSomeOptions" package="fb6ff5b0-7d5e-11de-8a39-0800200c9a66" break_on_break="true">
      <input name="Operationsmask1" var="_operationsmask1"/>
      <input name="Optionmask1" var="_optionsmask1"/>
      <input name="Operationsmask2" var="_operationsmask2"/>
      <input name="Optionmask2" var="_optionsmask2"/>
      <input name="Operationsmask3" var="_operationsmask3"/>
      <input name="Optionmask3" var="_optionsmask3"/>
      <input name="Operationsmask4" var="_operationsmask4"/>
      <input name="Optionmask4" var="_optionsmask4"/>
      <input name="Bspuuid" var="_bspuuid"/>
      <output name="Bsphandle" setvar="_bsphandle"/>
    </invoke>

    <!-- Capture -->
    <invoke activity="PrepareCapturedBIR"
            package="fb6ff5b0-7d5e-11de-8a39-0800200c9a66" break_on_break="true">
      <input name="Bsphandle" var="_bsphandle"/>
      <input name="Purpose" var="__BioAPI_PURPOSE_ENROLL"/>
      <input name="Outputformatowner" var="_outputformatowner"/>
      <input name="Outputformattype" var="_outputformattype"/>
      <output name="Form" setvar="_capturedbir_form"/>
      <output name="Birhandle" setvar="_capturedbir_birhandle"/>
    </invoke>

    <!-- If Test Number is 040206, 040207, 040208 or 040209, PrepareReferenceTemplate. -->
    <set name="_referencetemplate_form" value="0"/>
    <set name="_referencetemplate_birhandle" value=""/>
    <invoke activity="PrepareReferenceTemplate"
            package="fb6ff5b0-7d5e-11de-8a39-0800200c9a66" break_on_break="true">
      <only_if>
        <equal_to var1="_testnumber" value2="040206"/>
        <equal_to var1="_testnumber" value2="040207"/>
        <equal_to var1="_testnumber" value2="040208"/>
        <equal_to var1="_testnumber" value2="040209"/>
      </only_if>
      <input name="Bsphandle" var="_bsphandle"/>
      <input name="Outputformatowner" var="_outputformatowner"/>
      <input name="Outputformattype" var="_outputformattype"/>
      <output name="Form" setvar="_referencetemplate_form"/>
      <output name="Birhandle" setvar="_referencetemplate_birhandle"/>
    </invoke>

    <!-- Set an illegal value to the parameter. -->
    <add name="_bsphandle" value="1">
      <only_if><equal_to var1="_testnumber" value2="040202"/></only_if>
    </add>
    <add name="_referencetemplate_birhandle" value="1">
      <only_if>
        <equal_to var1="_testnumber" value2="040208"/>
        <equal_to var1="_testnumber" value2="040209"/>
      </only_if>
    </add>
    <set name="_capturedbir_form" value="0">
      <only_if><equal_to var1="_testnumber" value2="040203"/></only_if>
    </set>
    <add name="_capturedbir_birhandle" value="1">
      <only_if><equal_to var1="_testnumber" value2="040205"/></only_if>
    </add>

    <!-- Invoke the function BioAPI_CreateTemplate. -->
    <invoke function="BioAPI_CreateTemplate">
      <input name="BSPHandle" var="_bsphandle"/>
      <input name="CapturedBIR_Form" var="_capturedbir_form"/>
      <input name="CapturedBIR_BIRHandle" var="_capturedbir_birhandle"/>
      <input name="Referencetemplate_Form" var="_referencetemplate_form"/>
      <input name="Referencetemplate_BIRHandle" var="_referencetemplate_birhandle"/>
      <input name="OutputFormatOwner" var="_outputformatowner"/>
      <input name="OutputFormatType" var="_outputformattype"/>
      <input name="no_Newtemplate" var="_no_newtemplate"/>
      <input name="Payload" var="_payload"/>
      <input name="no_TemplateUUID" var="_no_templateuuid"/>
      <output name="NewTemplate" setvar="newtemplate"/>
      <output name="TemplateUUID" setvar="templateuuid"/>
      <return setvar="return"/>
    </invoke>

    <!-- Extract the error code from the error value. -->
    <invoke activity="ExtractErrorCode" package="fb6ff5b0-7d5e-11de-8a39-0800200c9a66" break_on_break="true">
      <output name="Returnvalue" setvar="return"/>
    </invoke>

    <!-- Assertion -->
    <assert_condition response_if_false="fail">
      <description>
        The function BioAPI_CreateTemplate has returned the expected return value.
      </description>
      <or>
        <!-- If the "_expected_return_value" parameter is 0xFFFFFFFF,
             then check the "return" parameter is not __BioAPI_OK. (error check only) -->
        <and>
          <equal_to var1="_expected_return_value" var2="_indeterminate_error"/>
          <not_equal_to var1="return" var2="__BioAPI_OK"/>
        </and>
        <and>
          <not_equal_to var1="_expected_return_value" var2="_indeterminate_error"/>
          <equal_to var1="return" var2="_expected_return_value"/>
        </and>
      </or>
    </assert_condition>

    <!-- If Test Number is 040202, make _bsphandle to an valid value. -->
    <subtract name="_bsphandle" value="1">
      <only_if><equal_to var1="_testnumber" value2="040202"/></only_if>
    </subtract>

    <!-- Invoke the functions Detach -->
    <invoke activity="BSPDetach" package="fb6ff5b0-7d5e-11de-8a39-0800200c9a66" break_on_break="true">
      <input name="Bsphandle" var="_bsphandle"/>
      <input name="Bspuuid" var="_bspuuid"/>
    </invoke>
  </activity>

  <!-- *************************************** -->
  <!-- Activity bound to BioSPI_CreateTemplate -->
  <!-- *************************************** -->
  <activity name="SPI_CreateTemplate">
    <input name="BSPHandle"/>
    <input name="CapturedBIR_Form"/>
    <input name="CapturedBIR_BIRHandle"/>
```

```
    <input name="Referencetemplate_Form"/>
    <input name="Referencetemplate_BIRHandle"/>
    <input name="OutputFormatOwner"/>
    <input name="OutputFormatType"/>
    <input name="no_Newtemplate"/>
    <input name="Payload"/>
    <input name="no_TemplateUUID"/>
    <output name="NewTemplate"/>
    <output name="TemplateUUID"/>
    <output name="return"/>

    <!-- check API=SPI -->
    <assert_condition response_if_false="fail">
      <and>
        <equal_to var1="BSPHandle" var2="_bsphandle"/>
        <equal_to var1="CapturedBIR_Form" var2="_capturedbir_form"/>
        <equal_to var1="CapturedBIR_BIRHandle" var2="_capturedbir_birhandle"/>
        <equal_to var1="Referencetemplate_Form" var2="_referencetemplate_form"/>
        <equal_to var1="Referencetemplate_BIRHandle" var2="_referencetemplate_birhandle"/>
        <equal_to var1="OutputFormatOwner" var2="_outputformatowner"/>
        <equal_to var1="OutputFormatType" var2="_outputformattype"/>
        <equal_to var1="no_Newtemplate" var2="_no_newtemplate"/>
        <equal_to var1="Payload" var2="_payload"/>
        <equal_to var1="no_TemplateUUID" var2="_no_templateuuid"/>
      </and>
    </assert_condition>
  </activity>
</package>
```

8.23 Assertion 4.3 – BioAPI_Process_And_BioSPI_Process

Description: This assertion calls BioAPI_Process with the input parameters described in Default Input Table and Test Condition Table, and checks if the framework under test returns BioAPI_OK or an error value in accordance with the description in Expected Result Table. The assertion is according to the following statement in the subclauses from BioAPI 2.0 standard document (ISO/IEC19784-1:2006), which are described in References in this subclause in this part of ISO/IEC 24709.

Excerpts:

Subclause 8.4.3

```
BioAPI_RETURN BioAPI BioAPI_Process
        (BioAPI_HANDLE                              BSPHandle,
        const BioAPI_INPUT_BIR                      *CapturedBIR,
        const BioAPI_BIR_BIOMETRIC_DATA_FORMAT      *OutputFormat,
        BioAPI_BIR_HANDLE                           *ProcessedBIR);
```

This function processes the intermediate data captured via a call to BioAPI_Capture for the purpose of either verification or identification. If the processing capability is supported by the attached BSP invocation, the BSP builds a "processed biometric sample" BIR; otherwise, ProcessedBIR is set to NULL, and this function returns BioAPIERR_BSP_FUNCTION_NOT_SUPPORTED.

This function results in the creation of a BIR by the BSP. The application can retrieve the BIR using the BIR handle through a call to BioAPI_GetBIRFromHandle, which also frees the handle, or can release the memory associated with the BIR handle only through a call to BioAPI_FreeBIRHandle.

A BioAPI_RETURN value indicates success or specifies a particular error condition. The value BioAPI_OK indicates success. All other values represent an error condition.

Subclause 9.3.4.3

```
BioAPI_RETURN BioAPI BioSPI_Process
        (BioAPI_HANDLE                              BSPHandle,
        const BioAPI_INPUT_BIR                      *CapturedBIR,
        const BioAPI_BIR_BIOMETRIC_DATA_FORMAT      *OutputFormat,
        BioAPI_BIR_HANDLE                           *ProcessedBIR);
```

References: 8.4.3, 9.3.4.3

Scenario:

a) The testing application does the followings:

1) Initialises the Framework under test, then installs, loads and attaches the testing BSP.
2) Captures a biometric sample to obtain a BIR handle.
3) Calls BioAPI_Process with the conditions given by the Default Input Table and by each row of the Test Condition Table.

b) The testing BSP does the followings:

1) Checks the parameters and the contents of the pointer variables as described in the column 'BioSPI function' in the Expected Result Table.
2) Sets the return value of the BioSPI_Process given by each row of the Test Condition Table, then returns to the Framework.

c) The testing application does the followings:

1) Checks the return value of the BioAPI_Process and if it is not as described in the Expected Result Table, then issues a fail conformity response.
2) Detaches, unloads and uninstalls the testing BSP, then terminates the Framework under test.

Parameters: The input parameters described below table are used for this assertion.

Table 70 — Default Input Table for BioAPI_Process_And_BioSPI_Process

Number	Input parameter name	Input parameter value (underscore:invalid)
1	BSPHandle	*Valid BSPHandle*
2	CapturedBIR_Form	2
3	CapturedBIR_BIRHandle	*Valid BIRHandle*
4	OutputFormatOwner	N/A
5	OutputFormatType	N/A
6	no_ProcessedBIR	false

Table 71 — Test Condition Table for BioAPI_Process_And_BioSPI_Process

Test number	Input parameter name	Input parameter value (underscore:invalid)	Supported options in BSP Schema		Return value (from BioSPI)
			OPERATIONS_MASK	OPTIONS_MASK	
040301	BSPHandle	*Valid BSPHandle*	*Process Capture*	-	*OK*
040302	BSPHandle	*Invalid BSPHandle*	*Process Capture*	-	-
040303	CapturedBIR_Form	0	*Process Capture*	-	*INVALID_INPUT_POINTER*
040304	CapturedBIR_BIRHandle	*Valid BIRHandle*	*Process Capture*	-	*OK*
040305	CapturedBIR_BIRHandle	*Invalid BIRHandle*	*Process Capture*	-	*INVALID_BIR_HANDLE*
040306	no_ProcessedBIR	true	*Process Capture*	-	*INVALID_OUTPUT_POINTER*
040307	BSPHandle	*Valid BSPHandle*	*Capture*	-	-

Table 72 — Expected Result Table for BioAPI_Process_And_BioSPI_Process

Test number	BioSPI function (BioAPI/BioSPI parameter check)	BioAPI function			Other conditions
		Return value	Output parameter name	Output parameter value	
040301	X	*OK*	-	-	-
040302	-	*indeterminate error*	-	-	-
040303	X	*indeterminate error*	-	-	-
040304	X	*OK*	-	-	-
040305	X	*INVALID_BIR_HANDLE*	-	-	-
040306	X	*indeterminate error*	-	-	-
040307	-	*indeterminate error*	-	-	-

Assertion language package

```
<?xml version="1.0" encoding="UTF-8"?>
<package name="c9a0a050-792d-11de-8a39-0800200c9a66">
  <author>ISO/IEC JTC1 SC37</author>
  <description>This package contains the assertion "BioAPI_Process_And_BioSPI_Process".</description>

  <!-- ************** -->
  <!-- Test assertion -->
  <!-- ************** -->
  <assertion name="BioAPI_Process_And_BioSPI_Process" model="frameworkTesting">
    <description>
      This assertion checks if BioAPI_Process is called with input parameters as described in Test Condition Table,
      the framework under test returns BioAPI_OK or error value in accordance with the description in Expected
      Result Table.
    </description>

    <!-- Parameter given by the CTS by referring to the Test Condition Table. -->
    <input name="_testnumber"/>
    <input name="_operationsmask1"/>
    <input name="_optionsmask1"/>
    <input name="_operationsmask2"/>
    <input name="_optionsmask2"/>
    <input name="_operationsmask3"/>
    <input name="_optionsmask3"/>
    <input name="_operationsmask4"/>
    <input name="_optionsmask4"/>
    <input name="_outputformatowner"/>
    <input name="_outputformattype"/>
    <input name="_no_processedbir"/>

    <!-- Parameter given by the CTS by referring to the Expected Result Table. -->
    <input name="_expected_return_value"/>

    <!-- Parameter given by the CTS. -->
    <input name="_bspuuid"/>

    <!-- Invocation of the primary activity of this assertion. -->
    <invoke activity="BioAPI_Process_And_BioSPI_Process"/>

    <!-- Bind activities to check the BioSPI function invoked by the framework under test -->
    <bind activity="SPI_Process" function="BioSPI_Process"/>
  </assertion>

  <!-- **************** -->
  <!-- Primary activity -->
  <!-- **************** -->
  <activity name="BioAPI_Process_And_BioSPI_Process">

    <!-- Attach -->
    <invoke activity="AttachWithSomeOptions" package="fb6ff5b0-7d5e-11de-8a39-0800200c9a66" break_on_break="true">
      <input name="Operationsmask1" var="_operationsmask1"/>
      <input name="Optionmask1" var="_optionsmask1"/>
      <input name="Operationsmask2" var="_operationsmask2"/>
      <input name="Optionmask2" var="_optionsmask2"/>
      <input name="Operationsmask3" var="_operationsmask3"/>
      <input name="Optionmask3" var="_optionsmask3"/>
      <input name="Operationsmask4" var="_operationsmask4"/>
      <input name="Optionmask4" var="_optionsmask4"/>
      <input name="Bspuuid" var="_bspuuid"/>
      <output name="Bsphandle" setvar="_bsphandle"/>
    </invoke>

    <!-- Capture -->
    <invoke activity="PrepareCapturedBIR"
            package="fb6ff5b0-7d5e-11de-8a39-0800200c9a66" break_on_break="true">
      <input name="Bsphandle" var="_bsphandle"/>
      <input name="Purpose" var="__BioAPI_PURPOSE_VERIFY"/>
      <input name="Outputformatowner" var="_outputformatowner"/>
      <input name="Outputformattype" var="_outputformattype"/>
      <output name="Form" setvar="_capturedbir_form"/>
      <output name="Birhandle" setvar="_capturedbir_birhandle"/>
    </invoke>

    <!-- Set an illegal value to the parameter. -->
    <add name="_bsphandle" value="1">
      <only_if><equal_to var1="_testnumber" value2="040302"/></only_if>
    </add>
    <set name="_capturedbir_form" value="0">
      <only_if><equal_to var1="_testnumber" value2="040303"/></only_if>
    </set>
    <add name="_capturedbir_birhandle" value="1">
      <only_if><equal_to var1="_testnumber" value2="040305"/></only_if>
    </add>
    <!-- Invoke the function BioAPI_Process. -->
    <invoke function="BioAPI_Process">
      <input name="BSPHandle" var="_bsphandle"/>
      <input name="CapturedBIR_Form" var="_capturedbir_form"/>
      <input name="CapturedBIR_BIRHandle" var="_capturedbir_birhandle"/>
      <input name="OutputFormatOwner" var="_outputformatowner"/>
      <input name="OutputFormatType" var="_outputformattype"/>
      <input name="no_ProcessedBIR" var="no_processedbir"/>
      <output name="ProcessedBIR" setvar="processedbir"/>
      <return setvar="return"/>
    </invoke>

    <!-- Extract the error code from the error value. -->
    <invoke activity="ExtractErrorCode" package="fb6ff5b0-7d5e-11de-8a39-0800200c9a66" break_on_break="true">
      <output name="Returnvalue" setvar="return"/>
    </invoke>

    <!-- Assertion -->
    <assert_condition response_if_false="fail">
      <description>
        The function BioAPI_Process has returned the expected return value.
      </description>
      <or>
        <!-- If the "_expected_return_value" parameter is 0xFFFFFFFF,
             then check the "return" parameter is not __BioAPI_OK. (error check only) -->
        <and>
          <equal_to var1="_expected_return_value" var2="_indeterminate_error"/>
          <not_equal_to var1="return" var2="__BioAPI_OK"/>
        </and>
        <and>
          <not_equal_to var1="_expected_return_value" var2="_indeterminate_error"/>
          <equal_to var1="return" var2="_expected_return_value"/>
        </and>
      </or>
    </assert_condition>

    <!-- If Test Number is 040302, make _bsphandle to an valid value. -->
    <subtract name="_bsphandle" value="1">
      <only_if><equal_to var1="_testnumber" value2="040302"/></only_if>
    </subtract>

    <!-- Invoke the functions Detach -->
    <invoke activity="BSPDetach" package="fb6ff5b0-7d5e-11de-8a39-0800200c9a66" break_on_break="true">
      <input name="Bsphandle" var="_bsphandle"/>
      <input name="Bspuuid" var="_bspuuid"/>
    </invoke>
  </activity>

  <!-- ******************************** -->
  <!-- Activity bound to BioSPI_Process -->
  <!-- ******************************** -->
  <activity name="SPI_Process">
    <input name="BSPHandle"/>
    <input name="CapturedBIR_Form"/>
    <input name="CapturedBIR_BIRHandle"/>
    <input name="OutputFormatOwner"/>
    <input name="OutputFormatType"/>
    <input name="no_ProcessedBIR"/>
    <output name="ProcessedBIR"/>
    <output name="return"/>

    <!-- check API=SPI -->
    <assert_condition response_if_false="fail">
      <and>
        <equal_to var1="BSPHandle" var2="_bsphandle"/>
        <equal_to var1="CapturedBIR_Form" var2="_capturedbir_form"/>
        <equal_to var1="CapturedBIR_BIRHandle" var2="_capturedbir_birhandle"/>
        <equal_to var1="OutputFormatOwner" var2="_outputformatowner"/>
        <equal_to var1="OutputFormatType" var2="_outputformattype"/>
        <equal_to var1="no_ProcessedBIR" var2="_no_processedbir"/>
      </and>
    </assert_condition>
  </activity>
</package>
```

8.24 Assertion 4.4 – BioAPI_ProcessWithAuxBIR_And_BioSPI_ProcessWithAuxBIR

Description: This assertion calls BioAPI_ProcessWithAuxBIR with the input parameters described in Default Input Table and Test Condition Table, and checks if the framework under test returns BioAPI_OK or an error value in accordance with the description in Expected Result Table. The assertion is according to the following statement in the subclauses from BioAPI 2.0 standard document (ISO/IEC19784-1:2006), which are described in References in this subclause in this part of ISO/IEC 24709.

Excerpts:

Subclause 8.4.4

```
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_BIR_HANDLE                             *ProcessedBIR);
```

This function processes the intermediate data previously captured via a call to BioAPI_Capture in conjunction with auxiliary data, creating processed biometric samples for the purpose of subsequent verification or identification. It enables implementations that require the input of auxiliary data to the process operation.

NOTE This capability may be used to support biometric match-on-card (MOC). See clause C.8 for a description of BioAPI use within the overall MOC process.

If the processing with auxiliary data capability is supported by the attached BSP invocation, the BSP builds a "processed biometric sample" BIR, otherwise, ProcessedBIR is set to NULL, and this function returns BioAPIERR_BSP_FUNCTION_NOT_SUPPORTED.

This function results in the creation of a BIR by the BSP. The application can retrieve the BIR using the BIR handle through a call to BioAPI_GetBIRFromHandle, which also frees the handle, or can release the memory associated with the BIR handle only through a call to BioAPI_FreeBIRHandle.

A BioAPI_RETURN value indicates success or specifies a particular error condition. The value BioAPI_OK indicates success. All other values represent an error condition.

Subclause 9.3.4.4

```
BioSPI_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_BIR_HANDLE                           *ProcessedBIR);
```

References: 8.4.4, 9.3.4.4

Scenario:

a) The testing application does the followings:

1) Initialises the Framework under test, then installs, loads and attaches the testing BSP.
2) Captures a biometric sample to obtain a BIR handle.
3) Opens the database, and takes out an auxiliary data from the database.
4) Calls BioAPI_ProcessWithAuxBIR with the conditions given by the Default Input Table and by each row of the Test Condition Table.

b) The testing BSP does the followings:

1) Checks the parameters and the contents of the pointer variables as described in the column 'BioSPI function' in the Expected Result Table.
2) Sets the return value of the BioSPI_ProcessWithAuxBIR given by each row of the Test Condition Table, then returns to the Framework.

c) The testing application does the followings:

1) Checks the return value of the BioAPI_ProcessWithAuxBIR and if it is not as described in the Expected Result Table, then issues a fail conformity response.
2) Closes the database.
3) Detaches, unloads and uninstalls the testing BSP, then terminates the Framework under test.

Parameters: The input parameters described below table are used for this assertion.

Table 73 — Default Input Table for BioAPI_ProcessWithAuxBIR_And_BioSPI_ProcessWithAuxBIR

Number	Input parameter name	Input parameter value (underscore:invalid)
1	BSPHandle	*Valid BSPHandle*
2	CapturedBIR_Form	2
3	CapturedBIR_BIRHandle	*Valid BIRHandle*
4	AuxiliaryData_Form	1
5	AuxiliaryData_DBHandle	*Valid DBHandle*
6	AuxiliaryData_KeyValue	*Valid KeyValue*
7	OutputFormatOwner	N/A
8	OutputFormatType	N/A
9	no_ProcessedBIR	false

Table 74 — Test Condition Table for BioAPI_ProcessWithAuxBIR_And_BioSPI_ProcessWithAuxBIR

Test number	Input parameter name	Input parameter value (underscore:invalid)	Supported options in BSP Schema		Return value (from BioSPI)
			OPERATIONS_MASK	OPTIONS_MASK	
040401	BSPHandle	*Valid BSPHandle*	*ProcessWithAuxBIR Capture*	-	*OK*
040402	BSPHandle	Invalid BSPHandle	*ProcessWithAuxBIR Capture*	-	-
040403	CapturedBIR_Form	0	*ProcessWithAuxBIR Capture*	-	*INVALID_INPUT_POINTER*
040404	CapturedBIR_BIRHandle	*Valid BIRHandle*	*ProcessWithAuxBIR Capture*	-	*OK*
040405	CapturedBIR_BIRHandle	*Invalid BIRHandle*	*ProcessWithAuxBIR Capture*	-	*INVALID_BIR_HANDLE*
040406	AuxiliaryData_Form	0	*ProcessWithAuxBIR Capture*	-	*INVALID_INPUT_POINTER*
040407	AuxiliaryData_DBHandle	*Valid DBHandle*	*ProcessWithAuxBIR Capture*	-	*OK*
040408	AuxiliaryData_DBHandle	*Invalid DBHandle*	*ProcessWithAuxBIR Capture*	-	*INVALID_BIR_HANDLE*
040409	no_ProcessedBIR	true	*ProcessWithAuxBIR Capture*	-	*INVALID_OUTPUT_POINTER*
040410	BSPHandle	*Valid BSPHandle*	*Capture*	-	-

Table 75 — Expected Result Table for BioAPI_ProcessWithAuxBIR_And_BioSPI_ProcessWithAuxBIR

Test number	BioSPI function (BioAPI/BioSPI parameter check)	BioAPI function			Other conditions
		Return value	Output parameter name	Output parameter value	
040401	X	*OK*	-	-	-
040402	-	*indeterminate error*	-	-	-
040403	X	*indeterminate error*	-	-	-
040404	X	*OK*	-	-	-
040405	X	*INVALID_BIR_HANDLE*	-	-	-
040406	X	*indeterminate error*	-	-	-
040407	X	*OK*	-	-	-
040408	X	*INVALID_BIR_HANDLE*	-	-	-
040409	X	*indeterminate error*	-	-	-
040410	-	*FUNCTION_NOT_SUPPORTED*	-	-	-

Assertion language package

```
<?xml version="1.0" encoding="UTF-8"?>
<package name="eab95fc0-792d-11de-8a39-0800200c9a66">
  <author>ISO/IEC JTC1 SC37</author>
  <description>
    This package contains the assertion "BioAPI_ProcessWithAuxBIR_And_BioSPI_ProcessWithAuxBIR".
  </description>

  <!-- ************** -->
  <!-- Test assertion -->
  <!-- ************** -->
  <assertion name="BioAPI_ProcessWithAuxBIR_And_BioSPI_ProcessWithAuxBIR" model="frameworkTesting">
    <description>
```

```
      This assertion checks if BioAPI_ProcessWithAuxBIR is called with input parameters as described in Test
      Condition Table, the framework under test returns BioAPI_OK or error value in accordance with the description
      in Expected Result Table.
    </description>

    <!-- Parameter given by the CTS by referring to the Test Condition Table. -->
    <input name="_testnumber"/>
    <input name="_operationsmask1"/>
    <input name="_optionsmask1"/>
    <input name="_operationsmask2"/>
    <input name="_optionsmask2"/>
    <input name="_operationsmask3"/>
    <input name="_optionsmask3"/>
    <input name="_operationsmask4"/>
    <input name="_optionsmask4"/>
    <input name="_outputformatowner"/>
    <input name="_outputformattype"/>
    <input name="_no_processedbir"/>

    <!-- Parameter given by the CTS by referring to the Expected Result Table. -->
    <input name="_expected_return_value"/>

    <!-- Parameter given by the CTS. -->
    <input name="_bspuuid"/>
    <input name="_dbuuid"/>
    <input name="_keyvalue"/>

    <!-- Invocation of the primary activity of this assertion. -->
    <invoke activity="BioAPI_ProcessWithAuxBIR_And_BioSPI_ProcessWithAuxBIR"/>

    <!-- Bind activities to check the BioSPI function invoked by the framework under test -->
    <bind activity="SPI_ProcessWithAuxBIR" function="BioSPI_ProcessWithAuxBIR"/>
  </assertion>

  <!-- **************** -->
  <!-- Primary activity -->
  <!-- **************** -->
  <activity name="BioAPI_ProcessWithAuxBIR_And_BioSPI_ProcessWithAuxBIR">

    <!-- Attach -->
    <invoke activity="AttachWithSomeOptions" package="fb6ff5b0-7d5e-11de-8a39-0800200c9a66" break_on_break="true">
      <input name="Operationsmask1" var="_operationsmask1"/>
      <input name="Optionmask1" var="_optionsmask1"/>
      <input name="Operationsmask2" var="_operationsmask2"/>
      <input name="Optionmask2" var="_optionsmask2"/>
      <input name="Operationsmask3" var="_operationsmask3"/>
      <input name="Optionmask3" var="_optionsmask3"/>
      <input name="Operationsmask4" var="_operationsmask4"/>
      <input name="Optionmask4" var="_optionsmask4"/>
      <input name="Bspuuid" var="_bspuuid"/>
      <output name="Bsphandle" setvar="_bsphandle"/>
    </invoke>

    <!-- Capture -->
    <invoke activity="PrepareCapturedBIR"
            package="fb6ff5b0-7d5e-11de-8a39-0800200c9a66" break_on_break="true">
      <input name="Bsphandle" var="_bsphandle"/>
      <input name="Purpose" var="__BioAPI_PURPOSE_VERIFY"/>
      <input name="Outputformatowner" var="_outputformatowner"/>
      <input name="Outputformattype" var="_outputformattype"/>
      <output name="Form" setvar="_capturedbir_form"/>
      <output name="Birhandle" setvar="_capturedbir_birhandle"/>
    </invoke>

    <!-- AuxiliaryData -->
    <invoke activity="MakeAuxiliaryData" break_on_break="true"/>

    <!-- Set an illegal value to the parameter. -->
    <add name="_bsphandle" value="1">
      <only_if><equal_to var1="_testnumber" value2="040402"/></only_if>
    </add>
    <set name="_capturedbir_form" value="0">
      <only_if><equal_to var1="_testnumber" value2="040403"/></only_if>
    </set>
    <add name="_capturedbir_birhandle" value="1">
      <only_if><equal_to var1="_testnumber" value2="040405"/></only_if>
    </add>
    <set name="_auxiliarydata_form" value="0">
      <only_if><equal_to var1="_testnumber" value2="040406"/></only_if>
    </set>
    <add name="_auxiliarydata_dbhandle" value="1">
      <only_if><equal_to var1="_testnumber" value2="040408"/></only_if>
    </add>

    <!-- Invoke the function BioAPI_ProcessWithAuxBIR. -->
    <invoke function="BioAPI_ProcessWithAuxBIR">
      <input name="BSPHandle" var="_bsphandle"/>
      <input name="CapturedBIR_Form" var="_capturedbir_form"/>
      <input name="CapturedBIR_BIRHandle" var="_capturedbir_birhandle"/>
      <input name="AuxiliaryData_Form" var="_auxiliarydata_form"/>
      <input name="AuxiliaryData_DBHandle" var="_auxiliarydata_dbhandle"/>
      <input name="AuxiliaryData_KeyValue" var="_auxiliarydata_keyvalue"/>
      <input name="OutputFormatOwner" var="_outputformatowner"/>
      <input name="OutputFormatType" var="_outputformattype"/>
      <input name="no_ProcessedBIR" var="no_processedbir"/>
      <output name="ProcessedBIR" setvar="processedbir"/>
      <return setvar="return"/>
    </invoke>

    <!-- Extract the error code from the error value. -->
    <invoke activity="ExtractErrorCode" package="fb6ff5b0-7d5e-11de-8a39-0800200c9a66" break_on_break="true">
      <output name="Returnvalue" setvar="return"/>
    </invoke>

    <!-- Assertion -->
    <assert_condition response_if_false="fail">
      <description>
        The function BioAPI_ProcessWithAuxBIR has returned the expected return value.
      </description>
      <or>
        <!-- If the "_expected_return_value" parameter is 0xFFFFFFFF,
             then check the "return" parameter is not __BioAPI_OK. (error check only) -->
        <and>
          <equal_to var1="_expected_return_value" var2="_indeterminate_error"/>
          <not_equal_to var1="return" var2="__BioAPI_OK"/>
        </and>
        <and>
          <not_equal_to var1="_expected_return_value" var2="_indeterminate_error"/>
          <equal_to var1="return" var2="_expected_return_value"/>
        </and>
      </or>
    </assert_condition>

    <!-- If Test Number is 040402, make _bsphandle to an valid value. -->
    <subtract name="_bsphandle" value="1">
      <only_if><equal_to var1="_testnumber" value2="040402"/></only_if>
    </subtract>

    <!-- Invoke the activity DB Close the framework under test -->
    <invoke activity="DbClose" package="fb6ff5b0-7d5e-11de-8a39-0800200c9a66" break_on_break="true">
      <input name="Bsphandle" var="_bsphandle"/>
      <input name="Dbhandle" var="_dbhandle"/>
    </invoke>

    <!-- Invoke the functions Detach -->
    <invoke activity="BSPDetach" package="fb6ff5b0-7d5e-11de-8a39-0800200c9a66" break_on_break="true">
      <input name="Bsphandle" var="_bsphandle"/>
      <input name="Bspuuid" var="_bspuuid"/>
    </invoke>
  </activity>

  <!-- ************************************ -->
  <!-- Activity to Make auxiliary data.(DB) -->
  <!-- ************************************ -->
  <activity name="MakeAuxiliaryData">
    <!-- Invoke the activity DB Open the framework under test -->
    <invoke activity="DbOpen" package="fb6ff5b0-7d5e-11de-8a39-0800200c9a66" break_on_break="true">
      <input name="Bsphandle" var="_bsphandle"/>
      <input name="Dbuuid" var="_dbuuid"/>
      <output name="Dbhandle" setvar="_dbhandle"/>
      <output name="Markerhandle" setvar="_markerhandle"/>
    </invoke>
    <set name="auxiliarydata_form" value="1"/>
    <set name="auxiliarydata_dbhandle" value="_dbhandle"/>
    <set name="auxiliarydata_keyvalue" value="_keyvalue"/>
  </activity>

  <!-- ****************************************** -->
  <!-- Activity bound to BioSPI_ProcessWithAuxBIR -->
  <!-- ****************************************** -->
  <activity name="SPI_ProcessWithAuxBIR">
    <input name="BSPHandle"/>
    <input name="CapturedBIR_Form"/>
    <input name="CapturedBIR_BIRHandle"/>
    <input name="AuxiliaryData_Form"/>
    <input name="AuxiliaryData_DBHandle"/>
    <input name="AuxiliaryData_KeyValue"/>
    <input name="OutputFormatOwner"/>
    <input name="OutputFormatType"/>
    <input name="no_ProcessedBIR"/>
    <output name="ProcessedBIR"/>
    <output name="return"/>
```

```
    <!-- check API=SPI -->
    <assert_condition response_if_false="fail">
      <and>
        <equal_to var1="BSPHandle" var2="_bsphandle"/>
        <equal_to var1="CapturedBIR_Form" var2="_capturedbir_form"/>
        <equal_to var1="CapturedBIR_BIRHandle" var2="_capturedbir_birhandle"/>
        <equal_to var1="AuxiliaryData_Form" var2="_auxiliarydata_form"/>
        <equal_to var1="AuxiliaryData_KeyValue" var2="_auxiliarydata_keyvalue"/>
        <equal_to var1="BSPHandle" var2="_bsphandle"/>
        <equal_to var1="OutputFormatOwner" var2="_outputformatowner"/>
        <equal_to var1="OutputFormatType" var2="_outputformattype"/>
        <equal_to var1="no_ProcessedBIR" var2="_no_processedbir"/>
      </and>
    </assert_condition>
  </activity>
</package>
```

8.25 Assertion 4.5 – BioAPI_VerifyMatch_And_BioSPI_VerifyMatch

Description: This assertion calls BioAPI_VerifyMatch with the input parameters described in Default Input Table and Test Condition Table, and checks if the framework under test returns BioAPI_OK or an error value in accordance with the description in Expected Result Table. The assertion is according to the following statement in the subclauses from BioAPI 2.0 standard document (ISO/IEC19784-1:2006), which are described in References in this subclause in this part of ISO/IEC 24709.

Excerpts:

Subclause 8.4.5

```
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);
```

This function performs a verification (1-to-1) match between two BIRs: the ProcessedBIR and the ReferenceTemplate. The ProcessedBIR is the "processed" BIR constructed specifically for this verification. The ReferenceTemplate was created at enrollment.

The application shall request a maximum FMR value criterion (threshold) for a successful match. The Boolean Result indicates whether verification was successful or not, and the FMRAchieved is a FMR value (score) indicating how closely the BIRs actually matched.

NOTE See clause C.4 for information on the use of the FMR concept for normalized scoring and thresholding.

By setting the AdaptedBIR pointer to an address other than NULL, the application can request that a BIR be constructed by adapting the ReferenceTemplate using the ProcessedBIR. A new handle is returned to the AdaptedBIR. If the match is successful, an attempt may be made to adapt the ReferenceTemplate with information taken from the ProcessedBIR. (Not all BSPs perform adaptation). The resulting AdaptedBIR should now be considered an optimal enrollment template, and be saved in the BIR database. (It is up to the application whether it uses or discards this data.) It is important to note that adaptation may not occur in all cases. In the event of an adaptation, this function stores the handle to the new BIR in the memory pointed to by the AdaptedBIR parameter.

If a Payload is associated with the ReferenceTemplate, the Payload may be returned upon successful verification if the FMRAchieved is sufficiently stringent; this is controlled by the policy of the BSP and specified in its schema.

NOTE 1 Not all BSPs support return of payloads.

NOTE 2 See clauses A.4.6.2.6 and C.5 for additional information regarding use of payloads.

The memory block returned by the BioAPI function call shall be freed by the application as soon as it is no longer needed using BioAPI_Free (see clause 8.7.2). If an adapted BIR is returned, its handle can be released through a call to BioAPI_FreeBIRHandle.

A BioAPI_RETURN value indicates success or specifies a particular error condition. The value BioAPI_OK indicates success. All other values represent an error condition.

Subclause 9.3.4.5

```
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);
```

References: 8.4.5, 9.3.4.5

Scenario:

a) The testing application does the followings:

1) Initialises the Framework under test, then installs, loads and attaches the testing BSP.

2) Enrolls to obtain a BIR handle.

3) Captures a biometric sample and processes it.

4) Calls BioAPI_VerifyMatch with the conditions given by the Default Input Table and by each row of the Test Condition Table.

b) The testing BSP does the followings:

1) Checks the parameters and the contents of the pointer variables as described in the column 'BioSPI function' in the Expected Result Table.

2) Sets the return value of the BioSPI_VerifyMatch given by each row of the Test Condition Table, then returns to the Framework.

c) The testing application does the followings:

1) Checks the return value of the BioAPI_VerifyMatch and if it is not as described in the Expected Result Table, then issues a fail conformity response.

2) Detaches, unloads and uninstalls the testing BSP, then terminates the Framework under test.

Parameters: The input parameters described below table are used for this assertion.

Table 76 — Default Input Table for BioAPI_VerifyMatch_And_BioSPI_VerifyMatch

Number	Input parameter name	Input parameter value (underscore:invalid)
1	BSPHandle	*Valid BSPHandle*
2	MaxFMRRequested	__BioAPI_NOT_SET
3	ProcessedBIR_Form	2
4	ProcessedBIR_BIRHandle	*Valid BIRHandle*
5	ReferenceTemplate_Form	2
6	ReferenceTemplate_BIRHandle	*Valid BIRHandle*
7	no_AdaptedBIR	true
8	no_Result	false
9	no_FMRAchieved	false
10	no_Payload	false

Table 77 — Test Condition Table for BioAPI_VerifyMatch_And_BioSPI_VerifyMatch

Test number	Input parameter name	Input parameter value (underscore:invalid)	Supported options in BSP Schema		Return value (from BioSPI)
			OPERATIONS_MASK	OPTIONS_MASK	
040501	BSPHandle	*Valid BSPHandle*	*VerifyMatch Capture Process Enroll*	-	*OK*
040502	BSPHandle	<u>*Invalid BSPHandle*</u>	*VerifyMatch Capture Process Enroll*	-	-
040503	MaxFMRRequested	0	*VerifyMatch Capture Process Enroll*	-	*OK*
040504	MaxFMRRequested	50	*VerifyMatch Capture Process Enroll*	-	*OK*
040505	MaxFMRRequested	100	*VerifyMatch Capture Process Enroll*	-	*OK*
040506	ProcessedBIR_Form	<u>0</u>	*VerifyMatch Capture Process Enroll*	-	*INVALID_INPUT_POINTER*
040507	ProcessedBIR_BIRHandle	*Valid BIRHandle*	*VerifyMatch Capture Process Enroll*	-	*OK*
040508	ProcessedBIR_BIRHandle	<u>*Invalid BIRHandle*</u>	*VerifyMatch Capture Process Enroll*	-	*INVALID_BIR_HANDLE*
040509	ReferenceTemplate_Form	<u>0</u>	*VerifyMatch Capture Process Enroll*	-	*INVALID_INPUT_POINTER*
040510	ReferenceTemplate_BIRHandle	*Valid BIRHandle*	*VerifyMatch Capture Process Enroll*	-	*OK*
040511	ReferenceTemplate_BIRHandle	<u>*Invalid BIRHandle*</u>	*VerifyMatch Capture Process Enroll*	-	*INVALID_BIR_HANDLE*
040512	no_AdaptedBIR	false	*VerifyMatch Capture Process Enroll*	-	*OK*
040513	no_Result	<u>true</u>	*VerifyMatch Capture Process Enroll*	-	*INVALID_OUTPUT_POINTER*
040514	no_FMRAchieved	<u>true</u>	*VerifyMatch Capture Process Enroll*	-	*FUNCTION_NOT_SUPPORTED*
040515	no_Payload	true	*VerifyMatch Capture Process Enroll*	-	*OK*
040516	BSPHandle	*Valid BSPHandle*	-	-	*FUNCTION_NOT_SUPPORTED*

Table 78 — Expected Result Table for BioAPI_VerifyMatch_And_BioSPI_VerifyMatch

Test number	BioSPI function (BioAPI/BioSPI parameter check)	BioAPI function			Other conditions
		Return value	Output parameter name	Output parameter value	
040501	X	*OK*	-	-	-
040502	-	*indeterminate error*	-	-	-
040503	X	*OK*	-	-	-
040504	X	*OK*	-	-	-
040505	X	*OK*	-	-	-
040506	X	*indeterminate error*	-	-	-
040507	X	*OK*	-	-	-
040508	X	*INVALID_BIR_HANDLE*	-	-	-
040509	X	*indeterminate error*	-	-	-
040510	X	*OK*	-	-	-
040511	X	*INVALID_BIR_HANDLE*	-	-	-
040512	X	*OK*	-	-	-
040513	X	*indeterminate error*	-	-	-
040514	X	*indeterminate error*	-	-	-
040515	X	*OK*	-	-	-
040516	-	*indeterminate error*	-	-	-

Assertion language package

```
<?xml version="1.0" encoding="UTF-8"?>
<package name="f66559a0-792d-11de-8a39-0800200c9a66">
  <author>ISO/IEC JTC1 SC37</author>
  <description>This package contains the assertion "BioAPI_VerifyMatch_And_BioSPI_VerifyMatch".</description>

  <!-- ************** -->
  <!-- Test assertion -->
  <!-- ************** -->
  <assertion name="BioAPI_VerifyMatch_And_BioSPI_VerifyMatch" model="frameworkTesting">
    <description>
      This assertion checks if BioAPI_VerifyMatch is called with input parameters as described in Test Condition
      Table, the framework under test returns BioAPI_OK or error value in accordance with the description in
      Expected Result Table.
    </description>

    <!-- Parameter given by the CTS by referring to the Test Condition Table. -->
    <input name="_testnumber"/>
    <input name="_operationsmask1"/>
    <input name="_optionsmask1"/>
    <input name="_operationsmask2"/>
    <input name="_optionsmask2"/>
    <input name="_operationsmask3"/>
    <input name="_optionsmask3"/>
    <input name="_operationsmask4"/>
    <input name="_optionsmask4"/>
    <input name="_maxfmrrequested"/>
    <input name="_no_adaptedbir"/>
    <input name="_no_result"/>
    <input name="_no_fmrachieved"/>
    <input name="_no_payload"/>

    <!-- Parameter given by the CTS by referring to the Expected Result Table. -->
    <input name="_expected_return_value"/>

    <!-- Parameter given by the CTS. -->
    <input name="_bspuuid"/>
    <input name="_dbuuid"/>
    <input name="_outputformatowner"/>
    <input name="_outputformattype"/>

    <!-- Invocation of the primary activity of this assertion. -->
    <invoke activity="BioAPI_VerifyMatch_And_BioSPI_VerifyMatch"/>

    <!-- Bind activities to check the BioSPI function invoked by the framework under test -->
    <bind activity="SPI_VerifyMatch" function="BioSPI_VerifyMatch"/>
  </assertion>

  <!-- **************** -->
  <!-- Primary activity -->
  <!-- **************** -->
  <activity name="BioAPI_VerifyMatch_And_BioSPI_VerifyMatch">

    <!-- Attach -->
    <invoke activity="AttachWithSomeOptions" package="fb6ff5b0-7d5e-11de-8a39-0800200c9a66" break_on_break="true">
      <input name="Operationsmask1" var="_operationsmask1"/>
      <input name="Optionmask1" var="_optionsmask1"/>
      <input name="Operationsmask2" var="_operationsmask2"/>
      <input name="Optionmask2" var="_optionsmask2"/>
      <input name="Operationsmask3" var="_operationsmask3"/>
      <input name="Optionmask3" var="_optionsmask3"/>
      <input name="Operationsmask4" var="_operationsmask4"/>
      <input name="Optionmask4" var="_optionsmask4"/>
      <input name="Bspuuid" var="_bspuuid"/>
      <output name="Bsphandle" setvar="_bsphandle"/>
    </invoke>

    <!-- PrepareReferenceTemplate. -->
    <invoke activity="PrepareReferenceTemplate"
            package="fb6ff5b0-7d5e-11de-8a39-0800200c9a66" break_on_break="true">
      <input name="Bsphandle" var="_bsphandle"/>
      <input name="Outputformatowner" var="_outputformatowner"/>
      <input name="Outputformattype" var="_outputformattype"/>
      <output name="Form" setvar="_referencetemplate_form"/>
      <output name="Birhandle" setvar="_referencetemplate_birhandle"/>
    </invoke>

    <!-- Capture and Process -->
    <invoke activity="PrepareProcessedBIR"
            package="fb6ff5b0-7d5e-11de-8a39-0800200c9a66" break_on_break="true">
      <input name="Bsphandle" var="_bsphandle"/>
      <input name="Purpose" var="__BioAPI_PURPOSE_VERIFY"/>
      <input name="Outputformatowner" var="_outputformatowner"/>
      <input name="Outputformattype" var="_outputformattype"/>
      <output name="Form" setvar="_processedbir_form"/>
      <output name="Birhandle" setvar="_processedbir_birhandle"/>
    </invoke>

    <!-- Set an illegal value to the parameter. -->
    <add name="_bsphandle" value="1">
      <only_if><equal_to var1="_testnumber" value2="040502"/></only_if>
    </add>
    <add name="_referencetemplate_birhandle" value="1">
      <only_if><equal_to var1="_testnumber" value2="040511"/></only_if>
    </add>
    <set name="_referencetemplate_form" value="0">
      <only_if><equal_to var1="_testnumber" value2="040509"/></only_if>
    </set>
    <add name="_processedbir_birhandle" value="1">
      <only_if><equal_to var1="_testnumber" value2="040508"/></only_if>
    </add>
    <set name="_processedbir_form" value="0">
      <only_if><equal_to var1="_testnumber" value2="040506"/></only_if>
    </set>

    <!-- Invoke the function BioAPI_VerifyMatch. -->
    <invoke function="BioAPI_VerifyMatch">
      <input name="BSPHandle" var="_bsphandle"/>
      <input name="MaxFMRRequested" var="_maxfmrrequested"/>
      <input name="ProcessedBIR_Form" var="_processedbir_form"/>
      <input name="ProcessedBIR_BIRHandle" var="_processedbir_birhandle"/>
      <input name="ReferenceTemplate_Form" var="_referencetemplate_form"/>
      <input name="ReferenceTemplate_BIRHandle" var="_referencetemplate_birhandle"/>
      <input name="no_AdaptedBIR" var="_no_adaptedbir"/>
      <input name="no_Result" var="_no_result"/>
      <input name="no_FMRAchieved" var="_no_fmrachieved"/>
      <input name="no_Payload" var="_no_payload"/>
      <output name="AdaptedBIR" setvar="adaptedbir"/>
      <output name="Result" setvar="result"/>
      <output name="FMRAchieved" setvar="fmrachieved"/>
      <output name="Payload" setvar="payload"/>
      <return setvar="return"/>
    </invoke>

    <!-- Extract the error code from the error value. -->
    <invoke activity="ExtractErrorCode" package="fb6ff5b0-7d5e-11de-8a39-0800200c9a66" break_on_break="true">
      <output name="Returnvalue" setvar="return"/>
    </invoke>

    <!-- Assertion -->
    <assert_condition response_if_false="fail">
      <description>
        The function BioAPI_VerifyMatch has returned the expected return value.
      </description>
      <or>
        <!-- If the "_expected_return_value" parameter is 0xFFFFFFFF,
             then check the "return" parameter is not __BioAPI_OK. (error check only) -->
        <and>
          <equal_to var1="_expected_return_value" var2="_indeterminate_error"/>
          <not_equal_to var1="return" var2="__BioAPI_OK"/>
        </and>
        <and>
          <not_equal_to var1="_expected_return_value" var2="_indeterminate_error"/>
          <equal_to var1="return" var2="_expected_return_value"/>
        </and>
      </or>
    </assert_condition>

    <!-- If Test Number is 040502, make _bsphandle to an valid value. -->
    <subtract name="_bsphandle" value="1">
      <only_if><equal_to var1="_testnumber" value2="040502"/></only_if>
    </subtract>

    <!-- Invoke the functions Detach -->
    <invoke activity="BSPDetach" package="fb6ff5b0-7d5e-11de-8a39-0800200c9a66" break_on_break="true">
      <input name="Bsphandle" var="_bsphandle"/>
      <input name="Bspuuid" var="_bspuuid"/>
    </invoke>
  </activity>

  <!-- ************************************ -->
  <!-- Activity bound to BioSPI_VerifyMatch -->
  <!-- ************************************ -->
  <activity name="SPI_VerifyMatch">
    <input name="BSPHandle"/>
    <input name="MaxFMRRequested"/>
    <input name="ProcessedBIR_Form"/>
    <input name="ProcessedBIR_BIRHandle"/>
    <input name="ReferenceTemplate_Form"/>
    <input name="ReferenceTemplate_BIRHandle"/>
    <input name="no_AdaptedBIR"/>
    <input name="no_Result"/>
    <input name="no_FMRAchieved"/>
    <input name="no_Payload"/>
    <output name="AdaptedBIR"/>
    <output name="Result"/>
    <output name="FMRAchieved"/>
    <output name="Payload"/>
    <output name="return"/>

    <!-- check API=SPI -->
    <assert_condition response_if_false="fail">
      <and>
        <equal_to var1="BSPHandle" var2="_bsphandle"/>
        <equal_to var1="MaxFMRRequested" var2="_maxfmrrequested"/>
        <equal_to var1="ProcessedBIR_Form" var2="_processedbir_form"/>
        <equal_to var1="ProcessedBIR_BIRHandle" var2="_processedbir_birhandle"/>
        <equal_to var1="ReferenceTemplate_Form" var2="_referencetemplate_form"/>
        <equal_to var1="ReferenceTemplate_BIRHandle" var2="_referencetemplate_birhandle"/>
        <equal_to var1="no_AdaptedBIR" var2="_no_adaptedbir"/>
        <equal_to var1="no_Result" var2="_no_result"/>
        <equal_to var1="no_FMRAchieved" var2="_no_fmrachieved"/>
        <equal_to var1="no_Payload" var2="_no_payload"/>
      </and>
    </assert_condition>
  </activity>
</package>
```

8.26 Assertion 4.6 – BioAPI_IdentifyMatch_And_BioSPI_IdentifyMatch

Description: This assertion calls BioAPI_IdentifyMatch with the input parameters described in Default Input Table and Test Condition Table, and checks if the framework under test returns BioAPI_OK or an error value in accordance with the description in Expected Result Table. The assertion is according to the following statement in the subclauses from BioAPI 2.0 standard document (ISO/IEC19784-1:2006), which are described in References in this subclause in this part of ISO/IEC 24709.

Excerpts:

Subclause 8.4.6

```
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);
```

This function performs an identification (1-to-many) match between a ProcessedBIR and a set of reference BIRs. The ProcessedBIR is the "processed" BIR captured specifically for this identification. The population that the match takes place against can be presented in one of two ways:

a) In a BIR database identified by an open database handle;
b) Input in an array of BIRs;

NOTE When using a BSP-controlled BIR database, this database must previously have been opened using BioAPI_DbOpen.

There is an option to use an array of BIRs, which can be specified in BioAPI_IDENTIFY _POPULATION_TYPE in the BioAPI_IDENTIFY_POPULATION structure. If it is specified as BioAPI_PRESET_ARRAY_TYPE (3), the array of BIRs which had been previously set in the BioAPI_PresetIdentifyPopulation call will be used. The preset array of BIRs will be freed internally by the BSP when BioAPI_BSPDetach is called.

The function performs the following actions (in order):

a) determines the set of candidates from the population that match according to the specified criteria;
b) allocates a memory block large enough to contain an array of elements of type BioAPI_CANDIDATE with as many elements as the number of candidates determined in (a);
c) fills the array with the candidate information for all candidates determined in (a), including the FMRAchieved of each candidate; and
d) returns the address of the array in the Candidates parameter and the size of the array in the NumberOfResults parameter.

NOTE See clause C.4 for information on the use of the FMR concept for normalized scoring and thresholding.

The memory block returned by the BioAPI function call shall be freed by the application using BioAPI_Free(see clause 8.7.2).

A BioAPI_RETURN value indicates success or specifies a particular error condition. The value BioAPI_OK indicates success. All other values represent an error condition.

Subclause 9.3.4.6

```
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                  **Candidates,
        Uint32_t                          Timeout);
```

References: 8.4.6, 9.3.4.6

Scenario:

a) The testing application does the followings:
1) Initialises the Framework under test, then installs, loads and attaches the testing BSP.
2) Opens the database to take an bir database handle.
3) Captures a biometric sample and processes it.
4) Calls BioAPI_IdentifyMatch with the conditions given by the Default Input Table and by each row of the Test Condition Table.

b) The testing BSP does the followings:

1) Checks the parameters and the contents of the pointer variables as described in the column 'BioSPI function' in the Expected Result Table.

2) Sets the return value of the BioSPI_IdentifyMatch given by each row of the Test Condition Table, then returns to the Framework.

c) The testing application does the followings:

1) Checks the return value of the BioAPI_IdentifyMatch and if it is not as described in the Expected Result Table, then issues a fail conformity response.

2) Closes the database.

3) Detaches, unloads and uninstalls the testing BSP, then terminates the Framework under test.

Parameters: The input parameters described below table are used for this assertion.

Table 79 — Default Input Table for BioAPI_IdentifyMatch_And_BioSPI_IdentifyMatch

Number	Input parameter name	Input parameter value (underscore:invalid)
1	BSPHandle	*Valid BSPHandle*
2	MaxFMRRequested	__BioAPI_NOT_SET
3	ProcessedBIR_Form	2
4	ProcessedBIR_BIRHandle	*Valid BIRHandle*
5	Type	__BioAPI_DB_TYPE
6	BIRDataBase	*Valid DBHandle*
7	TotalNumberOfTemplates	0
8	Binning	false
9	MaxNumberOfResults	5
10	no_NumberOfResults	false
11	no_Candidates	false
12	Timeout	-1

Table 80 — Test Condition Table for BioAPI_IdentifyMatch_And_BioSPI_IdentifyMatch

Test number	Input parameter name	Input parameter value (underscore:invalid)	Supported options in BSP Schema		Return value (from BioSPI)
			OPERATIONS_MASK	OPTIONS_MASK	
040601	BSPHandle	*Valid BSPHandle*	*VerifyMatch* *Capture* *Process* *DatabaseOperation*	-	*OK*
040602	BSPHandle	*Invalid BSPHandle*	*IdentifyMatch* *Capture* *Process* *DatabaseOperation*	-	-
040603	MaxFMRRequested	0	*VerifyMatch* *Capture* *Process* *DatabaseOperation*	-	*OK*
040604	MaxFMRRequested	50	*VerifyMatch* *Capture* *Process* *DatabaseOperation*	-	*OK*
040605	MaxFMRRequested	100	*VerifyMatch* *Capture* *Process* *DatabaseOperation*	-	*OK*
040606	ProcessedBIR_Form	0	*IdentifyMatch* *Capture* *Process* *DatabaseOperation*	-	*INVALID_INPUT_POINTE*
040607	ProcessedBIR_BIRHandle	*Valid BIRHandle*	*VerifyMatch* *Capture* *Process* *DatabaseOperation*	-	*OK*
040608	ProcessedBIR_BIRHandle	*Invalid BIRHandle*	*IdentifyMatch* *Capture* *Process* *DatabaseOperation*	-	*INVALID_BIR_HANDLE*

040609	Type	0	*IdentifyMatch Capture Process DatabaseOperation*	-	*INVALID_DATA*
040610	TotalNumberOfTemplates	10	*IdentifyMatch Capture Process DatabaseOperation*	-	*OK*
040611	Binning	true	*VerifyMatch Capture Process DatabaseOperation*	-	*OK*
040612	MaxNumberOfResults	0	*IdentifyMatch Capture Process DatabaseOperation*	-	*OK*
040613	no_NumberOfResults	true	*IdentifyMatch Capture Process DatabaseOperation*	-	*INVALID_OUTPUT_POINTER*
040614	no_Candidates	true	*IdentifyMatch Capture Process DatabaseOperation*	-	*INVALID_OUTPUT_POINTER*
040615	Timeout	0	*IdentifyMatch Capture Process DatabaseOperation*	-	*TIMEOUT_EXPIRED*
040616	Timeout	DefaultIdentifyTimeout	*VerifyMatch Capture Process DatabaseOperation*	-	*OK*
040617	Timeout	-2	*VerifyMatch Capture Process DatabaseOperation*	-	*INVALID_DATA*
040618	BSPHandle	*Valid BSPHandle*	*Capture Process DatabaseOperation*	-	-

NOTE: For the detail of the cell of the test number row "040619" and the column "Parameter values(underscore:invalid)", see Table1.

Table 81 — Expected Result Table for BioAPI_IdentifyMatch_And_BioSPI_IdentifyMatch

Test number	BioSPI function (BioAPI/BioSPI parameter check)	BioAPI function			Other conditions
		Return value	Output parameter name	Output parameter value	
040601	X	*OK*	-	-	-
040602	-	*indeterminate error*	-	-	-
040603	X	*OK*	-	-	-
040604	X	*OK*	-	-	-
040605	X	*OK*	-	-	-
040606	X	*indeterminate error*	-	-	-
040607	X	*OK*	-	-	-
040608	X	*INVALID_BIR_HANDLE*	-	-	-
040609	X	*indeterminate error*	-	-	-
040610	X	*OK*	-	-	-
040611	X	*OK*	-	-	-
040612	X	*OK*	-	-	-
040613	X	*indeterminate error*	-	-	-
040614	X	*indeterminate error*	-	-	-
040615	X	*TIMEOUT_EXPIRED*	-	-	-
040616	X	*OK*	-	-	-
040617	X	*indeterminate error*	-	-	-
040618	-	*FUNCTION_NOT_SUPPORTED*	-	-	-

Assertion language package

```
<?xml version="1.0" encoding="UTF-8"?>
<package name="028794f0-792e-11de-8a39-0800200c9a66">
  <author>ISO/IEC JTC1 SC37</author>
  <description>This package contains the assertion "BioAPI_IdentifyMatch_And_BioSPI_IdentifyMatch".</description>

  <!-- ************** -->
  <!-- Test assertion -->
  <!-- ************** -->
  <assertion name="BioAPI_IdentifyMatch_And_BioSPI_IdentifyMatch" model="frameworkTesting">
    <description>
      This assertion checks if BioAPI_IdentifyMatch is called with input parameters as described in Test Condition
      Table, the framework under test returns BioAPI_OK or error value in accordance with the description in
      Expected Result Table.
    </description>

    <!-- Parameter given by the CTS by referring to the Test Condition Table. -->
    <input name="_testnumber"/>
    <input name="_operationsmask1"/>
    <input name="_optionsmask1"/>
    <input name="_operationsmask2"/>
    <input name="_optionsmask2"/>
    <input name="_operationsmask3"/>
    <input name="_optionsmask3"/>
    <input name="_operationsmask4"/>
    <input name="_optionsmask4"/>
    <input name="_maxfmrrequested"/>
    <input name="_totalnumberoftemplates"/>
    <input name="_binning"/>
    <input name="_maxnumberofresults"/>
    <input name="_no_numberofresults"/>
    <input name="_no_candidates"/>
    <input name="_no_candidates"/>

    <!-- Parameter given by the CTS by referring to the Expected Result Table. -->
    <input name="_expected_return_value"/>

    <!-- Parameter given by the CTS. -->
    <input name="_bspuuid"/>
    <input name="_dbuuid"/>
    <input name="_outputformatowner"/>
    <input name="_outputformattype"/>

    <!-- Invocation of the primary activity of this assertion. -->
    <invoke activity="BioAPI_IdentifyMatch_And_BioSPI_IdentifyMatch"/>

    <!-- Bind activities to check the BioSPI function invoked by the framework under test -->
    <bind activity="SPI_IdentifyMatch" function="BioSPI_IdentifyMatch"/>
  </assertion>

  <!-- **************** -->
  <!-- Primary activity -->
  <!-- **************** -->
  <activity name="BioAPI_IdentifyMatch_And_BioSPI_IdentifyMatch">

    <!-- Attach -->
    <invoke activity="AttachWithSomeOptions" package="fb6ff5b0-7d5e-11de-8a39-0800200c9a66" break_on_break="true">
      <input name="Operationsmask1" var="_operationsmask1"/>
      <input name="Optionmask1" var="_optionsmask1"/>
      <input name="Operationsmask2" var="_operationsmask2"/>
      <input name="Optionmask2" var="_optionsmask2"/>
      <input name="Operationsmask3" var="_operationsmask3"/>
      <input name="Optionmask3" var="_optionsmask3"/>
      <input name="Operationsmask4" var="_operationsmask4"/>
      <input name="Optionmask4" var="_optionsmask4"/>
      <input name="Bspuuid" var="_bspuuid"/>
      <output name="Bsphandle" setvar="_bsphandle"/>
    </invoke>

    <!-- Invoke the activity DB Open the framework under test -->
    <invoke activity="DbOpen" package="fb6ff5b0-7d5e-11de-8a39-0800200c9a66" break_on_break="true">
      <input name="Bsphandle" var="_bsphandle"/>
      <input name="Dbuuid" var="_dbuuid"/>
      <output name="Dbhandle" setvar="dbhandle"/>
      <output name="Markerhandle" setvar="markerhandle"/>
    </invoke>
    <set name="_type" var="__BioAPI_DB_TYPE"/>
    <set name="_birdatabase" var="dbhandle"/>

    <!-- Capture and Process -->
    <invoke activity="PrepareProcessedBIR"
            package="fb6ff5b0-7d5e-11de-8a39-0800200c9a66" break_on_break="true">
      <input name="Bsphandle" var="_bsphandle"/>
      <input name="Purpose" var="__BioAPI_PURPOSE_IDENTIFY"/>
      <input name="Outputformatowner" var="_outputformatowner"/>
      <input name="Outputformattype" var="_outputformattype"/>
      <output name="Form" setvar="_processedbir_form"/>
      <output name="Birhandle" setvar="_processedbir_birhandle"/>
    </invoke>

    <!-- Set an illegal value to the parameter. -->
    <add name="_bsphandle" value="1">
      <only_if><equal_to var1="_testnumber" value2="040602"/></only_if>
    </add>
    <add name="_processedbir_birhandle" value="1">
      <only_if><equal_to var1="_testnumber" value2="040608"/></only_if>
    </add>
    <set name="_processedbir_form" value="0">
      <only_if><equal_to var1="_testnumber" value2="040606"/></only_if>
    </set>

    <!-- Invoke the function BioAPI_IdentifyMatch. -->
    <invoke function="BioAPI_IdentifyMatch">
      <input name="BSPHandle" var="_bsphandle"/>
      <input name="MaxFMRRequested" var="_maxfmrrequested"/>
      <input name="ProcessedBIR_Form" var="_processedbir_form"/>
      <input name="ProcessedBIR_BIRHandle" var="_processedbir_birhandle"/>
      <input name="Type" var="__type"/>
      <input name="BIRDataBase" var="__birdatabase"/>
      <input name="TotalNumberOfTemplates" var="_totalnumberoftemplates"/>
      <input name="Binning" var="_binning"/>
      <input name="MaxNumberOfResults" var="_maxnumberofresults"/>
      <input name="no_NumberOfResults" var="_no_numberofresults"/>
      <input name="no_Candidates" var="_no_candidates"/>
      <input name="Timeout" var="_timeout"/>
      <output name="NumberOfResults" setvar="numberofresults"/>
      <output name="Candidate_1_Type" setvar="candidate_1_type"/>
      <output name="Candidate_1_BIRInDataBase" setvar="candidate_1_birindatabase"/>
      <output name="Candidate_1_BIRInArray" setvar="candidate_1_birinarray"/>
      <output name="Candidate_1_FMRAchieved" setvar="candidate_1_fmrachieved"/>
      <return setvar="return"/>
    </invoke>

    <!-- Extract the error code from the error value. -->
    <invoke activity="ExtractErrorCode" package="fb6ff5b0-7d5e-11de-8a39-0800200c9a66" break_on_break="true">
      <output name="Returnvalue" setvar="return"/>
    </invoke>

    <!-- Assertion -->
    <assert_condition response_if_false="fail">
      <description>
        The function BioAPI_IdentifyMatch has returned the expected return value.
      </description>
      <or>
        <!-- If the "_expected_return_value" parameter is 0xFFFFFFFF,
             then check the "return" parameter is not __BioAPI_OK. (error check only) -->
        <and>
          <equal_to var1="_expected_return_value" var2="_indeterminate_error"/>
          <not_equal_to var1="return" var2="__BioAPI_OK"/>
        </and>
        <and>
          <not_equal_to var1="_expected_return_value" var2="_indeterminate_error"/>
          <equal_to var1="return" var2="_expected_return_value"/>
        </and>
      </or>
    </assert_condition>

    <!-- If Test Number is 040602, make _bsphandle to an valid value. -->
    <subtract name="_bsphandle" value="1">
      <only_if><equal_to var1="_testnumber" value2="040602"/></only_if>
    </subtract>

    <!-- Invoke the functions BioAPI_DbClose -->
    <invoke activity="DbClose" package="fb6ff5b0-7d5e-11de-8a39-0800200c9a66" break_on_break="true">
      <input name="Bsphandle" var="_bsphandle"/>
      <input name="Dbhandle" var="dbhandle"/>
    </invoke>

    <!-- Invoke the functions Detach -->
    <invoke activity="BSPDetach" package="fb6ff5b0-7d5e-11de-8a39-0800200c9a66" break_on_break="true">
      <input name="Bsphandle" var="_bsphandle"/>
      <input name="Bspuuid" var="_bspuuid"/>
    </invoke>
  </activity>

  <!-- ************************************** -->
  <!-- Activity bound to BioSPI_IdentifyMatch -->
  <!-- ************************************** -->
  <activity name="SPI_IdentifyMatch">
   <input name="BSPHandle"/>
   <input name="MaxFMRRequested"/>
   <input name="ProcessedBIR_Form"/>
```

```
    <input name="ProcessedBIR_BIRHandle"/>
    <input name="Type"/>
    <input name="BIRDataBase"/>
    <input name="TotalNumberOfTemplates"/>
    <input name="Binning"/>
    <input name="MaxNumberOfResults"/>
    <input name="no_NumberOfResults"/>
    <input name="no_Candidates"/>
    <input name="Timeout"/>
    <output name="NumberOfResults"/>
    <output name="Candidate_1_Type"/>
    <output name="Candidate_1_BIRInDataBase"/>
    <output name="Candidate_1_BIRInArray"/>
    <output name="Candidate_1_FMRAchieved"/>
     <output name="return"/>

     <!-- check API=SPI -->
     <assert_condition response_if_false="fail">
       <and>
         <equal_to var1="BSPHandle" var2="_bsphandle"/>
         <equal_to var1="MaxFMRRequested" var2="_maxfmrrequested"/>
         <equal_to var1="ProcessedBIR_Form" var2="_processedbir_form"/>
         <equal_to var1="ProcessedBIR_BIRHandle" var2="_processedbir_birhandle"/>
         <equal_to var1="Type" var2="_type"/>
         <equal_to var1="BIRDataBase" var2="_birdatabase"/>
         <equal_to var1="TotalNumberOfTemplates" var2="_totalnumberoftemplates"/>
         <equal_to var1="Binning" var2="_binning"/>
         <equal_to var1="MaxNumberOfResults" var2="_maxnumberofresults"/>
         <equal_to var1="no_NumberOfResults" var2="_no_numberofresults"/>
         <equal_to var1="no_Candidates" var2="_no_candidates"/>
         <equal_to var1="Timeout" var2="_timeout"/>
       </and>
     </assert_condition>
   </activity>
</package>
```

8.27 Assertion 4.7 – BioAPI_Enroll_And_BioSPI_Enroll

Description: This assertion calls BioAPI_Enroll with the input parameters described in Default Input Table and Test Condition Table, and checks if the framework under test returns BioAPI_OK or an error value in accordance with the description in Expected Result Table. The assertion is according to the following statement in the subclauses from BioAPI 2.0 standard document (ISO/IEC19784-1:2006), which are described in References in this subclause in this part of ISO/IEC 24709.

Excerpts:

Subclause 8.4.7

```
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);
```

This function captures biometric data from the attached device (sensor unit) for the purpose of creating a ProcessedBIR for the purpose of enrollment.

The optional input ReferenceTemplate is provided for use in creating the NewTemplate, if the BSP supports the template update capability. When present, use of the input ReferenceTemplate by the BSP to create the output NewTemplate is optional.

If the BSP supports an internal (or BSP-controlled) BIR database (e.g., smartcard or identification engine), it may optionally return the UUID assigned to the newly created ReferenceTemplate as stored within that BSP-controlled BIR database. The UUID value shall be the same as that included in the BIR header, if present.

Subclause 9.3.4.7

```
BioSPI_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);
```

References: 8.4.7, 9.3.4.7

Scenario:

a) The testing application does the followings:

1) Initialises the Framework under test, then installs, loads and attaches the testing BSP.

2) Enrolls to create a template.

3) Calls BioAPI_Enroll with the conditions given by the Default Input Table and by each row of the Test Condition Table.

b) The testing BSP does the followings:

1) Checks the parameters and the contents of the pointer variables as described in the column 'BioSPI function' in the Expected Result Table.

2) Sets the return value of the BioSPI_Enroll given by each row of the Test Condition Table, then returns to the Framework.

c) The testing application does the followings:

1) Checks the return value of the BioAPI_Enroll and if it is not as described in the Expected Result Table, then issues a fail conformity response.

2) Detaches, unloads and uninstalls the testing BSP, then terminates the Framework under test.

Parameters: The input parameters described below table are used for this assertion.

Table 82 — Default Input Table for BioAPI_Enroll_And_BioSPI_Enroll

Number	Input parameter name	Input parameter value (underscore:invalid)
1	BSPHandle	*Valid BSPHandle*
2	Purpose	_BioAPI_PURPOSE_ENROLL
3	Left	false
4	Right	false
5	Thumb	false
6	PointerFinger	false
7	MiddleFinger	false
8	RingFinger	false
9	LittleFinger	false
10	Multiple	false
11	OutputFormatOwner	N/A
12	OutputFormatType	N/A
13	ReferenceTemplate_Form	0
14	no_NewTemplate	false
15	Payload	N/A
16	Timeout	-1
17	no_AuditData	true
18	no_TemplateUUID	true

NOTE: These parameter names "Left","Right" ,"Thumb","PointerFinger","MiddleFinger","RingFinger","LittleFinger" and "Multiple" are based on the definitions in ISO/IEC 24709-1 9.2.6.

Table 83 — Test Condition Table for BioAPI_Enroll_And_BioSPI_Enroll

Test number	Input parameter name	Input parameter value (underscore:invalid)	Supported options in BSP Schema		Return value (from BioSPI)
			OPERATIONS_MASK	OPTIONS_MASK	
040701	BSPHandle	*Valid BSPHandle*	*Enroll*	-	*OK*
040702	BSPHandle	*Invalid BSPHandle*	*Enroll*	-	-
040703	Purpose	BioAPI_PURPOSE_VERIFY	*Enroll*	-	*INCONSISTENT_PURPOSE*
040704	Purpose	BioAPI_PURPOSE_IDENTIFY	*Enroll*	-	*INCONSISTENT_PURPOSE*
040705	Purpose	__BioAPI_PURPOSE_ENROLL_FOR_VERIFICATION_ONLY	*Enroll*	-	*OK*
040706	Purpose	__BioAPI_PURPOSE_ENROLL_FOR_IDENTIFICATION_ONLY	*Enroll*	-	*OK*
040707	Purpose	BioAPI_PURPOSE_AUDIT	*Enroll*	-	*INCONSISTENT_PURPOSE*
040708	Purpose	BioAPI_NO_PURPOSE_AVAILABLE	*Enroll*	-	*PURPOSE_NOT_SUPPORTED*
040709	Purpose	*Invalid Purpose*	*Enroll*	-	*INCONSISTENT_PURPOSE*
040710	Left	true	*Enroll*	*SubtypeToCapture*	*OK*
040711	Left	true	*Enroll*	-	*INVALID_DATA*
040712	Right	true	*Enroll*	*SubtypeToCapture*	*OK*
040713	Right	true	*Enroll*	-	*INVALID_DATA*
040714	(Left, Thumb)	(true, true)	*Enroll*	*SubtypeToCapture*	*OK*
040715	(Left, Thumb)	(true, true)	*Enroll*	-	*INVALID_DATA*
040716	(Left, PointerFinger)	(true, true)	*Enroll*	*SubtypeToCapture*	*OK*
040717	(Left, PointerFinger)	(true, true)	*Enroll*	-	*INVALID_DATA*
040718	(Left, MiddleFinger)	(true, true)	*Enroll*	*SubtypeToCapture*	*OK*
040719	(Left, MiddleFinger)	(true, true)	*Enroll*	-	*INVALID_DATA*
040720	(Left, RingFinger)	(true, true)	*Enroll*	*SubtypeToCapture*	*OK*
040721	(Left, RingFinger)	(true, true)	*Enroll*	-	*INVALID_DATA*
040722	(Left, LittleFinger)	(true, true)	*Enroll*	*SubtypeToCapture*	*OK*
040723	(Left, LittleFinger)	(true, true)	*Enroll*	-	*INVALID_DATA*
040724	(Right, Thumb)	(true, true)	*Enroll*	*SubtypeToCapture*	*OK*
040725	(Right, Thumb)	(true, true)	*Enroll*	-	*INVALID_DATA*
040726	(Right, PointerFinger)	(true, true)	*Enroll*	*SubtypeToCapture*	*OK*
040727	(Right, PointerFinger)	(true, true)	*Enroll*	-	*INVALID_DATA*
040728	(Right, MiddleFinger)	(true, true)	*Enroll*	*SubtypeToCapture*	*OK*
040729	(Right, MiddleFinger)	(true, true)	*Enroll*	-	*INVALID_DATA*
040730	(Right, RingFinger)	(true, true)	*Enroll*	*SubtypeToCapture*	*OK*
040731	(Right, RingFinger)	(true, true)	*Enroll*	-	*INVALID_DATA*
040732	(Right, LittleFinger)	(true, true)	*Enroll*	*SubtypeToCapture*	*OK*
040733	(Right, LittleFinger)	(true, true)	*Enroll*	-	*INVALID_DATA*
040734	Multiple	true	*Enroll*	*SubtypeToCapture*	*OK*
040735	Multiple	true	*Enroll*	-	*INVALID_DATA*
040736	Thumb	true	*Enroll*	*SubtypeToCapture*	*OK*
040737	Thumb	true	*Enroll*	-	*INVALID_DATA*
040738	PointerFinger	true	*Enroll*	*SubtypeToCapture*	*OK*
040739	PointerFinger	true	*Enroll*	-	*INVALID_DATA*
040740	MiddleFinger	true	*Enroll*	*SubtypeToCapture*	*OK*
040741	MiddleFinger	true	*Enroll*	-	*INVALID_DATA*
040742	RingFinger	true	*Enroll*	*SubtypeToCapture*	*OK*
040743	RingFinger	true	*Enroll*	-	*INVALID_DATA*
040744	LittleFinger	true	*Enroll*	*SubtypeToCapture*	*OK*
040745	LittleFinger	true	*Enroll*	-	*INVALID_DATA*
040746	(Left, Right, Thumb, PointerFinger, MiddleFinger, RingFinger, LittleFinger, Multiple)	(true, true, true, true, true, true, true, true)	*Enroll*	-	*INVALID_DATA*
040747	(ReferenceTemplate_Form, ReferenceTemplate_BIRHandle)	(2, *Valid BIR Handle)*	*Enroll*	*TemplateUpdate*	*OK*
040748	(ReferenceTemplate_Form, ReferenceTemplate_BIRHandle)	(2, *Valid BIR Handle)*	*Enroll*	-	*INVALID_DATA*

040749	(ReferenceTemplate_Form, ReferenceTemplate_BIRHandle)	(2, *Invalid BIR Handle*)	*Enroll*	*TemplateUpdate*	*INVALID_DATA*
040750	(ReferenceTemplate_Form, ReferenceTemplate_BIRHandle)	(2, *Invalid BIR Handle*)	*Enroll*	-	*INVALID_DATA*
040751	no_NewTemplate	true	*Enroll*	-	*INVALID_OUTPUT_ POINTER*
040752	Timeout	0	*Enroll*	-	*TIMEOUT_EXPIRED*
040753	Timeout	DefaultEnrollTimeout	*Enroll*	-	*OK*
040754	Timeout	-2	*Enroll*	-	*INVALID_DATA*
040755	no_AuditData	false	*Enroll*	*Raw QualityRaw*	*OK*
040756	no_AuditData	false	*Enroll*	-	*FUNCTION_NOT_ SUPPORTED*
040757	no_TemplateUUID	false	*Enroll*	-	*OK*
040758	BSPHandle	*Valid BSPHandle*	-	-	-

NOTE1: For the detail of the cell of the test number row "040767" and the column "Parameter values(underscore:invalid)", see Table1.
NOTE2: These parameter names "Left","Right" ,"Thumb","PointerFinger","MiddleFinger","RingFinger","LittleFinger" and "Multiple" are based on the definitions in ISO/IEC 24709-1 9.2.6.

Table 84 — Expected Result Table for BioAPI_Enroll_And_BioSPI_Enroll

Test number	BioSPI function (BioAPI/BioSPI parameter check)	BioAPI function			Other conditions
		Return value	Output parameter name	Output parameter value	
040701	X	*OK*	-	-	-
040702	-	*indeterminate error*	-	-	-
040703	X	*indeterminate error*	-	-	-
040704	X	*indeterminate error*	-	-	-
040705	X	*OK*	-	-	-
040706	X	*OK*	-	-	-
040707	X	*indeterminate error*	-	-	-
040708	X	*PURPOSE_NOT_SUPPORTED*	-	-	-
040709	X	*indeterminate error*	-	-	-
040710	X	*OK*	-	-	-
040711	X	*indeterminate error*	-	-	-
040712	X	*OK*	-	-	-
040713	X	*indeterminate error*	-	-	-
040714	X	*OK*	-	-	-
040715	X	*indeterminate error*	-	-	-
040716	X	*OK*	-	-	-
040717	X	*indeterminate error*	-	-	-
040718	X	*OK*	-	-	-
040719	X	*indeterminate error*	-	-	-
040720	X	*OK*	-	-	-
040721	X	*indeterminate error*	-	-	-
040722	X	*OK*	-	-	-
040723	X	*indeterminate error*	-	-	-
040724	X	*OK*	-	-	-
040725	X	*indeterminate error*	-	-	-
040726	X	*OK*	-	-	-
040727	X	*indeterminate error*	-	-	-
040728	X	*OK*	-	-	-
040729	X	*indeterminate error*	-	-	-
040730	X	*OK*	-	-	-
040731	X	*indeterminate error*	-	-	-
040732	X	*OK*	-	-	-
040733	X	*indeterminate error*	-	-	-
040734	X	*OK*	-	-	-
040735	X	*indeterminate error*	-	-	-
040736	X	*OK*	-	-	-
040737	X	*indeterminate error*	-	-	-
040738	X	*OK*	-	-	-
040739	X	*indeterminate error*	-	-	-
040740	X	*OK*	-	-	-
040741	X	*indeterminate error*	-	-	-
040742	X	*OK*	-	-	-

040743	X	*indeterminate error*	-	-	-
040744	X	*OK*	-	-	-
040745	X	*indeterminate error*	-	-	-
040746	X	*indeterminate error*	-	-	-
040747	X	*OK*	-	-	-
040748	X	*indeterminate error*	-	-	-
040749	X	*INVALID_BIR_HANDLE*	-	-	-
040750	X	*INVALID_BIR_HANDLE*	-	-	-
040751	X	*indeterminate error*	-	-	-
040752	X	*TIMEOUT_EXPIRED*	-	-	-
040753	X	*OK*	-	-	-
040754	X	*indeterminate error*	-	-	-
040755	X	*OK*	-	-	-
040756	X	*indeterminate error*	-	-	-
040757	X	*OK*	-	-	-
040758	-	*indeterminate error*	-	-	-

Assertion language package

```
<?xml version="1.0" encoding="UTF-8"?>
<package name="12892d50-792e-11de-8a39-0800200c9a66">
  <author>ISO/IEC JTC1 SC37</author>
  <description>This package contains the assertion "BioAPI_Enroll_And_BioSPI_Enroll".</description>

  <!-- ************** -->
  <!-- Test assertion -->
  <!-- ************** -->
  <assertion name="BioAPI_Enroll_And_BioSPI_Enroll" model="frameworkTesting">
    <description>
      This assertion checks if BioAPI_Enroll is called with input parameters as described in Test Condition Table,
      the framework under test returns BioAPI_OK or error value in accordance with the description in Expected
      Result Table.
    </description>

    <!-- Parameter given by the CTS by referring to the Test Condition Table. -->
    <input name="_testnumber"/>
    <input name="_operationsmask"/>
    <input name="_optionsmask"/>
    <input name="_purpose"/>
    <input name="_left"/>
    <input name="_right"/>
    <input name="_thumb"/>
    <input name="_pointerfinger"/>
    <input name="_middlefinger"/>
    <input name="_ringfinger"/>
    <input name="_littlefinger"/>
    <input name="_multiple"/>
    <input name="_outputformatowner"/>
    <input name="_outputformattype"/>
    <input name="_no_newtemplate"/>
    <input name="_payload"/>
    <input name="_timeout"/>
    <input name="_no_auditdata"/>
    <input name="_no_templateuuid"/>

    <!-- Parameter given by the CTS by referring to the Expected Result Table. -->
    <input name="_expected_return_value"/>

    <!-- Parameter given by the CTS. -->
    <input name="_bspuuid"/>

    <!-- Invocation of the primary activity of this assertion. -->
    <invoke activity="BioAPI_Enroll_And_BioSPI_Enroll"/>

    <!-- Bind activities to check the BioSPI function invoked by the framework under test -->
    <bind activity="SPI_Enroll" function="BioSPI_Enroll"/>
  </assertion>

  <!-- **************** -->
  <!-- Primary activity -->
  <!-- **************** -->
  <activity name="BioAPI_Enroll_And_BioSPI_Enroll">

    <!-- Invoke the activity BSP Attach the framework under test -->
    <invoke activity="BSPAttach" package="fb6ff5b0-7d5e-11de-8a39-0800200c9a66" break_on_break="true">
      <input name="Operationsmask" var="_operationsmask"/>
      <input name="Optionsmask" var="_optionsmask"/>
      <input name="Bspuuid" var="_bspuuid"/>
      <output name="Newbsphandle" setvar="_bsphandle"/>
    </invoke>

    <!-- If Test Number is 040757, 040758, 040759 or 040760, PrepareReferenceTemplate. -->
    <set name="_referencetemplate_form" value="0"/>
```

```
<set name="_referencetemplate_birhandle" value=""/>
<set name="_noparamcheck" value="false"/>

<!-- Prepare ReferenceTemplate. -->
<set name="_noparamcheck" value="true"/>
<invoke activity="PrepareReferenceTemplate"
        package="fb6ff5b0-7d5e-11de-8a39-0800200c9a66" break_on_break="true">
  <only_if>
    <equal_to var1="_testnumber" value2="040747"/>
    <equal_to var1="_testnumber" value2="040748"/>
    <equal_to var1="_testnumber" value2="040749"/>
    <equal_to var1="_testnumber" value2="040750"/>
  </only_if>
  <input name="Bsphandle" var="_bsphandle"/>
  <input name="Outputformatowner" var="_outputformatowner"/>
  <input name="Outputformattype" var="_outputformattype"/>
  <output name="Form" setvar="_referencetemplate_form"/>
  <output name="Birhandle" setvar="_referencetemplate_birhandle"/>
</invoke>
<set name="_noparamcheck" value="false"/>

<!-- Set an illegal value to the parameter. -->
<add name="_bsphandle" value="1">
  <only_if><equal_to var1="_testnumber" value2="040702"/></only_if>
</add>
<add name="_referencetemplate_birhandle" value="1">
  <only_if>
    <equal_to var1="_testnumber" value2="040749"/>
    <equal_to var1="_testnumber" value2="040750"/>
  </only_if>
</add>

<!-- Invoke the function BioAPI_Enroll. -->
<invoke function="BioAPI_Enroll">
  <input name="BSPHandle" var="_bsphandle"/>
  <input name="Purpose" var="_purpose"/>
  <input name="Left" var="_left"/>
  <input name="Right" var="_right"/>
  <input name="Thumb" var="_thumb"/>
  <input name="PointerFinger" var="_pointerfinger"/>
  <input name="MiddleFinger" var="_middlefinger"/>
  <input name="RingFinger" var="_ringfinger"/>
  <input name="LittleFinger" var="_littlefinger"/>
  <input name="Multiple" var="_multiple"/>
  <input name="OutputFormatOwner" var="_outputformatowner"/>
  <input name="OutputFormatType" var="_outputformattype"/>
  <input name="ReferenceTemplate_Form" var="_referencetemplate_form"/>
  <input name="ReferenceTemplate_BIRHandle" var="_referencetemplate_birhandle"/>
  <input name="no_NewTemplate" var="_no_newtemplate"/>
  <input name="Payload" var="_payload"/>
  <input name="Timeout" var="_timeout"/>
  <input name="no_AuditData" var="_no_auditdata"/>
  <input name="no_TemplateUUID" var="_no_templateuuid"/>
  <output name="NewTemplate" setvar="newtemplate"/>
  <output name="AuditData" setvar="auditdata"/>
  <output name="TemplateUUID" setvar="templateuuid"/>
  <return setvar="return"/>
</invoke>

<!-- Extract the error code from the error value. -->
<invoke activity="ExtractErrorCode" package="fb6ff5b0-7d5e-11de-8a39-0800200c9a66" break_on_break="true">
  <output name="Returnvalue" setvar="return"/>
</invoke>

<!-- Assertion -->
<assert_condition response_if_false="fail">
  <description>
    The function BioAPI_Enroll has returned the expected return value.
  </description>
  <or>
    <!-- If the "_expected_return_value" parameter is 0xFFFFFFFF,
         then check the "return" parameter is not __BioAPI_OK. (error check only) -->
    <and>
      <equal_to var1="_expected_return_value" var2="_indeterminate_error"/>
      <not_equal_to var1="return" var2="__BioAPI_OK"/>
    </and>
    <and>
      <not_equal_to var1="_expected_return_value" var2="_indeterminate_error"/>
      <equal_to var1="return" var2="_expected_return_value"/>
    </and>
  </or>
</assert_condition>

<!-- If Test Number is 040702, make _bsphandle to an valid value. -->
<subtract name="_bsphandle" value="1">
```

```
      <only_if><equal_to var1="_testnumber" value2="040702"/></only_if>
    </subtract>

    <!-- Invoke the functions Detach -->
    <invoke activity="BSPDetach" package="fb6ff5b0-7d5e-11de-8a39-0800200c9a66" break_on_break="true">
      <input name="Bsphandle" var="_bsphandle"/>
      <input name="Bspuuid" var="_bspuuid"/>
    </invoke>
  </activity>

  <!-- ****************************** -->
  <!-- Activity bound to BioSPI_Enroll -->
  <!-- ****************************** -->
  <activity name="SPI_Enroll">
    <input name="BSPHandle"/>
    <input name="Purpose"/>
    <input name="Left"/>
    <input name="Right"/>
    <input name="Thumb"/>
    <input name="PointerFinger"/>
    <input name="MiddleFinger"/>
    <input name="RingFinger"/>
    <input name="LittleFinger"/>
    <input name="Multiple"/>
    <input name="OutputFormatOwner"/>
    <input name="OutputFormatType"/>
    <input name="Referencetemplate_Form"/>
    <input name="Referencetemplate_BIRHandle"/>
    <input name="no_NewTemplate"/>
    <input name="Payload"/>
    <input name="TimeOut"/>
    <input name="no_Auditdata"/>
    <input name="no_TemplateUUID"/>
    <output name="NewTemplate"/>
    <output name="AuditData"/>
    <output name="TemplateUUID"/>
    <output name="return"/>

    <!-- check API=SPI -->
    <assert_condition response_if_false="fail">
      <or>
        <not var="_noparamcheck"/>
        <and>
          <equal_to var1="BSPHandle" var2="_bsphandle"/>
          <equal_to var1="Purpose" var2="_purpose"/>
          <equal_to var1="Left" var2="_left"/>
          <equal_to var1="Right" var2="_right"/>
          <equal_to var1="Thumb" var2="_thumb"/>
          <equal_to var1="PointerFinger" var2="_pointerfinger"/>
          <equal_to var1="MiddleFinger" var2="_middlefinger"/>
          <equal_to var1="RingFinger" var2="_ringfinger"/>
          <equal_to var1="LittleFinger" var2="_littlefinger"/>
          <equal_to var1="Multiple" var2="_multiple"/>
          <equal_to var1="OutputFormatOwner" var2="_outputformatowner"/>
          <equal_to var1="OutputFormatType" var2="_outputformattype"/>
          <equal_to var1="Referencetemplate_Form" var2="_referencetemplate_form"/>
          <equal_to var1="Referencetemplate_BIRHandle" var2="_referencetemplate_birhandle"/>
          <equal_to var1="no_NewTemplate" var2="_no_newtemplate"/>
          <equal_to var1="Payload" var2="_payload"/>
          <equal_to var1="TimeOut" var2="_timeout"/>
          <equal_to var1="no_Auditdata" var2="_no_auditdata"/>
          <equal_to var1="no_TemplateUUID" var2="_no_templateuuid"/>
        </and>
      </or>
    </assert_condition>
  </activity>
</package>
```

8.28 Assertion 4.8 – BioAPI_Verify_And_BioSPI_Verify

Description: This assertion calls BioAPI_Verify with the input parameters described in Default Input Table and Test Condition Table, and checks if the framework under test returns BioAPI_OK or an error value in accordance with the description in Expected Result Table. The assertion is according to the following statement in the subclauses from BioAPI 2.0 standard document (ISO/IEC19784-1:2006), which are described in References in this subclause in this part of ISO/IEC 24709.

Excerpts:

Subclause 8.4.8

```
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);
```

This function captures biometric data from the attached device (sensor unit), and compares it against the ReferenceTemplate.

The application shall request a maximum FMR value criterion (threshold) for a successful match. The Boolean Result indicates whether verification was successful or not, and the FMRAchieved is a FMR value (score) indicating how closely the BIRs actually matched.

By setting the AdaptedBIR pointer to non-NULL, the application can request that a BIR be constructed by adapting the ReferenceTemplate using the ProcessedBIR. A new handle is returned to the AdaptedBIR. If the match is successful, an attempt may be made to adapt the ReferenceTemplate with information taken from the ProcessedBIR. (Not all BSPs perform adaptation). The resulting AdaptedBIR should now be considered an optimal enrollment template, and be saved in the BIR database. (It is up to the application whether it uses or discards this data). It is important to note that adaptation may not occur in all cases. In the event of an adaptation, this function stores the handle to the new BIR in the memory pointed to by the AdaptedBIR parameter.

If a Payload is associated with the ReferenceTemplate, the Payload may be returned upon successful verification if the FMRAchieved is sufficiently stringent; this is controlled by the policy of the BSP and specified in its schema.

Since the BioAPI_Verify operation includes a capture, it serializes use of the sensor device. If two or more applications are racing for the device, the losers will wait until the operation completes or the timeout expires. This serialization takes place in all functions that capture data. The BSP is responsible for serializing. It may do this by either returning ‘busy’ (BioAPI_UNIT_ IN _USE) or by queuing requests.

The memory block returned by the BioAPI function call shall be freed by the application as soon as it is no longer needed using BioAPI_Free (see clause 8.7.2). Output BIRs can be retrieved by a call to BioAPI_GetBIRFromHandle, which releases the handle, or the handle can be released without retrieving the BIR through BioAPI_FreeBIRHandle.

Subclause 9.3.4.8

```
BioSPI_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);
```

References: 8.4.8, 9.3.4.8

Scenario:

a) The testing application does the followings:

1) Initialises the Framework under test, then installs, loads and attaches the testing BSP.
2) Enrolls to create a template.
3) Calls BioAPI_Verify with the conditions given by the Default Input Table and by each row of the Test Condition Table.

b) The testing BSP does the followings:

1) Checks the parameters and the contents of the pointer variables as described in the column 'BioSPI function' in the Expected Result Table.
2) Sets the return value of the BioSPI_Verify given by each row of the Test Condition Table, then returns to the Framework.

c) The testing application does the followings:

1) Checks the return value of the BioAPI_Verify and if it is not as described in the Expected Result Table, then issues a fail conformity response.
2) Detaches, unloads and uninstalls the testing BSP, then terminates the Framework under test.

Parameters: The input parameters described below table are used for this assertion.

Table 85 — Default Input Table for BioAPI_Verify_And_BioSPI_Verify

Number	Input parameter name	Input parameter value (underscore:invalid)
1	BSPHandle	*Valid BSPHandle*
2	MaxFMRRequested	__BioAPI_NOT_SET
3	ReferenceTemplate_Form	2
4	ReferenceTemplate_BIRHandle	*Valid BIRHandle*
5	Left	false
6	Right	false
7	Thumb	false
8	PointerFinger	false
9	MiddleFinger	false
10	RingFinger	false
11	LittleFinger	false
12	Multiple	false
13	no_AdaptedBIR	true
14	no_Result	false
15	no_FMRAchieved	false
16	no_Payload	true
17	Timeout	-1
18	no_AuditData	true

NOTE: These parameter names "Left","Right" ,"Thumb","PointerFinger","MiddleFinger","RingFinger","LittleFinger" and "Multiple" are based on the definitions in ISO/IEC 24709-1 9.2.6.

Table 86 — Test Condition Table for BioAPI_Verify_And_BioSPI_Verify

Test number	Input parameter name	Input parameter value (underscore:invalid)	Supported options in BSP Schema		Return value (from BioSPI)
			OPERATIONS_MASK	OPTIONS_MASK	
040801	BSPHandle	*Valid BSPHandle*	*Verify Capture Enroll*	-	*OK*
040802	BSPHandle	*<u>Invalid BSPHandle</u>*	*Verify Capture Enroll*	-	-
040803	MaxFMRRequested	0	*Verify Capture Enroll*	-	*OK*
040804	MaxFMRRequested	50	*Verify Capture Enroll*	-	*OK*

040805	MaxFMRRequested	100	*Verify Capture Enroll*	-	*OK*
040806	ReferenceTemplate_Form	0	*Verify Capture Enroll*	-	*INVALID_DATA*
040807	ReferenceTemplate_BIRHandle	*Valid BIRHandle*	*Verify Capture Enroll*	-	*OK*
040808	ReferenceTemplate_BIRHandle	*Invalid BIRHandle*	*Verify Capture Enroll*	-	*INVALID_BIR_HANDLE*
040809	Left	true	*Verify Capture Enroll*	*SubtypeToCapture*	*OK*
040810	Left	true	*Verify Capture Enroll*	-	*INVALID_DATA*
040811	Right	true	*Verify Capture Enroll*	*SubtypeToCapture*	*OK*
040812	Right	true	*Verify Capture Enroll*	-	*INVALID_DATA*
040813	(Left, Thumb)	(true, true)	*Verify Capture Enroll*	*SubtypeToCapture*	*OK*
040814	(Left, Thumb)	(true, true)	*Verify Capture Enroll*	-	*INVALID_DATA*
040815	(Left, PointerFinger)	(true, true)	*Verify Capture Enroll*	*SubtypeToCapture*	*OK*
040816	(Left, PointerFinger)	(true, true)	*Verify Capture Enroll*	-	*INVALID_DATA*
040817	(Left, MiddleFinger)	(true, true)	*Verify Capture Enroll*	*SubtypeToCapture*	*OK*
040818	(Left, MiddleFinger)	(true, true)	*Verify Capture Enroll*	-	*INVALID_DATA*
040819	(Left, RingFinger)	(true, true)	*Verify Capture Enroll*	*SubtypeToCapture*	*OK*
040820	(Left, RingFinger)	(true, true)	*Verify Capture Enroll*	-	*INVALID_DATA*
040821	(Left, LittleFinger)	(true, true)	*Verify Capture Enroll*	*SubtypeToCapture*	*OK*
040822	(Left, LittleFinger)	(true, true)	*Verify Capture Enroll*	-	*INVALID_DATA*
040823	(Right, Thumb)	(true, true)	*Verify Capture Enroll*	*SubtypeToCapture*	*OK*
040824	(Right, Thumb)	(true, true)	*Verify Capture Enroll*	-	*INVALID_DATA*
040825	(Right, PointerFinger)	(true, true)	*Verify Capture Enroll*	*SubtypeToCapture*	*OK*
040826	(Right, PointerFinger)	(true, true)	*Verify Capture Enroll*	-	*INVALID_DATA*
040827	(Right, MiddleFinger)	(true, true)	*Verify Capture Enroll*	*SubtypeToCapture*	*OK*
040828	(Right, MiddleFinger)	(true, true)	*Verify Capture Enroll*	-	*INVALID_DATA*

040829	(Right, RingFinger)	(true, true)	*Verify Capture Enroll*	*SubtypeToCapture*	*OK*
040830	(Right, RingFinger)	(true, true)	*Verify Capture Enroll*	-	*INVALID_DATA*
040831	(Right, LittleFinger)	(true, true)	*Verify Capture Enroll*	*SubtypeToCapture*	*OK*
040832	(Right, LittleFinger)	(true, true)	*Verify Capture Enroll*	-	*INVALID_DATA*
040833	Multiple	true	*Verify Capture Enroll*	*SubtypeToCapture*	*OK*
040834	Multiple	true	*Verify Capture Enroll*	-	*INVALID_DATA*
040835	Thumb	true	*Verify Capture Enroll*	*SubtypeToCapture*	*OK*
040836	Thumb	true	*Verify Capture Enroll*	-	*INVALID_DATA*
040837	PointerFinger	true	*Verify Capture Enroll*	*SubtypeToCapture*	*OK*
040838	PointerFinger	true	*Verify Capture Enroll*	-	*INVALID_DATA*
040839	MiddleFinger	true	*Verify Capture Enroll*	*SubtypeToCapture*	*OK*
040840	MiddleFinger	true	*Verify Capture Enroll*	-	*INVALID_DATA*
040841	RingFinger	true	*Verify Capture Enroll*	*SubtypeToCapture*	*OK*
040842	RingFinger	true	*Verify Capture Enroll*	-	*INVALID_DATA*
040843	LittleFinger	true	*Verify Capture Enroll*	*SubtypeToCapture*	*OK*
040844	LittleFinger	true	*Verify Capture Enroll*	-	*INVALID_DATA*
040845	(Left, Right, Thumb, PointerFinger, MiddleFinger, RingFinger, LittleFinger, Multiple)	(true, true, true, true, true, true, true, true)	*Verify Capture Enroll*	-	*INVALID_DATA*
040846	no_AdaptedBIR	false	*Verify Capture Enroll*	-	*OK*
040847	no_Result	true	*Verify Capture Enroll*	-	*INVALID_OUTPUT_POINTER*
040848	no_FMRAchieved	true	*Verify Capture Enroll*	-	*FUNCTION_NOT_SUPPORTED*
040849	Timeout	0	*Verify Capture Enroll*	-	*TIMEOUT_EXPIRED*
040850	Timeout	DefaultVerifyTimeout	*Verify Capture Enroll*	-	*OK*
040851	Timeout	-2	*Verify Capture Enroll*	-	*INVALID_DATA*
040852	no_AuditData	false	*Verify Capture Enroll*	*Raw QualityRaw*	*OK*

040853	no_AuditData	false	*Verify Capture Enroll*	-	*FUNCTION_NOT_SUPPORTED*
040854	BSPHandle	*Valid BSPHandle*	*Capture Enroll*	-	-

NOTE1: For the detail of the cell of the test number row "040854" and the column "Parameter values(underscore:invalid)", see Table1.
NOTE2: These parameter names "Left","Right" ,"Thumb","PointerFinger","MiddleFinger","RingFinger","LittleFinger" and "Multiple" are based on the definitions in ISO/IEC 24709-1 9.2.6.

Table 87 — Expected Result Table for BioAPI_Verify_And_BioSPI_Verify

Test number	BioSPI function (BioAPI/BioSPI parameter check)	BioAPI function			Other conditions
		Return value	Output parameter name	Output parameter value	
040801	X	*OK*	-	-	-
040802	-	*indeterminate error*	-	-	-
040803	X	*OK*	-	-	-
040804	X	*OK*	-	-	-
040805	X	*OK*	-	-	-
040806	X	*indeterminate error*	-	-	-
040807	X	*OK*	-	-	-
040808	X	*INVALID_BIR_HANDLE*	-	-	-
040809	X	*OK*	-	-	-
040810	X	*indeterminate error*	-	-	-
040811	X	*OK*	-	-	-
040812	X	*indeterminate error*	-	-	-
040813	X	*OK*	-	-	-
040814	X	*indeterminate error*	-	-	-
040815	X	*OK*	-	-	-
040816	X	*indeterminate error*	-	-	-
040817	X	*OK*	-	-	-
040818	X	*indeterminate error*	-	-	-
040819	X	*OK*	-	-	-
040820	X	*indeterminate error*	-	-	-
040821	X	*OK*	-	-	-
040822	X	*indeterminate error*	-	-	-
040823	X	*OK*	-	-	-
040824	X	*indeterminate error*	-	-	-
040825	X	*OK*	-	-	-
040826	X	*indeterminate error*	-	-	-
040827	X	*OK*	-	-	-
040828	X	*indeterminate error*	-	-	-
040829	X	*OK*	-	-	-
040830	X	*indeterminate error*	-	-	-
040831	X	*OK*	-	-	-
040832	X	*indeterminate error*	-	-	-
040833	X	*OK*	-	-	-
040834	X	*indeterminate error*	-	-	-
040835	X	*OK*	-	-	-
040836	X	*indeterminate error*	-	-	-
040837	X	*OK*	-	-	-
040838	X	*indeterminate error*	-	-	-
040839	X	*OK*	-	-	-
040840	X	*indeterminate error*	-	-	-
040841	X	*OK*	-	-	-
040842	X	*indeterminate error*	-	-	-
040843	X	*OK*	-	-	-
040844	X	*indeterminate error*	-	-	-
040845	X	*indeterminate error*	-	-	-
040846	X	*OK*	-	-	-
040847	X	*indeterminate error*	-	-	-
040848	X	*indeterminate error*	-	-	-
040849	X	*TIMEOUT_EXPIRED*	-	-	-
040850	X	*indeterminate error*	-	-	-
040851	X	*OK*	-	-	-
040852	X	*OK*	-	-	-
040853	X	*indeterminate error*	-	-	-
040854	-	*indeterminate error*	-	-	-

Assertion language package

```
<?xml version="1.0" encoding="UTF-8"?>
<package name="233d9af0-792e-11de-8a39-0800200c9a66">
  <author>ISO/IEC JTC1 SC37</author>
  <description>This package contains the assertion "BioAPI_Verify_And_BioSPI_Verify".</description>

  <!-- ************** -->
  <!-- Test assertion -->
  <!-- ************** -->
  <assertion name="BioAPI_Verify_And_BioSPI_Verify" model="frameworkTesting">
    <description>
      This assertion checks if BioAPI_Verify is called with input parameters as described in Test Condition Table,
      the framework under test returns BioAPI_OK or error value in accordance with the description in Expected
      Result Table.
    </description>

    <!-- Parameter given by the CTS by referring to the Test Condition Table. -->
    <input name="_testnumber"/>
    <input name="_operationsmask1"/>
    <input name="_optionsmask1"/>
    <input name="_operationsmask2"/>
    <input name="_optionsmask2"/>
    <input name="_operationsmask3"/>
    <input name="_optionsmask3"/>
    <input name="_operationsmask4"/>
    <input name="_optionsmask4"/>
    <input name="_maxfmrrequested"/>
    <input name="_left"/>
    <input name="_right"/>
    <input name="_thumb"/>
    <input name="_pointerfinger"/>
    <input name="_middlefinger"/>
    <input name="_ringfinger"/>
    <input name="_littlefinger"/>
    <input name="_multiple"/>
    <input name="_no_adaptedbir"/>
    <input name="_no_result"/>
    <input name="_no_fmrachieved"/>
    <input name="_no_payload"/>
    <input name="_timeout"/>
    <input name="_no_auditdata"/>

    <!-- Parameter given by the CTS by referring to the Expected Result Table. -->
    <input name="_expected_return_value"/>

    <!-- Parameter given by the CTS. -->
    <input name="_bspuuid"/>
    <input name="_outputformatowner"/>
    <input name="_outputformattype"/>

    <!-- Invocation of the primary activity of this assertion. -->
    <invoke activity="BioAPI_Verify_And_BioSPI_Verify"/>

    <!-- Bind activities to check the BioSPI function invoked by the framework under test -->
    <bind activity="SPI_Verify" function="BioSPI_Verify"/>
  </assertion>

  <!-- **************** -->
  <!-- Primary activity -->
  <!-- **************** -->
  <activity name="BioAPI_Verify_And_BioSPI_Verify">

    <!-- Attach -->
    <invoke activity="AttachWithSomeOptions"
            package="fb6ff5b0-7d5e-11de-8a39-0800200c9a66" break_on_break="true">
      <input name="Operationsmask1" var="_operationsmask1"/>
      <input name="Optionmask1" var="_optionsmask1"/>
      <input name="Operationsmask2" var="_operationsmask2"/>
      <input name="Optionmask2" var="_optionsmask2"/>
      <input name="Operationsmask3" var="_operationsmask3"/>
      <input name="Optionmask3" var="_optionsmask3"/>
      <input name="Operationsmask4" var="_operationsmask4"/>
      <input name="Optionmask4" var="_optionsmask4"/>
      <input name="Bspuuid" var="_bspuuid"/>
      <output name="Bsphandle" setvar="_bsphandle"/>
    </invoke>

    <!-- PrepareReferenceTemplate. -->
    <invoke activity="PrepareReferenceTemplate"
            package="fb6ff5b0-7d5e-11de-8a39-0800200c9a66" break_on_break="true">
      <input name="Bsphandle" var="_bsphandle"/>
      <input name="Outputformatowner" var="_outputformatowner"/>
      <input name="Outputformattype" var="_outputformattype"/>
      <output name="Form" setvar="_referencetemplate_form"/>
      <output name="Birhandle" setvar="_referencetemplate_birhandle"/>
    </invoke>

    <!-- Set an illegal value to the parameter. -->
    <add name="_bsphandle" value="1">
      <only_if><equal_to var1="_testnumber" value2="040802"/></only_if>
    </add>
    <add name="_referencetemplate_birhandle" value="1">
      <only_if><equal_to var1="_testnumber" value2="040808"/></only_if>
    </add>
    <set name="_referencetemplate_form" value="0">
      <only_if><equal_to var1="_testnumber" value2="040806"/></only_if>
    </set>

    <!-- Invoke the function BioAPI_Verify. -->
    <invoke function="BioAPI_Verify">
      <input name="BSPHandle" var="_bsphandle"/>
      <input name="MaxFMRRequested" var="_maxfmrrequested"/>
      <input name="ReferenceTemplate_Form" var="_referencetemplate_form"/>
      <input name="ReferenceTemplate_BIRHandle" var="_referencetemplate_birhandle"/>
      <input name="Left" var="_left"/>
      <input name="Right" var="_right"/>
      <input name="Thumb" var="_thumb"/>
      <input name="PointerFinger" var="_pointerfinger"/>
      <input name="MiddleFinger" var="_middlefinger"/>
      <input name="RingFinger" var="_ringfinger"/>
      <input name="LittleFinger" var="_littlefinger"/>
      <input name="Multiple" var="_multiple"/>
      <input name="no_AdaptedBIR" var="_no_adaptedbir"/>
      <input name="no_Result" var="_no_result"/>
      <input name="no_FMRAchieved" var="_no_fmrachieved"/>
      <input name="no_Payload" var="_no_payload"/>
      <input name="Timeout" var="_timeout"/>
      <input name="no_AuditData" var="_no_auditdata"/>
      <output name="AdaptedBIR" setvar="adaptedbir"/>
      <output name="Result" setvar="result"/>
      <output name="FMRAchieved" setvar="fmrachieved"/>
      <output name="Payload" setvar="payload"/>
      <output name="AuditData" setvar="auditdata"/>
      <return setvar="return"/>
    </invoke>

    <!-- Extract the error code from the error value. -->
    <invoke activity="ExtractErrorCode"
            package="fb6ff5b0-7d5e-11de-8a39-0800200c9a66" break_on_break="true">
      <output name="Returnvalue" setvar="return"/>
    </invoke>

    <!-- Assertion -->
    <assert_condition response_if_false="fail">
      <description>
        The function BioAPI_Verify has returned the expected return value.
      </description>
      <or>
        <!-- If the "_expected_return_value" parameter is 0xFFFFFFFF,
             then check the "return" parameter is not __BioAPI_OK. (error check only) -->
        <and>
          <equal_to var1="_expected_return_value" var2="_indeterminate_error"/>
          <not_equal_to var1="return" var2="__BioAPI_OK"/>
        </and>
        <and>
          <not_equal_to var1="_expected_return_value" var2="_indeterminate_error"/>
          <equal_to var1="return" var2="_expected_return_value"/>
        </and>
      </or>
    </assert_condition>

    <!-- If Test Number is 040802, make _bsphandle to an valid value. -->
    <subtract name="_bsphandle" value="1">
      <only_if><equal_to var1="_testnumber" value2="040802"/></only_if>
    </subtract>

    <!-- Invoke the functions Detach -->
    <invoke activity="BSPDetach" package="fb6ff5b0-7d5e-11de-8a39-0800200c9a66" break_on_break="true">
      <input name="Bsphandle" var="_bsphandle"/>
      <input name="Bspuuid" var="_bspuuid"/>
    </invoke>
  </activity>

  <!-- ****************************** -->
  <!-- Activity bound to BioSPI_Verify -->
  <!-- ****************************** -->
  <activity name="SPI_Verify">
    <input name="BSPHandle"/>
    <input name="MaxFMRRequested"/>
    <input name="ReferenceTemplate_Form"/>
    <input name="ReferenceTemplate_BIRHandle"/>
    <input name="Left"/>
    <input name="Right"/>
    <input name="Thumb"/>
    <input name="PointerFinger"/>
    <input name="MiddleFinger"/>
    <input name="RingFinger"/>
    <input name="LittleFinger"/>
    <input name="Multiple"/>
    <input name="no_AdaptedBIR"/>
    <input name="no_Result"/>
    <input name="no_FMRAchieved"/>
    <input name="no_Payload"/>
    <input name="Timeout"/>
    <input name="no_AuditData"/>
    <output name="AdaptedBIR"/>
    <output name="Result"/>
    <output name="FMRAchieved"/>
    <output name="Payload"/>
    <output name="AuditData"/>

    <!-- check API=SPI -->
    <assert_condition response_if_false="fail">
      <and>
        <equal_to var1="BSPHandle" var2="_bsphandle"/>
        <equal_to var1="MaxFMRRequested" var2="_maxfmrrequested"/>
        <equal_to var1="ReferenceTemplate_Form" var2="_referencetemplate_form"/>
        <equal_to var1="ReferenceTemplate_BIRHandle" var2="_referencetemplate_birhandle"/>
        <equal_to var1="Left" var2="_left"/>
        <equal_to var1="Right" var2="_right"/>
        <equal_to var1="Thumb" var2="_thumb"/>
        <equal_to var1="PointerFinger" var2="_pointerfinger"/>
        <equal_to var1="MiddleFinger" var2="_middlefinger"/>
        <equal_to var1="RingFinger" var2="_ringfinger"/>
        <equal_to var1="LittleFinger" var2="_littlefinger"/>
        <equal_to var1="Multiple" var2="_multiple"/>
        <equal_to var1="no_AdaptedBIR" var2="_no_adaptedbir"/>
        <equal_to var1="no_Result" var2="_no_result"/>
        <equal_to var1="no_FMRAchieved" var2="_no_fmrachieved"/>
        <equal_to var1="no_Payload" var2="_no_payload"/>
        <equal_to var1="Timeout" var2="_timeout"/>
        <equal_to var1="no_AuditData" var2="_no_auditdata"/>
      </and>
    </assert_condition>
  </activity>
</package>
```

8.29 Assertion 4.9 – BioAPI_Identify_And_BioSPI_Identify

Description: This assertion calls BioAPI_Identify with the input parameters described in Default Input Table and Test Condition Table, and checks if the framework under test returns BioAPI_OK or an error value in accordance with the description in Expected Result Table. The assertion is according to the following statement in the subclauses from BioAPI 2.0 standard document (ISO/IEC19784-1:2006), which are described in References in this subclause in this part of ISO/IEC 24709.

Excerpts:

Subclause 8.4.9

```
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);
```

This function captures biometric data from the attached device (sensor unit), and compares it against a set of reference BIRs (the Population).

The population that the match takes place against can be presented in one of two ways:

a) in a BIR database identified by an open database handle;
b) input in an array of BIRs;

NOTE When using a BSP-controlled BIR database, this database must previously have been opened using BioAPI_DbOpen.

There is an option to use an array of BIRs, which can be specified in BioAPI_IDENTIFY_POPULATION_TYPE in the BioAPI_IDENTIFY_POPULATION structure. If it is specified as BioAPI_PRESET_ARRAY_TYPE (3), the array of BIRs which had been previously set in the BioAPI_PresetIdentifyPopulation call will be used. The preset array of BIRs will be freed internally by the BSP when BioAPI_BSPDetach is called.

The application shall request a maximum FMR value criterion for a successful match.

The function performs the following actions (in order):

a) captures a sample and processes it as appropriate;
b) determines the set of candidates from the population that match according to the specified criteria;
c) allocates a memory block large enough to contain an array of elements of type BioAPI_CANDIDATE with as many elements as the number of candidates determined in (b);
d) fills the array with the candidate information for all candidates determined in (b), including the FMRAchieved of each candidate; and
e) returns the address of the array in the Candidates parameter and the size of the array in the NumberOfResults parameter.

NOTE See clause C.4 for information on the use of the FMR concept for normalized scoring and thresholding.

By default, the BSP is responsible for providing the user interface associated with the identify operation. The application may, however, request control of the GUI "look-and-feel" by providing a GUI callback pointer in BioAPI_SetGUICallbacks. See clause C.7 for additional explanation of user interface features.

Since the BioAPI_Identify operation includes a capture, it serializes use of the sensor device. If two or more biometric applications are racing for the sensor, the losers will wait until the operation completes or the timeout expires. This serialization takes place in all functions that capture data. The BSP is responsible for serializing. It may do this by either returning ‘busy’ (BioAPI_UNIT_IN_USE) or by queuing requests.

The memory block returned by this BioAPI function call shall be freed by the application using BioAPI_Free (see clause 8.7.2). Output BIRs can be retrieved by a call to BioAPI_GetBIRFromHandle, which releases the handle, or the handle can be released without retrieving the BIR through BioAPI_FreeBIRHandle.

A BioAPI_RETURN value indicates success or specifies a particular error condition. The value BioAPI_OK indicates success. All other values represent an error condition.

Subclause 9.3.4.9

```
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);
```

References: 8.4.9, 9.3.4.9

Scenario:

a) The testing application does the followings:

1) Initialises the Framework under test, then installs, loads and attaches the testing BSP.
2) Opens the database, and takes an bir database handle.
3) Calls BioAPI_Identify with the conditions given by the Default Input Table and by each row of the Test Condition Table.

b) The testing BSP does the followings:

1) Checks the parameters and the contents of the pointer variables as described in the column 'BioSPI function' in the Expected Result Table.
2) Sets the return value of the BioSPI_Identify given by each row of the Test Condition Table, then returns to the Framework.

c) The testing application does the followings:

1) Checks the return value of the BioAPI_Identify and if it is not as described in the Expected Result Table, then issues a fail conformity response.
2) Closes the database.
3) Detaches, unloads and uninstalls the testing BSP, then terminates the Framework under test.

Parameters: The input parameters described below table are used for this assertion.

Table 88 — Default Input Table for BioAPI_Identify_And_BioSPI_Identify

Number	Input parameter name	Input parameter value (underscore:invalid)
1	BSPHandle	*Valid BSPHandle*
2	MaxFMRRequested	__BioAPI_NOT_SET
3	Left	false
4	Right	false
5	Thumb	false
6	PointerFinger	false
7	MiddleFinger	false
8	RingFinger	false
9	LittleFinger	false
10	Multiple	false
11	Type	__BioAPI_DB_TYPE
12	BIRDataBase	*Valid DBHandle*
13	TotalNumberOfTemplates	0
14	Binning	false
15	MaxNumberOfResults	5
16	no_NumberOfResults	false
17	no_Candidates	false
18	Timeout	-1
19	no_AuditData	true

NOTE: These parameter names "Left","Right" ,"Thumb","PointerFinger","MiddleFinger","RingFinger","LittleFinger" and "Multiple" are based on the definitions in ISO/IEC 24709-1 9.2.6.

Table 89 — Test Condition Table for BioAPI_Identify_And_BioSPI_Identify

Test number	Input parameter name	Input parameter value (underscore:invalid)	Supported options in BSP Schema		Return value (from BioSPI)
			OPERATIONS_MASK	OPTIONS_MASK	
040901	BSPHandle	*Valid BSPHandle*	*Identify DatabaseOperation*	-	*OK*
040902	BSPHandle	*<u>Invalid BSPHandle</u>*	*Identify DatabaseOperation*	-	-
040903	MaxFMRRequested	0	*Identify DatabaseOperation*	-	*OK*
040904	MaxFMRRequested	50	*Identify DatabaseOperation*	-	*OK*
040905	MaxFMRRequested	100	*Identify DatabaseOperation*	-	*OK*

040906	Left	true	*Identify DatabaseOperation*	*SubtypeToCapture*	*OK*
040907	Left	true	*Identify DatabaseOperation*	-	*INVALID_DATA*
040908	Right	true	*Identify DatabaseOperation*	*SubtypeToCapture*	*OK*
040909	Right	true	*Identify DatabaseOperation*	-	*INVALID_DATA*
040910	(Left, Thumb)	(true, true)	*Identify DatabaseOperation*	*SubtypeToCapture*	*OK*
040911	(Left, Thumb)	(true, true)	*Identify DatabaseOperation*	-	*INVALID_DATA*
040912	(Left, PointerFinger)	(true, true)	*Identify DatabaseOperation*	*SubtypeToCapture*	*OK*
040913	(Left, PointerFinger)	(true, true)	*Identify DatabaseOperation*	-	*INVALID_DATA*
040914	(Left, MiddleFinger)	(true, true)	*Identify DatabaseOperation*	*SubtypeToCapture*	*OK*
040915	(Left, MiddleFinger)	(true, true)	*Identify DatabaseOperation*	-	*INVALID_DATA*
040916	(Left, RingFinger)	(true, true)	*Identify DatabaseOperation*	*SubtypeToCapture*	*OK*
040917	(Left, RingFinger)	(true, true)	*Identify DatabaseOperation*	-	*INVALID_DATA*
040918	(Left, LittleFinger)	(true, true)	*Identify DatabaseOperation*	*SubtypeToCapture*	*OK*
040919	(Left, LittleFinger)	(true, true)	*Identify DatabaseOperation*	-	*INVALID_DATA*
040920	(Right, Thumb)	(true, true)	*Identify DatabaseOperation*	*SubtypeToCapture*	*OK*
040921	(Right, Thumb)	(true, true)	*Identify DatabaseOperation*	-	*INVALID_DATA*
040922	(Right, PointerFinger)	(true, true)	*Identify DatabaseOperation*	*SubtypeToCapture*	*OK*
040923	(Right, PointerFinger)	(true, true)	*Identify DatabaseOperation*	-	*INVALID_DATA*
040924	(Right, MiddleFinger)	(true, true)	*Identify DatabaseOperation*	*SubtypeToCapture*	*OK*
040925	(Right, MiddleFinger)	(true, true)	*Identify DatabaseOperation*	-	*INVALID_DATA*
040926	(Right, RingFinger)	(true, true)	*Identify DatabaseOperation*	*SubtypeToCapture*	*OK*
040927	(Right, RingFinger)	(true, true)	*Identify DatabaseOperation*	-	*INVALID_DATA*
040928	(Right, LittleFinger)	(true, true)	*Identify DatabaseOperation*	*SubtypeToCapture*	*OK*
040929	(Right, LittleFinger)	(true, true)	*Identify DatabaseOperation*	-	*INVALID_DATA*
040930	Multiple	true	*Identify DatabaseOperation*	*SubtypeToCapture*	*OK*
040931	Multiple	true	*Identify DatabaseOperation*	-	*INVALID_DATA*
040932	Thumb	true	*Identify DatabaseOperation*	*SubtypeToCapture*	*OK*

040933	Thumb	true	*Identify DatabaseOperation*	-	*INVALID_DATA*
040934	PointerFinger	true	*Identify DatabaseOperation*	*SubtypeToCapture*	*OK*
040935	PointerFinger	true	*Identify DatabaseOperation*	-	*INVALID_DATA*
040936	MiddleFinger	true	*Identify DatabaseOperation*	*SubtypeToCapture*	*OK*
040937	MiddleFinger	true	*Identify DatabaseOperation*	-	*INVALID_DATA*
040938	RingFinger	true	*Identify DatabaseOperation*	*SubtypeToCapture*	*OK*
040939	RingFinger	true	*Identify DatabaseOperation*	-	*INVALID_DATA*
040940	LittleFinger	true	*Identify DatabaseOperation*	*SubtypeToCapture*	*OK*
040941	LittleFinger	true	*Identify DatabaseOperation*	-	*INVALID_DATA*
040942	(Left, Right, Thumb, PointerFinger, MiddleFinger, RingFinger, LittleFinger, Multiple)	(true, true, true, true, true, true, true, true)	*Identify DatabaseOperation*	-	*INVALID_DATA*
040943	Type	0	*Identify DatabaseOperation*	-	*INVALID_DATA*
040944	TotalNumberOfTemplates	10	*Identify DatabaseOperation*	-	*OK*
040945	Binning	true	*Identify DatabaseOperation*	-	*OK*
040946	MaxNumberOfResults	0	*Identify DatabaseOperation*	-	*OK*
040947	no_NumberOfResults	true	*Identify DatabaseOperation*	-	*INVALID_OUTPUT_POINTER*
040948	no_Candidates	true	*Identify DatabaseOperation*	-	*INVALID_OUTPUT_POINTER*
040949	Timeout	0	*Identify DatabaseOperation*	-	*TIMEOUT_EXPIRED*
040950	Timeout	DefaultIdentifyTimeout	*Identify DatabaseOperation*	-	*OK*
040951	Timeout	-2	*Identify DatabaseOperation*	-	*INVALID_DATA*
040952	no_AuditData	false	*Identify DatabaseOperation*	*Raw QualityRaw*	*OK*
040953	no_AuditData	false	*Identify DatabaseOperation*	-	*FUNCTION_NOT_SUPPORTED*
040954	BSPHandle	*Valid BSPHandle*	*DatabaseOperation*	-	-

NOTE1: For the detail of the cell of the test number row "040950" and the column "Parameter values(underscore:invalid)", see Table1.
NOTE2: These parameter names "Left","Right" ,"Thumb","PointerFinger","MiddleFinger","RingFinger","LittleFinger" and "Multiple" are based on the definitions in ISO/IEC 24709-1 9.2.6.

Table 90 — Expected Result Table for BioAPI_Identify_And_BioSPI_Identify

Test number	BioSPI function (BioAPI/BioSPI parameter check)	BioAPI function			Other conditions
		Return value	Output parameter name	Output parameter value	
040901	X	*OK*	-	-	-
040902	-	*indeterminate error*	-	-	-
040903	X	*OK*	-	-	-
040904	X	*OK*	-	-	-
040905	X	*OK*	-	-	-
040906	X	*OK*	-	-	-
040907	X	*indeterminate error*	-	-	-
040908	X	*OK*	-	-	-
040909	X	*indeterminate error*	-	-	-
040910	X	*OK*	-	-	-
040911	X	*indeterminate error*	-	-	-
040912	X	*OK*	-	-	-
040913	X	*indeterminate error*	-	-	-
040914	X	*OK*	-	-	-
040915	X	*indeterminate error*	-	-	-
040916	X	*OK*	-	-	-
040917	X	*indeterminate error*	-	-	-
040918	X	*OK*	-	-	-
040919	X	*indeterminate error*	-	-	-
040920	X	*OK*	-	-	-
040921	X	*indeterminate error*	-	-	-
040922	X	*OK*	-	-	-
040923	X	*indeterminate error*	-	-	-
040924	X	*OK*	-	-	-
040925	X	*indeterminate error*	-	-	-
040926	X	*OK*	-	-	-
040927	X	*indeterminate error*	-	-	-
040928	X	*OK*	-	-	-
040929	X	*indeterminate error*	-	-	-
040930	X	*OK*	-	-	-
040931	X	*indeterminate error*	-	-	-
040932	X	*OK*	-	-	-
040933	X	*indeterminate error*	-	-	-
040934	X	*OK*	-	-	-
040935	X	*indeterminate error*	-	-	-
040936	X	*OK*	-	-	-
040937	X	*indeterminate error*	-	-	-
040938	X	*OK*	-	-	-
040939	X	*indeterminate error*	-	-	-
040940	X	*OK*	-	-	-
040941	X	*indeterminate error*	-	-	-
040942	X	*indeterminate error*	-	-	-
040943	X	*indeterminate error*	-	-	-
040944	X	*OK*	-	-	-
040945	X	*OK*	-	-	-
040946	X	*OK*	-	-	-
040947	X	*indeterminate error*	-	-	-
040948	X	*indeterminate error*	-	-	-
040949	X	*TIMEOUT_EXPIRED*	-	-	-
040950	X	*OK*	-	-	-
040951	X	*indeterminate error*	-	-	-
040952	X	*OK*	-	-	-
040953	X	*indeterminate error*	-	-	-
040954	-	*indeterminate error*	-	-	-

Assertion language package

```
<?xml version="1.0" encoding="UTF-8"?>
<package name="30b63e80-792e-11de-8a39-0800200c9a66">
  <author>ISO/IEC JTC1 SC37</author>
  <description>This package contains the assertion "BioAPI_Identify_And_BioSPI_Identify".</description>

  <!-- ************** -->
  <!-- Test assertion -->
  <!-- ************** -->
  <assertion name="BioAPI_Identify_And_BioSPI_Identify" model="frameworkTesting">
    <description>
      This assertion checks if BioAPI_Identify is called with input parameters as described in Test Condition Table,
      the framework under test returns BioAPI_OK or error value in accordance with the description in Expected
      Result Table.
    </description>

    <!-- Parameter given by the CTS by referring to the Test Condition Table. -->
    <input name="_testnumber"/>
    <input name="_operationsmask1"/>
    <input name="_optionsmask1"/>
    <input name="_operationsmask2"/>
    <input name="_optionsmask2"/>
    <input name="_operationsmask3"/>
    <input name="_optionsmask3"/>
    <input name="_operationsmask4"/>
    <input name="_optionsmask4"/>
    <input name="_maxfmrrequested"/>
    <input name="_left"/>
    <input name="_right"/>
    <input name="_thumb"/>
    <input name="_pointerfinger"/>
    <input name="_middlefinger"/>
    <input name="_ringfinger"/>
    <input name="_littlefinger"/>
    <input name="_multiple"/>
    <input name="_totalnumberoftemplates"/>
    <input name="_binning"/>
    <input name="_maxnumberofresults"/>
    <input name="_no_numberofresults"/>
    <input name="_no_candidates"/>
    <input name="_timeout"/>
    <input name="_no_auditdata"/>

    <!-- Parameter given by the CTS by referring to the Expected Result Table. -->
    <input name="_expected_return_value"/>

    <!-- Parameter given by the CTS. -->
    <input name="_bspuuid"/>
    <input name="_dbuuid"/>
    <input name="_outputformatowner"/>
    <input name="_outputformattype"/>

    <!-- Invocation of the primary activity of this assertion. -->
    <invoke activity="BioAPI_Identify_And_BioSPI_Identify"/>

    <!-- Bind activities to check the BioSPI function invoked by the framework under test -->
    <bind activity="SPI_Identify" function="BioSPI_Identify"/>
  </assertion>

  <!-- **************** -->
  <!-- Primary activity -->
  <!-- **************** -->
  <activity name="BioAPI_Identify_And_BioSPI_Identify">

    <!-- Attach -->
    <invoke activity="AttachWithSomeOptions" package="fb6ff5b0-7d5e-11de-8a39-0800200c9a66" break_on_break="true">
      <input name="Operationsmask1" var="_operationsmask1"/>
      <input name="Optionmask1" var="_optionsmask1"/>
      <input name="Operationsmask2" var="_operationsmask2"/>
      <input name="Optionmask2" var="_optionsmask2"/>
      <input name="Operationsmask3" var="_operationsmask3"/>
      <input name="Optionmask3" var="_optionsmask3"/>
      <input name="Operationsmask4" var="_operationsmask4"/>
      <input name="Optionmask4" var="_optionsmask4"/>
      <input name="Bspuuid" var="_bspuuid"/>
      <output name="Bsphandle" setvar="_bsphandle"/>
    </invoke>

    <!-- Invoke the activity DB Open the framework under test -->
    <invoke activity="DbOpen" package="fb6ff5b0-7d5e-11de-8a39-0800200c9a66" break_on_break="true">
      <input name="Bsphandle" var="_bsphandle"/>
      <input name="Dbuuid" var="_dbuuid"/>
      <output name="Dbhandle" setvar="dbhandle"/>
      <output name="Markerhandle" setvar="markerhandle"/>
    </invoke>
    <set name="_type" var="__BioAPI_DB_TYPE"/>
    <set name="_birdatabase" var="dbhandle"/>

    <!-- Set an illegal value to the parameter. -->
    <add name="_bsphandle" value="1">
      <only_if><equal_to var1="_testnumber" value2="040902"/></only_if>
    </add>
    <set name="_type" value="0">
      <only_if><equal_to var1="_testnumber" value2="041103"/></only_if>
    </set>

    <!-- Invoke the function BioAPI_Identify. -->
    <invoke function="BioAPI_Identify">
      <input name="BSPHandle" var="_bsphandle"/>
```

```
      <input name="MaxFMRRequested" var="_maxfmrrequested"/>
      <input name="Left" var="_left"/>
      <input name="Right" var="_right"/>
      <input name="Thumb" var="_thumb"/>
      <input name="PointerFinger" var="_pointerfinger"/>
      <input name="MiddleFinger" var="_middlefinger"/>
      <input name="RingFinger" var="_ringfinger"/>
      <input name="LittleFinger" var="_littlefinger"/>
      <input name="Multiple" var="_multiple"/>
      <input name="Type" var="_type"/>
      <input name="BIRDataBase" var="_birdatabase"/>
      <input name="TotalNumberOfTemplates" var="_totalnumberoftemplates"/>
      <input name="Binning" var="_binning"/>
      <input name="MaxNumberOfResults" var="_maxnumberofresults"/>
      <input name="no_NumberOfResults" var="_no_numberofresults"/>
      <input name="no_Candidates" var="_no_candidates"/>
      <input name="Timeout" var="_timeout"/>
      <input name="no_AuditData" var="_no_auditdata"/>
      <output name="NumberOfResults" setvar="numberofresults"/>
      <output name="Candidate_1_Type" setvar="candidate_1_type"/>
      <output name="Candidate_1_BIRInDataBase" setvar="candidate_1_birindatabase"/>
      <output name="Candidate_1_BIRInArray" setvar="candidate_1_birinarray"/>
      <output name="Candidate_1_FMRAchieved" setvar="candidate_1_fmrachieved"/>
      <output name="AuditData" setvar="auditdata"/>
      <return setvar="return"/>
    </invoke>

    <!-- Extract the error code from the error value. -->
    <invoke activity="ExtractErrorCode" package="fb6ff5b0-7d5e-11de-8a39-0800200c9a66" break_on_break="true">
      <output name="Returnvalue" setvar="return"/>
    </invoke>

    <!-- Assertion -->
    <assert_condition response_if_false="fail">
      <description>
        The function BioAPI_IdentifyMatch has returned the expected return value.
      </description>
      <or>
        <!-- If the "_expected_return_value" parameter is 0xFFFFFFFF,
             then check the "return" parameter is not __BioAPI_OK. (error check only) -->
        <and>
          <equal_to var1="_expected_return_value" var2="_indeterminate_error"/>
          <not_equal_to var1="return" var2="__BioAPI_OK"/>
        </and>
        <and>
          <not_equal_to var1="_expected_return_value" var2="_indeterminate_error"/>
          <equal_to var1="return" var2="_expected_return_value"/>
        </and>
      </or>
    </assert_condition>

    <!-- If Test Number is 040902, make _bsphandle to an valid value. -->
    <subtract name="_bsphandle" value="1">
      <only_if><equal_to var1="_testnumber" value2="040902"/></only_if>
    </subtract>

    <!-- Invoke the functions BioAPI_DbClose -->
    <invoke activity="DbClose" package="fb6ff5b0-7d5e-11de-8a39-0800200c9a66" break_on_break="true">
      <input name="Bsphandle" var="_bsphandle"/>
      <input name="Dbhandle" var="dbhandle"/>
    </invoke>

    <!-- Invoke the functions Detach -->
    <invoke activity="BSPDetach" package="fb6ff5b0-7d5e-11de-8a39-0800200c9a66" break_on_break="true">
      <input name="Bsphandle" var="_bsphandle"/>
      <input name="Bspuuid" var="_bspuuid"/>
    </invoke>
  </activity>

  <!-- ********************************* -->
  <!-- Activity bound to BioSPI_Identify -->
  <!-- ********************************* -->
  <activity name="SPI_Identify">
    <input name="BSPHandle"/>
    <input name="MaxFMRRequested"/>
    <input name="Left"/>
    <input name="Right"/>
    <input name="Thumb"/>
    <input name="PointerFinger"/>
    <input name="MiddleFinger"/>
    <input name="RingFinger"/>
    <input name="LittleFinger"/>
    <input name="Multiple"/>
    <input name="Type"/>
    <input name="BIRDataBase"/>
    <input name="TotalNumberOfTemplates"/>
    <input name="Binning"/>
    <input name="MaxNumberOfResults"/>
    <input name="no_NumberOfResults"/>
    <input name="no_Candidates"/>
    <input name="Timeout"/>
    <input name="no_AuditData"/>
    <output name="NumberOfResults"/>
    <output name="Candidate_1_Type"/>
    <output name="Candidate_1_BIRInDataBase"/>
    <output name="Candidate_1_BIRInArray"/>
    <output name="Candidate_1_FMRAchieved"/>
    <output name="AuditData"/>
    <output name="return"/>

    <!-- check API=SPI -->
    <assert_condition response_if_false="fail">
      <and>
        <equal_to var1="BSPHandle" var2="_bsphandle"/>
        <equal_to var1="MaxFMRRequested" var2="_maxfmrrequested"/>
        <equal_to var1="Left" var2="_left"/>
        <equal_to var1="Right" var2="_right"/>
        <equal_to var1="Thumb" var2="_thumb"/>
        <equal_to var1="PointerFinger" var2="_pointerfinger"/>
        <equal_to var1="MiddleFinger" var2="_middlefinger"/>
        <equal_to var1="RingFinger" var2="_ringfinger"/>
        <equal_to var1="LittleFinger" var2="_littlefinger"/>
        <equal_to var1="Multiple" var2="_multiple"/>
        <equal_to var1="Type" var2="_type"/>
        <equal_to var1="BIRDataBase" var2="_birdatabase"/>
        <equal_to var1="TotalNumberOfTemplates" var2="_totalnumberoftemplates"/>
        <equal_to var1="Binning" var2="_binning"/>
        <equal_to var1="MaxNumberOfResults" var2="_maxnumberofresults"/>
        <equal_to var1="no_NumberOfResults" var2="_no_numberofresults"/>
        <equal_to var1="no_Candidates" var2="_no_candidates"/>
        <equal_to var1="Timeout" var2="_timeout"/>
        <equal_to var1="no_AuditData" var2="_no_auditdata"/>
      </and>
    </assert_condition>
  </activity>
</package>
```

8.30 Assertion 4.10 – BioAPI_Import_And_BioSPI_Import

Description: This assertion calls BioAPI_Import with the input parameters described in Default Input Table and Test Condition Table, and checks if the framework under test returns BioAPI_OK or an error value in accordance with the description in Expected Result Table. The assertion is according to the following statement in the subclauses from BioAPI 2.0 standard document (ISO/IEC19784-1:2006), which are described in References in this subclause in this part of ISO/IEC 24709.

Excerpts:

Subclause 8.4.10

```
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);
```

This function passes raw biometric data obtained by a biometric application by any means, and requests the specified BSP attach session to construct a BIR for the purpose specified. InputData identifies the memory buffer containing the raw biometric data, while InputFormat identifies the form of the raw biometric data. The InputFormats that a particular BSP will be prepared to accept are determined by the BSP (see the error __BioAPIERR_UNSUPPORTED_FORMAT). The function returns a handle to the ConstructedBIR. If the application needs to acquire the BIR either to store it in a database or to send it to a server, the application can retrieve the data with the BioAPI_GetBIRFromHandle function, or store it directly using BioAPI_DbStoreBIR.

The output ConstructedBIR can be retrieved by a call to BioAPI_GetBIRFromHandle, which releases the handle, or the handle can be released without retrieving the BIR through BioAPI_FreeBIRHandle.

A BioAPI_RETURN value indicates success or specifies a particular error condition. The value BioAPI_OK indicates success. All other values represent an error condition.

Subclause 9.3.4.10

```
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);
```

References: 8.4.10, 9.3.4.10

Scenario:

a) The testing application does the followings:

1) Initialises the Framework under test, then installs, loads and attaches the testing BSP.

2) Calls BioAPI_Import with the conditions given by the Default Input Table and by each row of the Test Condition Table.

b) The testing BSP does the followings:

1) Checks the parameters and the contents of the pointer variables as described in the column 'BioSPI function' in the Expected Result Table.

2) Sets the return value of the BioSPI_Import given by each row of the Test Condition Table, then returns to the Framework.

c) The testing application does the followings:

1) Checks the return value of the BioAPI_Import and if it is not as described in the Expected Result Table, then issues a fail conformity response.

2) Detaches, unloads and uninstalls the testing BSP, then terminates the Framework under test.

Parameters: The input parameters described below table are used for this assertion.

Table 91 — Default Input Table for BioAPI_Import_And_BioSPI_Import

Number	Input parameter name	Input parameter value (underscore:invalid)
1	BSPHandle	*Valid BSPHandle*
2	InputData	N/A
3	InputFormatOwner	N/A
4	InputFormatType	N/A
5	OutputFormatOwner	N/A
6	OutputFormatType	N/A
7	Purpose	__BioAPI_NO_PURPOSE_AVAILABLE
8	no_ConstructedBIR	false

Table 92 — Test Condition Table for BioAPI_Import_And_BioSPI_Import

Test number	Input parameter name	Input parameter value (underscore:invalid)	Supported options in BSP Schema		Return value (from BioSPI)
			OPERATIONS_MASK	OPTIONS_MASK	
041001	BSPHandle	*Valid BSPHandle*	*Import*	-	*OK*
041002	BSPHandle	*Invalid BSPHandle*	*Import*	-	-
041003	Purpose	__BioAPI_PURPOSE _VERIFY	*Import*	-	*OK*
041004	Purpose	__BioAPI_PURPOSE _IDENTIFY	*Import*	-	*OK*
041005	Purpose	__BioAPI_PURPOSE _ENROLL	*Import*	-	*OK*
041006	Purpose	__BioAPI_PURPOSE _ENROLL_FOR_VER IFICATION_ONLY	*Import*	-	*OK*
041007	Purpose	__BioAPI_PURPOSE _ENROLL_FOR_IDE NTIFICATION_ONLY	*Import*	-	*OK*
041008	Purpose	__BioAPI_PURPOSE _AUDIT	*Import*	-	*OK*
041009	Purpose	*Invalid Purpose*	*Import*	-	
041010	no_ConstructedBIR	true	*Import*	-	
041011	BSPHandle	*Valid BSPHandle*	-	-	-

Table 93 — Expected Result Table for BioAPI_Import_And_BioSPI_Import

Test number	BioSPI function (BioAPI/BioSPI parameter check)	BioAPI function			Other conditions
		Return value	Output parameter name	Output parameter value	
041001	X	*OK*	-	-	-
041002	-	*indeterminate error*	-	-	-
041003	X	*OK*	-	-	-
041004	X	*OK*	-	-	-
041005	X	*OK*	-	-	-
041006	X	*OK*	-	-	-
041007	X	*OK*	-	-	-
041008	X	*OK*	-	-	-
041009	X	*PURPOSE_NOT_SUPPORTED*	-	-	-
041010	X	*indeterminate error*	-	-	-
041011	-	*FUNCTION_NOT_SUPPORTED*	-	-	-

Assertion language package

```
<?xml version="1.0" encoding="UTF-8"?>
<package name="4019c220-792e-11de-8a39-0800200c9a66">
  <author>ISO/IEC JTC1 SC37</author>
  <description>This package contains the assertion "BioAPI_Import_And_BioSPI_Import".</description>

  <!-- ************** -->
  <!-- Test assertion -->
  <!-- ************** -->
  <assertion name="BioAPI_Import_And_BioSPI_Import" model="frameworkTesting">
    <description>
      This assertion checks if BioAPI_Import is called with input parameters as described in Test Condition Table,
      the framework under test returns BioAPI_OK or error value in accordance with the description in Expected
      Result Table.
    </description>

    <!-- Parameter given by the CTS by referring to the Test Condition Table. -->
    <input name="_testnumber"/>
    <input name="_operationsmask"/>
    <input name="_optionsmask"/>
    <input name="_inputdata"/>
    <input name="_inputformatowner"/>
    <input name="_inputformattype"/>
    <input name="_outputformatowner"/>
    <input name="_outputformattype"/>
    <input name="_purpose"/>
    <input name="_no_constructedbir"/>

    <!-- Parameter given by the CTS by referring to the Expected Result Table. -->
    <input name="_expected_return_value"/>

    <!-- Parameter given by the CTS. -->
    <input name="_bspuuid"/>

    <!-- Invocation of the primary activity of this assertion. -->
    <invoke activity="BioAPI_Import_And_BioSPI_Import"/>

    <!-- Bind activities to check the BioSPI function invoked by the framework under test -->
    <bind activity="SPI_Import" function="BioSPI_Import"/>
  </assertion>

  <!-- **************** -->
  <!-- Primary activity -->
  <!-- **************** -->
  <activity name="BioAPI_Import_And_BioSPI_Import">

    <!-- Invoke the activity BSP Attach the framework under test -->
    <invoke activity="BSPAttach" package="fb6ff5b0-7d5e-11de-8a39-0800200c9a66" break_on_break="true">
      <input name="Operationsmask" var="_operationsmask"/>
      <input name="Optionsmask" var="_optionsmask"/>
      <input name="Bspuuid" var="_bspuuid"/>
      <output name="Newbsphandle" setvar="_bsphandle"/>
    </invoke>

    <!-- If Test Number is 041002, make _bsphandle to an illegal value. -->
    <add name="_bsphandle" value="1">
      <only_if><equal_to var1="_testnumber" value2="041002"/></only_if>
    </add>

    <!-- Invoke the function BioAPI_Import. -->
    <invoke function="BioAPI_Import">
      <input name="BSPHandle" var="_bsphandle"/>
      <input name="InputData" var="_inputdata"/>
      <input name="InputFormatOwner" var="_inputformatowner"/>
      <input name="InputFormatType" var="_inputformattype"/>
      <input name="OutputFormatOwner" var="_outputformatowner"/>
      <input name="OutputFormatType" var="_outputformattype"/>
      <input name="Purpose" var="_purpose"/>
      <input name="no_ConstructedBIR" var="_no_constructedbir"/>
      <output name="ConstructedBIR" setvar="constructedbir"/>
      <return setvar="return"/>
    </invoke>

    <!-- Extract the error code from the error value. -->
    <invoke activity="ExtractErrorCode" package="fb6ff5b0-7d5e-11de-8a39-0800200c9a66" break_on_break="true">
      <output name="Returnvalue" setvar="return"/>
    </invoke>

    <!-- Assertion -->
    <assert_condition response_if_false="fail">
      <description>
        The function BioAPI_Import has returned the expected return value.
      </description>
      <or>
        <!-- If the "_expected_return_value" parameter is 0xFFFFFFFF,
             then check the "return" parameter is not __BioAPI_OK. (error check only) -->
        <and>
          <equal_to var1="_expected_return_value" var2="_indeterminate_error"/>
          <not_equal_to var1="return" var2="__BioAPI_OK"/>
        </and>
        <and>
          <not_equal_to var1="_expected_return_value" var2="_indeterminate_error"/>
          <equal_to var1="return" var2="_expected_return_value"/>
        </and>
      </or>
    </assert_condition>

    <!-- If Test Number is 041002, make _bsphandle to an valid value. -->
    <subtract name="_bsphandle" value="1">
      <only_if><equal_to var1="_testnumber" value2="041002"/></only_if>
    </subtract>

    <!-- Invoke the functions Detach -->
    <invoke activity="BSPDetach" package="fb6ff5b0-7d5e-11de-8a39-0800200c9a66" break_on_break="true">
      <input name="Bsphandle" var="_bsphandle"/>
      <input name="Bspuuid" var="_bspuuid"/>
    </invoke>
  </activity>

  <!-- ****************************** -->
  <!-- Activity bound to BioSPI_Import -->
  <!-- ****************************** -->
  <activity name="SPI_Import">
    <input name="BSPHandle"/>
    <input name="InputData"/>
    <input name="InputFormatOwner"/>
    <input name="InputFormatType"/>
    <input name="OutputFormatOwner"/>
    <input name="OutputFormatType"/>
    <input name="Purpose"/>
    <input name="no_ConstructedBIR"/>
    <output name="ConstructedBIR"/>
```

```
    <output name="return"/>

    <!-- check API=SPI -->
    <assert_condition response_if_false="fail">
      <and>
        <equal_to var1="BSPHandle" var2="_bsphandle"/>
        <equal_to var1="Inputdata" var2="_inputdata"/>
        <equal_to var1="InputFormatOwner" var2="_inputformatowner"/>
        <equal_to var1="InputFormatType" var2="_inputformattype"/>
        <equal_to var1="OutputFormatOwner" var2="_outputformatowner"/>
        <equal_to var1="OutputFormatType" var2="_outputformattype"/>
        <equal_to var1="Purpose" var2="_purpose"/>
        <equal_to var1="no_ConstructedBIR" var2="_no_constructedbir"/>
      </and>
    </assert_condition>
  </activity>
</package>
```

8.31 Assertion 4.11 – BioAPI_PresetIdentifyPopulation_And_BioSPI_PresetIdentifyPopulation

Description: This assertion calls BioAPI_PresetIdentifyPopulation with the input parameters described in Default Input Table and Test Condition Table, and checks if the framework under test returns BioAPI_OK or an error value in accordance with the description in Expected Result Table. The assertion is according to the following statement in the subclauses from BioAPI 2.0 standard document (ISO/IEC19784-1:2006), which are described in References in this subclause in this part of ISO/IEC 24709.

Excerpts:

Subclause 8.4.11

```
BioAPI_RETURN BioAPI BioAPI_PresetIdentifyPopulation
       (BioAPI_HANDLE                          BSPHandle,
       const BioAPI_IDENTIFY_POPULATION        *Population);
```

This function provides the population of BIRs to the BSP, as specified in the BSPHandle. The enrollment population to be matched against can be presented in one of two ways:

a) in a BIR database identified by an open database handle;
b) input in an array of BIRs;

The BSP allocates a memory block and transfers the population of BIRs to the memory block with a data format (not standardized) which is supported by the currently attached matching algorithm BioAPI Unit. After this function is called successfully, an application can call BioAPI_Identify or BioAPI_IdentifyMatch, specifying BioAPI_PRESET_ARRAY_TYPE in the BioAPI_IDENTIFY_POPULATION structure. The BSP keeps this memory block until another BioAPI_PresetIdentifyPopulation or BioAPI_BSPDetach is called.

A BioAPI_RETURN value indicates success or specifies a particular error condition. The value BioAPI_OK indicates success. All other values represent an error condition.

Subclause 9.3.4.11

```
BioAPI_RETURN BioAPI BioSPI_PresetIdentifyPopulation
       (BioAPI_HANDLE                          BSPHandle,
       const BioAPI_IDENTIFY_POPULATION        *Population);
```

References: 8.4.11, 9.3.4.11

Scenario:

a) The testing application does the followings:

1) Initialises the Framework under test, then installs, loads and attaches the testing BSP.
2) Opens the database to take an bir database handle.
3) Calls BioAPI_PresetIdentifyPopulation with the conditions given by the Default Input Table and by each row of the Test Condition Table.

b) The testing BSP does the followings:

1) Checks the parameters and the contents of the pointer variables as described in the column 'BioSPI function' in the Expected Result Table.
2) Sets the return value of the BioSPI_PresetIdentifyPopulation given by each row of the Test Condition Table, then returns to the Framework.

c) The testing application does the followings:

1) Checks the return value of the BioAPI_PresetIdentifyPopulation and if it is not as described in the Expected Result Table, then issues a fail conformity response.
2) Detaches, unloads and uninstalls the testing BSP, then terminates the Framework under test.

Parameters: The input parameters described below table are used for this assertion.

Table 94 — Default Input Table for BioAPI_PresetIdentifyPopulation_And_BioSPI_PresetIdentifyPopulation

Number	Input parameter name	Input parameter value (underscore:invalid)
1	BSPHandle	*Valid BSPHandle*
2	Type	__BioAPI_DB_TYPE
3	BIRDataBase	*Valid DBHandle*

Table 95 — Test Condition Table for BioAPI_PresetIdentifyPopulation_And_BioSPI_PresetIdentifyPopulation

Test number	Input parameter name	Input parameter value (underscore:invalid)	Supported options in BSP Schema		Return value (from BioSPI)
			OPERATIONS_MASK	OPTIONS_MASK	
041101	BSPHandle	*Valid BSPHandle*	*PresetIdentifyPopulation*	-	*OK*
041102	BSPHandle	*Invalid BSPHandle*	*PresetIdentifyPopulation*	-	-
041103	Type	0	*PresetIdentifyPopulation*	-	*INVALID_INPUT_POINTER*
041104	BSPHandle	*Valid BSPHandle*	-	-	-

Table 96 — Expected Result Table for BioAPI_PresetIdentifyPopulation_AndBioSPI_PresetIdentifyPopulation

Test number	BioSPI function (BioAPI/BioSPI parameter check)	BioAPI function			Other conditions
		Return value	Output parameter name	Output parameter value	
041101	X	*OK*	-	-	-
041102	-	*indeterminate error*	-	-	-
041103	X	*indeterminate error*	-	-	-
041104	-	*FUNCTION_NOT_SUPPORTED*	-	-	-

Assertion language package

```
<?xml version="1.0" encoding="UTF-8"?>
<package name="55440070-792e-11de-8a39-0800200c9a66">
  <author>ISO/IEC JTC1 SC37</author>
  <description>
   This package contains the assertion "BioAPI_PresetIdentifyPopulation_And_BioSPI_PresetIdentifyPopulation".
 </description>

  <!-- ************** -->
  <!-- Test assertion -->
  <!-- ************** -->
  <assertion name="BioAPI_PresetIdentifyPopulation_And_BioSPI_PresetIdentifyPopulation" model="frameworkTesting">
    <description>
      This assertion checks if BioAPI_PresetIdentifyPopulation is called with input parameters as described in Test
      Condition Table, the framework under test returns BioAPI_OK or error value in accordance with the description
      in Expected Result Table.
    </description>

    <!-- Parameter given by the CTS by referring to the Test Condition Table. -->
    <input name="_testnumber"/>
    <input name="_operationsmask"/>
    <input name="_optionsmask"/>

    <!-- Parameter given by the CTS by referring to the Expected Result Table. -->
    <input name="_expected_return_value"/>
    <!-- Parameter given by the CTS. -->
    <input name="_bspuuid"/>
    <input name="_dbuuid"/>

    <!-- Invocation of the primary activity of this assertion. -->
    <invoke activity="BioAPI_PresetIdentifyPopulation_And_BioSPI_PresetIdentifyPopulation"/>

    <!-- Bind activities to check the BioSPI function invoked by the framework under test -->
    <bind activity="SPI_PresetIdentifyPopulation" function="BioSPI_PresetIdentifyPopulation"/>
  </assertion>

  <!-- **************** -->
  <!-- Primary activity -->
  <!-- **************** -->
  <activity name="BioAPI_PresetIdentifyPopulation_And_BioSPI_PresetIdentifyPopulation">

    <!-- Invoke the activity BSP Attach the framework under test -->
    <invoke activity="BSPAttach" package="fb6ff5b0-7d5e-11de-8a39-0800200c9a66" break_on_break="true">
      <input name="Operationsmask" var="_operationsmask"/>
      <input name="Optionsmask" var="_optionsmask"/>
      <input name="Bspuuid" var="_bspuuid"/>
      <output name="Newbsphandle" setvar="_bsphandle"/>
    </invoke>

    <!-- Invoke the activity DbOpen -->
    <invoke activity="DbOpen" package="fb6ff5b0-7d5e-11de-8a39-0800200c9a66" break_on_break="true">
      <input name="Bsphandle" var="_bsphandle"/>
      <input name="Dbuuid" var="_dbuuid"/>
      <output name="Dbhandle" setvar="dbhandle"/>
      <output name="Markerhandle" setvar="markerhandle"/>
    </invoke>

    <!-- Set the Parameter group "Identify population". -->
    <set name="_birdatabase" var="dbhandle"/>
    <set name="_type" value="__BioAPI_DB_TYPE"/>

    <!-- Set an illegal value to the parameter. -->
    <add name="_bsphandle" value="1">
      <only_if><equal_to var1="_testnumber" value2="041102"/></only_if>
    </add>
    <set name="_type" value="0">
      <only_if><equal_to var1="_testnumber" value2="041103"/></only_if>
    </set>

    <!-- Invoke the function BioAPI_PresetIdentifyPopulation. -->
    <invoke function="BioAPI_PresetIdentifyPopulation">
      <input name="BSPHandle" var="_bsphandle"/>
      <input name="Type" var="_type"/>
      <input name="BIRDataBase" var="_birdatabase"/>
      <return setvar="return"/>
    </invoke>

    <!-- Extract the error code from the error value. -->
    <invoke activity="ExtractErrorCode" package="fb6ff5b0-7d5e-11de-8a39-0800200c9a66" break_on_break="true">
      <output name="Returnvalue" setvar="return"/>
    </invoke>

    <!-- Assertion -->
    <assert_condition response_if_false="fail">
      <description>
        The function BioAPI_PresetIdentifyPopulation has returned the expected return value.
      </description>
      <or>
        <!-- If the "_expected_return_value" parameter is 0xFFFFFFFF,
             then check the "return" parameter is not __BioAPI_OK. (error check only) -->
        <and>
          <equal_to var1="_expected_return_value" var2="_indeterminate_error"/>
          <not_equal_to var1="return" var2="__BioAPI_OK"/>
        </and>
        <and>
          <not_equal_to var1="_expected_return_value" var2="_indeterminate_error"/>
          <equal_to var1="return" var2="_expected_return_value"/>
        </and>
      </or>
    </assert_condition>

    <!-- If Test Number is 041102, make _bsphandle to an valid value. -->
    <subtract name="_bsphandle" value="1">
      <only_if>
        <equal_to var1="_testnumber" value2="041102"/>
      </only_if>
    </subtract>

    <!-- Invoke the functions Detach -->
    <invoke activity="BSPDetach" package="fb6ff5b0-7d5e-11de-8a39-0800200c9a66" break_on_break="true">
      <input name="Bsphandle" var="_bsphandle"/>
```

```
      <input name="Bspuuid" var="_bspuuid"/>
    </invoke>
  </activity>

  <!-- ************************************************ -->
  <!-- Activity bound to BioSPI_PresetIdentifyPopulation -->
  <!-- ************************************************ -->
  <activity name="SPI_PresetIdentifyPopulation">
    <input name="BSPHandle"/>
    <input name="Type"/>
    <input name="BIRDataBase"/>
    <output name="return"/>

    <!-- check API=SPI -->
    <assert_condition response_if_false="fail">
      <or>
        <not var="_noparamcheck"/>
        <and>
          <equal_to var1="BSPHandle" var2="_bsphandle"/>
          <equal_to var1="Type" var2="_type"/>
          <equal_to var1="BIRDataBase" var2="_birdatabase"/>
        </and>
      </or>
    </assert_condition>
  </activity>
</package>
```

8.32 Assertion 5.1 – BioAPI_DbOpen_And_BioSPI_DbOpen

Description: This assertion calls BioAPI_DbOpen with the input parameters described in Default Input Table and Test Condition Table, and checks if the framework under test returns BioAPI_OK or an error value in accordance with the description in Expected Result Table. The assertion is according to the following statement in the subclauses from BioAPI 2.0 standard document (ISO/IEC19784-1:2006), which are described in References in this subclause in this part of ISO/IEC 24709.

Excerpts:

Subclause 8.5.1

```
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);
```

This function opens a BIR database maintained by the currently attached archive of the identified BSP invocation, using the access mode specified by the *AccessRequest*. A new marker is created and set to point to the first record in the BIR database, and a handle for that marker is returned.

Some BSPs may only support a single BIR database or may have a preferred database. The application can allow the BSP to select the BIR database to open by using a NULL pointer value for the database UUID parameter.

A BioAPI_RETURN value indicates success or specifies a particular error condition. The value BioAPI_OK indicates success. All other values represent an error condition.

Subclause 9.3.5.1

```
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);
```

References: 8.5.1, 9.3.5.1

Scenario:

a) The testing application does the followings:

1) Initialises the Framework under test, then installs, loads and attaches the testing BSP.
2) Calls BioAPI_DbOpen with the conditions given by the Default Input Table and by each row of the Test Condition Table.

b) The testing BSP does the followings:

1) Checks the parameters and the contents of the pointer variables as described in the column 'BioSPI function' in the Expected Result Table.
2) Sets the return value of the BioSPI_DbOpen given by each row of the Test Condition Table, then returns to the Framework.

c) The testing application does the followings:

1) Checks the return value of the BioAPI_DbOpen and if it is not as described in the Expected Result Table, then issues a fail conformity response.
2) Closes the database.
3) Detaches, unloads and uninstalls the testing BSP, then terminates the Framework under test.

Parameters: The input parameters described below table are used for this assertion.

Table 97 — Default Input Table for BioAPI_DbOpen_And_BioSPI_DbOpen

Number	Input parameter name	Input parameter value (underscore:invalid)
1	BSPHandle	*Valid BSPHandle*
2	DbUuid	*Valid Uuid*
3	no_DbUuid	false
4	ReadAccess	true
5	WriteAccess	false
6	no_DbHandle	false
7	no_MarkerHandle	false

NOTE: The parameter name "no_DbUuid" is not defined in ISO/IEC 24709-1 so the implementation of this parameter depends on each CTS.

Table 98 — Test Condition Table for BioAPI_DbOpen_And_BioSPI_DbOpen

Test number	Input parameter name	Input parameter value (underscore:invalid)	Supported options in BSP Schema		Return value (from BioSPI)
			OPERATIONS_MASK	OPTIONS_MASK	
050101	BSPHandle	*Valid BSPHandle*	*DatabaseOperations*	-	*OK*
050102	BSPHandle	*Invalid BSPHandle*	*DatabaseOperations*	-	-
050103	no_DbUuid	true	*DatabaseOperations*	-	*OK*
050104	DbUuid	*Invalid Uuid*	*DatabaseOperations*	-	*INVALID_UUID*
050105	WriteAccess	true	*DatabaseOperations*	-	*OK*
050106	ReadAccess	false	*DatabaseOperations*	-	*INVALID_ACCESS_REQUEST*
050107	no_DbHandle	true	*DatabaseOperations*	-	*INVALID_OUTPUT_POINTER*
050108	no_MarkerHandle	true	*DatabaseOperations*	-	*INVALID_OUTPUT_POINTER*
050109	BSPHandle	*Valid BSPHandle*	-	-	-

NOTE: The parameter name "no_DbUuid" is not defined in ISO/IEC 24709-1 so the implementation of this parameter depends on each CTS.

Table 99 — Expected Result Table for BioAPI_DbOpen_And_BioSPI_DbOpen

Test number	BioSPI function (BioAPI/BioSPI parameter check)	BioAPI function			Other conditions
		Return value	Output parameter name	Output parameter value	
050101	X	*OK*	-	-	-
050102	-	*indeterminate error*	-	-	-
050103	X	*OK*	-	-	-
050104	X	*INVALID_UUID*	-	-	-
050105	X	*OK*	-	-	-
050106	X	*INVALID_ACCESS_REQUEST*	-	-	-
050107	X	*indeterminate error*	-	-	-
050108	X	*indeterminate error*	-	-	-
050109	-	*indeterminate error*	-	-	-

Assertion language package

```
<?xml version="1.0" encoding="UTF-8"?>
<package name="9f31c870-792e-11de-8a39-0800200c9a66">
  <author>ISO/IEC JTC1 SC37</author>
  <description>This package contains the assertion "BioAPI_DbOpen_And_BioSPI_DbOpen".</description>

  <!-- ************** -->
  <!-- Test assertion -->
  <!-- ************** -->
  <assertion name="BioAPI_DbOpen_And_BioSPI_DbOpen" model="frameworkTesting">
    <description>
      This assertion checks if BioAPI_DbOpen is called with input parameters as described in Test Condition Table,
      the framework under test returns BioAPI_OK or error value in accordance with the description in Expected
      Result Table.
    </description>

    <!-- Parameter given by the CTS by referring to the Test Condition Table. -->
    <input name="_testnumber"/>
    <input name="_operationsmask"/>
    <input name="_optionsmask"/>
    <input name="_dbuuid"/>
    <input name="_readaccess"/>
    <input name="_writeaccess"/>
    <input name="_no_dbhandle"/>
    <input name="_no_markerhandle"/>
    <input name="_no_dbuuid"/>

    <!-- Parameter given by the CTS by referring to the Expected Result Table. -->
    <input name="_expected_return_value"/>

    <!-- Parameter given by the CTS. -->
    <input name="_bspuuid"/>
    <input name="_dbuuid"/>

    <!-- Invocation of the primary activity of this assertion. -->
    <invoke activity="BioAPI_DbOpen_And_BioSPI_DbOpen"/>

    <!-- Bind activities to check the BioSPI function invoked by the framework under test -->
    <bind activity="SPI_DbOpen" function="BioSPI_DbOpen"/>
  </assertion>

  <!-- **************** -->
  <!-- Primary activity -->
  <!-- **************** -->
  <activity name="BioAPI_DbOpen_And_BioSPI_DbOpen">

    <!-- Invoke the activity BSP Attach the framework under test -->
    <invoke activity="BSPAttach" package="fb6ff5b0-7d5e-11de-8a39-0800200c9a66" break_on_break="true">
      <input name="Operationsmask" var="_operationsmask"/>
      <input name="Optionsmask" var="_optionsmask"/>
      <input name="Bspuuid" var="_bspuuid"/>
      <output name="Newbsphandle" setvar="_bsphandle"/>
    </invoke>

    <!-- If Test Number is 050102, make _bsphandle to an illegal value. -->
    <add name="_bsphandle" value="1">
      <only_if><equal_to var1="_testnumber" value2="050102"/></only_if>
    </add>

    <!-- Invoke the function BioAPI_DbOpen to open the specified database. -->
    <invoke function="BioAPI_DbOpen">
      <input name="BSPHandle" var="_bsphandle"/>
      <input name="DbUuid" var="_dbuuid"/>
      <input name="ReadAccess" var="_readaccess"/>
      <input name="WriteAccess" var="_writeaccess"/>
      <input name="no_DbHandle" var="_no_dbhandle"/>
      <input name="no_MarkerHandle" var="_no_markerhandle"/>
      <input name="no_DbUuid" var="_no_dbuuid"/>
      <output name="DbHandle" setvar="dbhandle"/>
      <output name="MarkerHandle" setvar="markerhandle"/>
      <return setvar="return"/>
    </invoke>

    <!-- Extract the error code from the error value. -->
    <invoke activity="ExtractErrorCode" package="fb6ff5b0-7d5e-11de-8a39-0800200c9a66" break_on_break="true">
      <output name="Returnvalue" setvar="return"/>
    </invoke>

    <!-- Assertion -->
    <assert_condition response_if_false="fail">
      <description>
        The function BioAPI_DbOpen has returned the expected return value.
      </description>
      <or>
        <!-- If the "_expected_return_value" parameter is 0xFFFFFFFF,
             then check the "return" parameter is not __BioAPI_OK. (error check only) -->
        <and>
          <equal_to var1="_expected_return_value" var2="_indeterminate_error"/>
          <not_equal_to var1="return" var2="__BioAPI_OK"/>
        </and>
        <and>
          <not_equal_to var1="_expected_return_value" var2="_indeterminate_error"/>
          <equal_to var1="return" var2="_expected_return_value"/>
        </and>
      </or>
    </assert_condition>

    <!-- If Test Number is 050102, make _bsphandle to an valid value. -->
    <subtract name="_bsphandle" value="1">
      <only_if><equal_to var1="_testnumber" value2="050102"/></only_if>
    </subtract>

    <!-- Invoke the functions BioAPI_DbClose -->
    <invoke activity="DbClose" package="fb6ff5b0-7d5e-11de-8a39-0800200c9a66" break_on_break="true">
      <input name="Bsphandle" var="_bsphandle"/>
      <input name="Dbhandle" var="_dbhandle"/>
    </invoke>

    <!-- Invoke the functions Detach -->
    <invoke activity="BSPDetach" package="fb6ff5b0-7d5e-11de-8a39-0800200c9a66" break_on_break="true">
      <input name="Bsphandle" var="_bsphandle"/>
      <input name="Bspuuid" var="_bspuuid"/>
    </invoke>
  </activity>

  <!-- ****************************** -->
  <!-- Activity bound to BioSPI_DbOpen -->
  <!-- ****************************** -->
  <activity name="SPI_DbOpen">
    <input name="BSPHandle"/>
    <input name="DbUuid"/>
    <input name="ReadAccess"/>
    <input name="WriteAccess"/>
    <input name="no_DbHandle"/>
    <input name="no_MarkerHandle"/>
    <input name="no_DbUuid"/>
    <output name="DbHandle"/>
    <output name="MarkerHandle"/>
    <output name="return"/>

    <!-- check API=SPI -->
    <assert_condition response_if_false="fail">
      <equal_to var1="BSPHandle" var2="_bsphandle"/>
      <equal_to var1="DbUuid" var2="_dbuuid"/>
      <equal_to var1="ReadAccess" var2="_readaccess"/>
      <equal_to var1="WriteAccess" var2="_writeaccess"/>
      <equal_to var1="no_DbHandle" var2="_no_dbhandle"/>
      <equal_to var1="no_MarkerHandle" var2="_no_markerhandle"/>
      <equal_to var1="no_DbUuid" var2="_no_dbuuid"/>
    </assert_condition>
  </activity>
</package>
```

8.33 Assertion 5.2 – BioAPI_DbClose_And_BioSPI_DbClose

Description: This assertion calls BioAPI_DbClose with the input parameters described in Default Input Table and Test Condition Table, and checks if the framework under test returns BioAPI_OK or an error value in accordance with the description in Expected Result Table. The assertion is according to the following statement in the subclauses from BioAPI 2.0 standard document (ISO/IEC19784-1:2006), which are described in References in this subclause in this part of ISO/IEC 24709.

Excerpts:

Subclause 8.5.2

```
BioAPI_RETURN BioAPI BioAPI_DbClose
        (BioAPI_HANDLE              BSPHandle,
        BioAPI_DB_HANDLE            DbHandle);
```

This function closes an open BIR database. All markers currently set for records in the database are freed and their handles become invalid.

NOTE A database opened in BioAPI_DB_ACCESS_WRITE mode can be damaged if it is left unclosed.

A BioAPI_RETURN value indicates success or specifies a particular error condition. The value BioAPI_OK indicates success. All other values represent an error condition.

Subclause 9.3.5.2

```
BioAPI_RETURN BioAPI BioSPI_DbClose
        (BioAPI_HANDLE          BSPHandle,
        BioAPI_DB_HANDLE        DbHandle);
```

References: 8.5.2, 9.3.5.2

Scenario:

a) The testing application does the followings:

1) Initialises the Framework under test, then installs, loads and attaches the testing BSP.

2) Opens the database.

3) Calls BioAPI_DbClose with the conditions given by the Default Input Table and by each row of the Test Condition Table.

b) The testing BSP does the followings:

1) Checks the parameters and the contents of the pointer variables as described in the column 'BioSPI function' in the Expected Result Table.

2) Sets the return value of the BioSPI_DbClose given by each row of the Test Condition Table, then returns to the Framework.

c) The testing application does the followings:

1) Checks the return value of the BioAPI_DbClose and if it is not as described in the Expected Result Table, then issues a fail conformity response.

2) Detaches, unloads and uninstalls the testing BSP, then terminates the Framework under test.

Parameters: The input parameters described below table are used for this assertion.

Table 100 — Default Input Table for BioAPI_DbClose_And_BioSPI_DbClose

Number	Input parameter name	Input parameter value (underscore:invalid)
1	BSPHandle	*Valid BSPHandle*
2	DbHandle	*Valid DBHandle*

Table 101 — Test Condition Table for BioAPI_DbClose_And_BioSPI_Dbclose

Test number	Input parameter name	Input parameter value (underscore:invalid)	Supported options in BSP Schema		Return value (from BioSPI)
			OPERATIONS_MASK	OPTIONS_MASK	
050201	BSPHandle	*Valid BSPHandle*	*DatabaseOperations*	-	*OK*
050202	BSPHandle	*<u>Invalid BSPHandle</u>*	*DatabaseOperations*	-	-
050203	DbHandle	*<u>Invalid DBHandle</u>*	*DatabaseOperations*	-	*INVALID_DB_HANDLE*
050204	BSPHandle	*Valid BSPHandle*	-	-	-

Table 102 — Expected Result Table for BioAPI_DbClose_And_BioSPI_DbClose

Test number	BioSPI function (BioAPI/BioSPI parameter check)	BioAPI function			Other conditions
		Return value	Output parameter name	Output parameter value	
050201	X	*OK*	-	-	-
050202	-	*indeterminate error*	-	-	-
050203	X	*indeterminate error*	-	-	-
050204	-	*indeterminate error*	-	-	-

Assertion language package

```
<?xml version="1.0" encoding="UTF-8"?>
<package name="aa8f2d20-792e-11de-8a39-0800200c9a66">
  <author>ISO/IEC JTC1 SC37</author>
  <description>This package contains the assertion "BioAPI_DbClose_And_BioSPI_DbClose".</description>

  <!-- ************** -->
  <!-- Test assertion -->
  <!-- ************** -->
  <assertion name="BioAPI_DbClose_And_BioSPI_DbClose" model="frameworkTesting">
    <description>
      This assertion checks if BioAPI_DbClose is called with input parameters as described in Test Condition Table,
      the framework under test returns BioAPI_OK or error value in accordance with the description in Expected
      Result Table.
    </description>

    <!-- Parameter given by the CTS by referring to the Test Condition Table. -->
    <input name="_testnumber"/>
    <input name="_operationsmask"/>
    <input name="_optionsmask"/>

    <!-- Parameter given by the CTS by referring to the Expected Result Table. -->
    <input name="_expected_return_value"/>

    <!-- Parameter given by the CTS. -->
    <input name="_bspuuid"/>
    <input name="_dbuuid"/>

    <!-- Invocation of the primary activity of this assertion. -->
    <invoke activity="BioAPI_DbClose_And_BioSPI_DbClose"/>

    <!-- Bind activities to check the BioSPI function invoked by the framework under test -->
    <bind activity="SPI_DbClose" function="BioSPI_DbClose"/>
  </assertion>

  <!-- **************** -->
  <!-- Primary activity -->
  <!-- **************** -->
  <activity name="BioAPI_DbClose_And_BioSPI_DbClose">

    <!-- Invoke the activity BSP Attach the framework under test -->
    <invoke activity="BSPAttach" package="fb6ff5b0-7d5e-11de-8a39-0800200c9a66" break_on_break="true">
      <input name="Operationsmask" var="_operationsmask"/>
      <input name="Optionsmask" var="_optionsmask"/>
      <input name="Bspuuid" var="_bspuuid"/>
      <output name="Newbsphandle" setvar="_bsphandle"/>
    </invoke>

    <!-- Invoke the activity DB Open the framework under test -->
    <invoke activity="DbOpen" package="fb6ff5b0-7d5e-11de-8a39-0800200c9a66" break_on_break="true">
      <input name="Bsphandle" var="_bsphandle"/>
      <input name="Dbuuid" var="_dbuuid"/>
      <output name="Dbhandle" setvar="_dbhandle"/>
      <output name="Markerhandle" setvar="_markerhandle"/>
    </invoke>

    <!-- Set an illegal value to the parameter. -->
    <add name="_bsphandle" value="1">
      <only_if><equal_to var1="_testnumber" value2="050202"/></only_if>
    </add>
    <add name="_dbhandle" value="1">
      <only_if><equal_to var1="_testnumber" value2="050203"/></only_if>
    </add>

    <!-- Invoke the function BioAPI_DbClose to open the specified database. -->
    <invoke function="BioAPI_DbClose">
      <input name="BSPHandle" var="_bsphandle"/>
      <input name="DbHandle" var="_dbhandle"/>
      <return setvar="return"/>
    </invoke>

    <!-- Extract the error code from the error value. -->
    <invoke activity="ExtractErrorCode" package="fb6ff5b0-7d5e-11de-8a39-0800200c9a66" break_on_break="true">
      <output name="Returnvalue" setvar="return"/>
    </invoke>

    <!-- Assertion -->
    <assert_condition response_if_false="fail">
      <description>
        The function BioAPI_DbClose has returned the expected return value.
      </description>
      <or>
        <!-- If the "_expected_return_value" parameter is 0xFFFFFFFF,
             then check the "return" parameter is not __BioAPI_OK. (error check only) -->
        <and>
```

```
          <equal_to var1="_expected_return_value" var2="_indeterminate_error"/>
          <not_equal_to var1="return" var2="__BioAPI_OK"/>
        </and>
        <and>
          <not_equal_to var1="_expected_return_value" var2="_indeterminate_error"/>
          <equal_to var1="return" var2="_expected_return_value"/>
        </and>
      </or>
    </assert_condition>

    <!-- If Test Number is 050202, make _bsphandle to an valid value. -->
    <subtract name="_bsphandle" value="1">
      <only_if><equal_to var1="_testnumber" value2="050202"/></only_if>
    </subtract>

    <!-- Invoke the functions Detach -->
    <invoke activity="BSPDetach" package="fb6ff5b0-7d5e-11de-8a39-0800200c9a66" break_on_break="true">
      <input name="Bsphandle" var="_bsphandle"/>
      <input name="Bspuuid" var="_bspuuid"/>
    </invoke>
  </activity>

  <!-- ******************************** -->
  <!-- Activity bound to BioSPI_DbClose -->
  <!-- ******************************** -->
  <activity name="SPI_DbClose">
    <input name="BSPHandle"/>
    <input name="DbHandle"/>
    <output name="return"/>

    <!-- check API=SPI -->
    <assert_condition response_if_false="fail">
      <and>
        <equal_to var1="BSPHandle" var2="_bsphandle"/>
        <equal_to var1="DbHandle" var2="_dbhandle"/>
      </and>
    </assert_condition>
  </activity>
</package>
```

8.34 Assertion 5.3 – BioAPI_DbCreate_And_BioSPI_DbCreate

Description: This assertion calls BioAPI_DbCreate with the input parameters described in Default Input Table and Test Condition Table, and checks if the framework under test returns BioAPI_OK or an error value in accordance with the description in Expected Result Table. The assertion is according to the following statement in the subclauses from BioAPI 2.0 standard document (ISO/IEC19784-1:2006), which are described in References in this subclause in this part of ISO/IEC 24709.

Excerpts:

Subclause 8.5.3

```
BioAPI_RETURN BioAPI BioAPI_DbCreate
       (BioAPI_HANDLE                  BSPHandle,
       const BioAPI_UUID               *DbUuid,
       uint32_t                        NumberOfRecords,
       BioAPI_DB_ACCESS_TYPE           AccessRequest,
       BioAPI_DB_HANDLE                *DbHandle);
```

This function creates and opens a new BIR database on the currently attached archive unit of the identified BSP invocation. The identification of the new database is specified by the input parameter DbUuid which shall be created by the biometric application, and shall be distinct from any current database UUID supported by that archive unit, whether currently open or not. The newly created BIR database is opened under the specified access mode.

NOTE This function is used create a new BIR database. It does not transport any information to the archive unit about the new database except its UUID and access conditions. There are archives which are only able to deal with static database sizes or where much higher effort will be needed to manage a database with dynamic size (e.g., smartcards storing templates in transparent or structured files, which might have a static size dependent on the characteristics of the smartcard operating system). Archives may need size information to create a new BIR database. Because the calling application might not have the knowledge about typical or maximum template sizes in bytes, the number of records to be stored at maximum in the database will be provided. Archives which are able to deal with database sizes dynamically may ignore the NumberOfRecords parameter.

A BioAPI_RETURN value indicates success or specifies a particular error condition. The value BioAPI_OK indicates success. All other values represent an error condition.

Subclause 9.3.5.3

```
BioAPI_RETURN BioAPI BioSPI_DbCreate
        (BioAPI_HANDLE                BSPHandle,
        const BioAPI_UUID             *DbUuid,
        uint32_t                      NumberOfRecords,
        BioAPI_DB_ACCESS_TYPE         AccessRequest,
        BioAPI_DB_HANDLE              *DbHandle);
```

References: 8.5.3, 9.3.5.3

Scenario:

a) The testing application does the followings:

1) Initialises the Framework under test, then installs, loads and attaches the testing BSP.
2) Calls BioAPI_DbCreate with the conditions given by the Default Input Table and by each row of the Test Condition Table.

b) The testing BSP does the followings:

1) Checks the parameters and the contents of the pointer variables as described in the column 'BioSPI function' in the Expected Result Table.
2) Sets the return value of the BioSPI_DbCreate given by each row of the Test Condition Table, then returns to the Framework.

c) The testing application does the followings:

1) Checks the return value of the BioAPI_DbCreate and if it is not as described in the Expected Result Table, then issues a fail conformity response.
2) Closes the database.
3) Detaches, unloads and uninstalls the testing BSP, then terminates the Framework under test.

Parameters: The input parameters described below table are used for this assertion.

Table 103 — Default Input Table for BioAPI_DbCreate_And_BioSPI_DbCreate

Number	Input parameter name	Input parameter value (underscore:invalid)
1	BSPHandle	*Valid BSPHandle*
2	DbUuid	*Valid Uuid*
3	no_DbUuid	false
4	NumberOfRecords	100
5	ReadAccess	true
6	WriteAccess	false
7	no_DbHandle	false

NOTE: The parameter name "no_DbUuid" is not defined in ISO/IEC 24709-1 so the implementation of this parameter depends on each CTS.

Table 104 — Test Condition Table for BioAPI_DbCreate_And_BioSPI_DbCreate

Test number	Input parameter name	Input parameter value (underscore:invalid)	Supported options in BSP Schema		Return value (from BioSPI)
			OPERATIONS_MASK	OPTIONS_MASK	
050301	BSPHandle	*Valid BSPHandle*	*DatabaseOperations*	-	*OK*
050302	BSPHandle	*Invalid BSPHandle*	*DatabaseOperations*	-	*INVALID_BSP_HANDLE*
050303	no_DbUuid	true	*DatabaseOperations*	-	*INVALID_UUID*
050304	NumberOfRecords	0	*DatabaseOperations*	-	*UNABLE_TO_CREATE_DATABASE*
050305	WriteAccess	true	*DatabaseOperations*	-	*OK*
050306	ReadAccess	false	*DatabaseOperations*	-	*INVALID_ACCESS_REQUEST*
050307	no_DbHandle	true	*DatabaseOperations*	-	*INVALID_OUTPUT_POINTER*
050308	BSPHandle	*Valid BSPHandle*	-	-	*FUNCTION_NOT_SUPPORTED*

NOTE: The parameter name "no_DbUuid" is not defined in ISO/IEC 24709-1 so the implementation of this parameter depends on each CTS.

Table 105 — Expected Result Table for BioAPI_DbCreate_And_BioSPI_DbCreate

Test number	BioSPI function (BioAPI/BioSPI parameter check)	BioAPI function			Other conditions
		Return value	Output parameter name	Output parameter value	
050301	X	*OK*	-	-	-
050302	-	*indeterminate error*	-	-	-
050303	X	*INVALID_UUID*	-	-	-
050304	X	*UNABLE_TO_CREATE_DATABASE*	-	-	-
050305	X	*OK*	-	-	-
050306	X	*INVALID_ACCESS_REQUEST*	-	-	-
050307	X	*indeterminate error*	-	-	-
050308	-	*indeterminate error*	-	-	-

Assertion language package

```
<?xml version="1.0" encoding="UTF-8"?>
<package name="b5f8ede0-792e-11de-8a39-0800200c9a66">
  <author>ISO/IEC JTC1 SC37</author>
  <description>This package contains the assertion "BioAPI_DbCreate_And_BioSPI_DbCreate".</description>

  <!-- ************** -->
  <!-- Test assertion -->
  <!-- ************** -->
  <assertion name="BioAPI_DbCreate_And_BioSPI_DbCreate" model="frameworkTesting">
    <description>
      This assertion checks if BioAPI_DbCreate is called with input parameters as described in Test Condition Table,
      the framework under test returns BioAPI_OK or error value in accordance with the description in Expected
      Result Table.
    </description>

    <!-- Parameter given by the CTS by referring to the Test Condition Table. -->
    <input name="_testnumber"/>
    <input name="_operationsmask"/>
    <input name="_optionsmask"/>
    <input name="_dbuuid"/>
    <input name="_numberofrecords"/>
    <input name="_readaccess"/>
    <input name="_writeaccess"/>
    <input name="_no_dbhandle"/>
    <input name="_no_dbuuid"/>

    <!-- Parameter given by the CTS by referring to the Expected Result Table. -->
    <input name="_expected_return_value"/>

    <!-- Parameter given by the CTS. -->
    <input name="_bspuuid"/>
    <input name="_dbuuid"/>

    <!-- Invocation of the primary activity of this assertion. -->
    <invoke activity="BioAPI_DbCreate_And_BioSPI_DbCreate"/>

    <!-- Bind activities to check the BioSPI function invoked by the framework under test -->
    <bind activity="SPI_DbCreate" function="BioSPI_DbCreate"/>
  </assertion>

  <!-- **************** -->
  <!-- Primary activity -->
  <!-- **************** -->
  <activity name="BioAPI_DbCreate_And_BioSPI_DbCreate">
    <!-- Invoke the activity BSP Attach the framework under test -->
    <invoke activity="BSPAttach" package="fb6ff5b0-7d5e-11de-8a39-0800200c9a66" break_on_break="true">
      <input name="Operationsmask" var="_operationsmask"/>
      <input name="Optionsmask" var="_optionsmask"/>
      <input name="Bspuuid" var="_bspuuid"/>
      <output name="Newbsphandle" setvar="_bsphandle"/>
    </invoke>

    <!-- If Test Number is 050302, make _bsphandle to an illegal value. -->
    <add name="_bsphandle" value="1">
      <only_if><equal_to var1="_testnumber" value2="050302"/></only_if>
    </add>

    <!-- Invoke the function BioAPI_DbCreate to open the specified database. -->
    <invoke function="BioAPI_DbCreate">
      <input name="BSPHandle" var="_bsphandle"/>
      <input name="DBUuid" var="_dbuuid"/>
      <input name="NumberOfRecords" var="_numberofrecords"/>
      <input name="ReadAccess" var="_readaccess"/>
      <input name="WriteAccess" var="_writeaccess"/>
      <input name="no_DbHandle" var="_no_dbhandle"/>
      <input name="no_DbUuid" var="_no_dbuuid"/>
      <output name="DbHandle" setvar="dbhandle"/>
      <return setvar="return"/>
    </invoke>

    <!-- Extract the error code from the error value. -->
    <invoke activity="ExtractErrorCode" package="fb6ff5b0-7d5e-11de-8a39-0800200c9a66" break_on_break="true">
      <output name="Returnvalue" setvar="return"/>
    </invoke>

    <!-- Assertion -->
    <assert_condition response_if_false="fail">
      <description>
        The function BioAPI_DbCreate has returned the expected return value.
      </description>
      <or>
        <!-- If the "_expected_return_value" parameter is 0xFFFFFFFF,
             then check the "return" parameter is not __BioAPI_OK. (error check only) -->
        <and>
          <equal_to var1="_expected_return_value" var2="_indeterminate_error"/>
          <not_equal_to var1="return" var2="__BioAPI_OK"/>
        </and>
        <and>
          <not_equal_to var1="_expected_return_value" var2="_indeterminate_error"/>
          <equal_to var1="return" var2="_expected_return_value"/>
        </and>
      </or>
    </assert_condition>

    <!-- If Test Number is 050302, make _bsphandle to an valid value. -->
    <subtract name="_bsphandle" value="1">
      <only_if><equal_to var1="_testnumber" value2="050302"/></only_if>
    </subtract>

    <!-- Invoke the functions BioAPI_DbClose -->
    <invoke activity="DbClose" package="fb6ff5b0-7d5e-11de-8a39-0800200c9a66" break_on_break="true">
      <input name="Bsphandle" var="_bsphandle"/>
      <input name="Dbhandle" var="dbhandle"/>
    </invoke>

    <!-- Invoke the functions Detach -->
    <invoke activity="BSPDetach" package="fb6ff5b0-7d5e-11de-8a39-0800200c9a66" break_on_break="true">
      <input name="Bsphandle" var="_bsphandle"/>
      <input name="Bspuuid" var="_bspuuid"/>
    </invoke>
  </activity>

  <!-- ********************************* -->
  <!-- Activity bound to BioSPI_DbCreate -->
  <!-- ********************************* -->
  <activity name="SPI_DbCreate">
    <input name="BSPHandle"/>
    <input name="DBUuid"/>
    <input name="NumberOfRecords"/>
    <input name="ReadAccess"/>
    <input name="WriteAccess"/>
    <input name="no_DbHandle"/>
    <input name="no_DbUuid"/>
    <output name="DbHandle"/>
    <output name="return"/>

    <!-- check API=SPI -->
    <assert_condition response_if_false="fail">
      <equal_to var1="BSPHandle" var2="_bsphandle"/>
      <equal_to var1="DbUuid" var2="_dbuuid"/>
```

```
      <equal_to var1="NumberOfRecords" var2="_numberofrecords"/>
      <equal_to var1="ReadAccess" var2="_readaccess"/>
      <equal_to var1="WriteAccess" var2="_writeaccess"/>
      <equal_to var1="no_DbHandle" var2="_no_dbhandle"/>
      <equal_to var1="no_DbUuid" var2="_no_dbuuid"/>
    </assert_condition>
  </activity>
</package>
```

8.35 Assertion 5.4 – BioAPI_DbDelete_And_BioSPI_DbDelete

Description: This assertion calls BioAPI_DbDelete with the input parameters described in Default Input Table and Test Condition Table, and checks if the framework under test returns BioAPI_OK or an error value in accordance with the description in Expected Result Table. The assertion is according to the following statement in the subclauses from BioAPI 2.0 standard document (ISO/IEC19784-1:2006), which are described in References in this subclause in this part of ISO/IEC 24709.

Excerpts:

Subclause 8.5.4

```
BioAPI_RETURN BioAPI BioAPI_DbDelete
       (BioAPI_HANDLE              BSPHandle,
       const BioAPI_UUID           *DbUuid);
```

This function deletes all records from the specified BIR database and removes all state information associated with that database.

A BioAPI_RETURN value indicates success or specifies a particular error condition. The value BioAPI_OK indicates success. All other values represent an error condition.

Subclause 9.3.5.4

```
BioAPI_RETURN BioAPI BioSPI_DbDelete
       (BioAPI_HANDLE              BSPHandle,
       const BioAPI_UUID           *DbUuid);
```

References: 8.5.4, 9.3.5.4

Scenario:

a) The testing application does the followings:
   1) Initialises the Framework under test, then installs, loads and attaches the testing BSP.
   2) Creates and closes the database.
   3) Calls BioAPI_DbDelete with the conditions given by the Default Input Table and by each row of the Test Condition Table.

b) The testing BSP does the followings:
   1) Checks the parameters and the contents of the pointer variables as described in the column 'BioSPI function' in the Expected Result Table.
   2) Sets the return value of the BioSPI_DbDelete given by each row of the Test Condition Table, then returns to the Framework.

c) The testing application does the followings:
   1) Checks the return value of the BioAPI_DbDelete and if it is not as described in the Expected Result Table, then issues a fail conformity response.
   2) Detaches, unloads and uninstalls the testing BSP, then terminates the Framework under test.

Parameters: The input parameters described below table are used for this assertion.

Table 106 — Default Input Table for BioAPI_DbDelete_And_BioSPI_DbDelete

Number	Input parameter name	Input parameter value (underscore:invalid)
1	BSPHandle	*Valid BSPHandle*
2	DbUuid	*Valid Uuid*
3	no_DbUuid	false

NOTE: The parameter name "no_DbUuid" is not defined in ISO/IEC 24709-1 so the implementation of this parameter depends on each CTS.

Table 107 — Test Condition Table for BioAPI_DbDelete_And_BioSPI_DbDelete

Test number	Input parameter name	Input parameter value (underscore:invalid)	Supported options in BSP Schema		Return value (from BioSPI)
			OPERATIONS_MASK	OPTIONS_MASK	
050401	BSPHandle	*Valid BSPHandle*	*DatabaseOperations*	-	*OK*
050402	BSPHandle	<u>*Invalid BSPHandle*</u>	*DatabaseOperations*	-	-
050403	no_DbUuid	<u>true</u>	*DatabaseOperations*	-	*INVALID_UUID*
050404	DbUuid	<u>*Invalid Uuid*</u>	*DatabaseOperations*	-	*INVALID_UUID*
050405	BSPHandle	*Valid BSPHandle*	-	-	-

NOTE: The parameter name "no_DbUuid" is not defined in ISO/IEC 24709-1 so the implementation of this parameter depends on each CTS.

Table 108 — Expected Reusut Table for BioAPI_DbDelete_And_BioSPI_DbDelete

Test number	BioSPI function (BioAPI/BioSPI parameter check)	BioAPI function			Other conditions
		Return value	Output parameter name	Output parameter value	
050401	X	*OK*	-	-	-
050402	-	*indeterminate error*	-	-	-
050403	X	*INVALID_UUID*	-	-	-
050404	X	*INVALID_UUID*	-	-	-
050405	-	*indeterminate error*	-	-	-

Assertion language package

```
<?xml version="1.0" encoding="UTF-8"?>
<package name="c60b0100-792e-11de-8a39-0800200c9a66">
  <author>ISO/IEC JTC1 SC37</author>
  <description>This package contains the assertion "BioAPI_DbDelete_And_BioSPI_DbDelete".</description>

  <!-- ************** -->
  <!-- Test assertion -->
  <!-- ************** -->
  <assertion name="BioAPI_DbDelete_And_BioSPI_DbDelete" model="frameworkTesting">
    <description>
      This assertion checks if BioAPI_DbDelete is called with input parameters as described in Test Condition Table,
      the framework under test returns BioAPI_OK or error value in accordance with the description in Expected
      Result Table.
    </description>

    <!-- Parameter given by the CTS by referring to the Test Condition Table. -->
    <input name="_testnumber"/>
    <input name="_operationsmask"/>
    <input name="_optionsmask"/>
    <input name="_dbuuid"/>

    <!-- Parameter given by the CTS by referring to the Expected Result Table. -->
    <input name="_expected_return_value"/>

    <!-- Parameter given by the CTS. -->
    <input name="_bspuuid"/>

    <!-- Invocation of the primary activity of this assertion. -->
    <invoke activity="BioAPI_DbDelete_And_BioSPI_DbDelete"/>

    <!-- Bind activities to check the BioSPI function invoked by the framework under test -->
    <bind activity="SPI_DbDelete" function="BioSPI_DbDelete"/>
  </assertion>

  <!-- **************** -->
  <!-- Primary activity -->
  <!-- **************** -->
  <activity name="BioAPI_DbDelete_And_BioSPI_DbDelete">

    <!-- Invoke the activity BSP Attach the framework under test -->
    <invoke activity="BSPAttach" package="fb6ff5b0-7d5e-11de-8a39-0800200c9a66" break_on_break="true">
      <input name="Operationsmask" var="_operationsmask"/>
      <input name="Optionsmask" var="_optionsmask"/>
      <input name="Bspuuid" var="_bspuuid"/>
      <output name="Newbsphandle" setvar="_bsphandle"/>
    </invoke>

    <!-- Invoke the activity DB Create the framework under test -->
    <invoke activity="DbCreate" package="fb6ff5b0-7d5e-11de-8a39-0800200c9a66" break_on_break="true">
      <input name="Bsphandle" var="_bsphandle"/>
      <input name="Dbuuid" var="_dbuuid"/>
      <output name="dbhandle" setvar="_dbhandle"/>
    </invoke>

    <!-- Invoke the activity DB Close the framework under test -->
    <invoke activity="DbClose" package="fb6ff5b0-7d5e-11de-8a39-0800200c9a66" break_on_break="true">
      <input name="Bsphandle" var="_bsphandle"/>
      <input name="Dbhandle" var="_dbhandle"/>
    </invoke>

    <!-- If Test Number is 050402, make _bsphandle to an illegal value. -->
    <add name="_bsphandle" value="1">
      <only_if><equal_to var1="_testnumber" value2="050402"/></only_if>
    </add>

    <!-- Invoke the function BioAPI_DbDelete to open the specified database. -->
    <invoke function="BioAPI_DbDelete">
      <input name="BSPHandle" var="_bsphandle"/>
      <input name="DbUuid" var="_dbuuid"/>
      <return setvar="return"/>
    </invoke>

    <!-- Extract the error code from the error value. -->
    <invoke activity="ExtractErrorCode" package="fb6ff5b0-7d5e-11de-8a39-0800200c9a66" break_on_break="true">
      <output name="Returnvalue" setvar="return"/>
    </invoke>

    <!-- Assertion -->
    <assert_condition response_if_false="fail">
      <description>
        The function BioAPI_DbDelete has returned the expected return value.
      </description>
      <or>
        <!-- If the "_expected_return_value" parameter is 0xFFFFFFFF,
             then check the "return" parameter is not __BioAPI_OK. (error check only) -->
        <and>
          <equal_to var1="_expected_return_value" var2="_indeterminate_error"/>
          <not_equal_to var1="return" var2="__BioAPI_OK"/>
        </and>
        <and>
          <not_equal_to var1="_expected_return_value" var2="_indeterminate_error"/>
          <equal_to var1="return" var2="_expected_return_value"/>
        </and>
      </or>
    </assert_condition>

    <!-- If Test Number is 050402, make _bsphandle to an valid value. -->
    <subtract name="_bsphandle" value="1">
      <only_if><equal_to var1="_testnumber" value2="050402"/></only_if>
    </subtract>

    <!-- Invoke the functions Detach -->
    <invoke activity="BSPDetach" package="fb6ff5b0-7d5e-11de-8a39-0800200c9a66" break_on_break="true">
      <input name="Bsphandle" var="_bsphandle"/>
      <input name="Bspuuid" var="_bspuuid"/>
    </invoke>
  </activity>

  <!-- ********************************* -->
  <!-- Activity bound to BioSPI_DbDelete -->
  <!-- ********************************* -->
  <activity name="SPI_DbDelete">
    <input name="BSPHandle"/>
    <input name="DbUuid"/>
    <output name="return"/>

    <!-- check API=SPI -->
    <assert_condition response_if_false="fail">
      <equal_to var1="BSPHandle" var2="_bsphandle"/>
      <equal_to var1="DbUuid" var2="_dbuuid"/>
    </assert_condition>
  </activity>
</package>
```

8.36 Assertion 5.5 – BioAPI_DbSetMarker_And_BioSPI_DbSetMarker

Description: This assertion calls BioAPI_DbSetMarker with the input parameters described in Default Input Table and Test Condition Table, and checks if the framework under test returns BioAPI_OK or an error value in accordance with the description in Expected Result Table. The assertion is according to the following statement in the subclauses from BioAPI 2.0 standard document (ISO/IEC19784-1:2006), which are described in References in this subclause in this part of ISO/IEC 24709.

Excerpts:

Subclause 8.5.5

```
BioAPI_RETURN BioAPI BioAPI_DbSetMarker
        (BioAPI_HANDLE                      BSPHandle,
        BioAPI_DB_HANDLE                    DbHandle,
        const BioAPI_UUID                   *KeyValue,
        BioAPI_DB_MARKER_HANDLE             MarkerHandle);
```

The marker identified by the MarkerHandle is set to point to the record indicated by the KeyValue in the BIR database identified by the DbHandle. A NULL value will set the marker to the first record in the database.

NOTE When an error occurs, the position of the marker does not change.

A BioAPI_RETURN value indicates success or specifies a particular error condition. The value BioAPI_OK indicates success. All other values represent an error condition.

Subclause 9.3.5.5

```
BioAPI_RETURN BioAPI BioSPI_DbSetMarker
        (BioAPI_HANDLE                      BSPHandle,
        BioAPI_DB_HANDLE                    DbHandle,
        const BioAPI_UUID                   *KeyValue,
        BioAPI_DB_MARKER_HANDLE             MarkerHandle);
```

References: 8.5.5, 9.3.5.5

Scenario:

a) The testing application does the followings:

1) Initialises the Framework under test, then installs, loads and attaches the testing BSP.

2) Opens the database.

3) Calls BioAPI_DbSetMarker with the conditions given by the Default Input Table and by each row of the Test Condition Table.

b) The testing BSP does the followings:

1) Checks the parameters and the contents of the pointer variables as described in the column 'BioSPI function' in the Expected Result Table.

2) Sets the return value of the BioSPI_DbSetMarker given by each row of the Test Condition Table, then returns to the Framework.

c) The testing application does the followings:

1) Checks the return value of the BioAPI_DbSetMarker and if it is not as described in the Expected Result Table, then issues a fail conformity response.

2) Closes the database.

3) Detaches, unloads and uninstalls the testing BSP, then terminates the Framework under test.

Parameters: The input parameters described below table are used for this assertion.

Table 109 — Default Input Table for BioAPI_DbSetMarker_And_BioSPI_DbSetMarker

Number	Input parameter name	Input parameter value (underscore:invalid)
1	BSPHandle	*Valid BSPHandle*
2	DbHandle	*Valid DBHandle*
3	KeyValue	*Valid KeyValue*
4	MarkerHandle	*Valid MarkerHandle*

Table 110 — Test Condition Table for BioAPI_DbSetMarker_And_BioSPI_DbSetMarker

Test number	Input parameter name	Input parameter value (underscore:invalid)	Supported options in BSP Schema		Return value (from BioSPI)
			OPERATIONS_MASK	OPTIONS_MASK	
050501	BSPHandle	*Valid BSPHandle*	*DatabaseOperations*	-	*OK*
050502	BSPHandle	*Invalid BSPHandle*	*DatabaseOperations*	-	-
050503	DbHandle	*Invalid DBHandle*	*DatabaseOperations*	-	*IINVALID_DB_HANDLE*
050504	KeyValue	*Invalid KeyValue*	*DatabaseOperations*	-	*OK*
050505	MarkerHandle	*Invalid MarkerHandle*	*DatabaseOperations*	-	*MARKER_HANDLE_IS_INVALID*
050506	BSPHandle	*Valid BSPHandle*	-	-	-

Table 111 — Expected Result Table for BioAPI_DbSetMarker_And_BioSPI_DbSetMarker

Test number	BioSPI function (BioAPI/BioSPI parameter check)	BioAPI function			Other conditions
		Return value	Output parameter name	Output parameter value	
050501	X	*OK*	-	-	-
050502	-	*indeterminate error*	-	-	-
050503	X	*IINVALID_DB_HANDLE*	-	-	-
050504	X	*OK*	-	-	-
050505	X	*indeterminate error*	-	-	-
050506	-	*indeterminate error*	-	-	-

Assetion language package

```
<?xml version="1.0" encoding="UTF-8"?>
<package name="d06aa4c0-792e-11de-8a39-0800200c9a66">
  <author>ISO/IEC JTC1 SC37</author>
  <description>This package contains the assertion "BioAPI_DbSetMarker_And_BioSPI_DbSetMarker".</description>

  <!-- ************** -->
  <!-- Test assertion -->
  <!-- ************** -->
  <assertion name="BioAPI_DbSetMarker_And_BioSPI_DbSetMarker" model="frameworkTesting">
    <description>
      This assertion checks if BioAPI_DbSetmarker is called with input parameters as described in Test Condition
      Table, the framework under test returns BioAPI_OK or error value in accordance with the description in
      Expected Result Table.
    </description>

    <!-- Parameter given by the CTS by referring to the Test Condition Table. -->
    <input name="_testnumber"/>
    <input name="_operationsmask"/>
    <input name="_optionsmask"/>
    <input name="_keyvalue"/>

    <!-- Parameter given by the CTS by referring to the Expected Result Table. -->
    <input name="_expected_return_value"/>

    <!-- Parameter given by the CTS. -->
    <input name="_bspuuid"/>
    <input name="_dbuuid"/>

    <!-- Invocation of the primary activity of this assertion. -->
    <invoke activity="BioAPI_DbSetMarker_And_BioSPI_DbSetMarker"/>

    <!-- Bind activities to check the BioSPI function invoked by the framework under test -->
    <bind activity="SPI_DbSetMarker" function="BioSPI_DbSetMarker"/>
  </assertion>

  <!-- **************** -->
  <!-- Primary activity -->
  <!-- **************** -->
  <activity name="BioAPI_DbSetMarker_And_BioSPI_DbSetMarker">
```

```
    <!-- Invoke the activity BSP Attach the framework under test -->
    <invoke activity="BSPAttach" package="fb6ff5b0-7d5e-11de-8a39-0800200c9a66" break_on_break="true">
      <input name="Operationsmask" var="_operationsmask"/>
      <input name="Optionsmask" var="_optionsmask"/>
      <input name="Bspuuid" var="_bspuuid"/>
      <output name="Newbsphandle" setvar="_bsphandle"/>
    </invoke>

    <!-- Invoke the activity DB Open the framework under test -->
    <invoke activity="DbOpen" package="fb6ff5b0-7d5e-11de-8a39-0800200c9a66" break_on_break="true">
      <input name="Bsphandle" var="_bsphandle"/>
      <input name="Dbuuid" var="_dbuuid"/>
      <output name="Dbhandle" setvar="_dbhandle"/>
      <output name="Markerhandle" setvar="_markerhandle"/>
    </invoke>

    <!-- Set an illegal value to the parameter. -->
    <add name="_bsphandle" value="1">
      <only_if><equal_to var1="_testnumber" value2="050502"/></only_if>
    </add>
    <add name="_dbhandle" value="1">
      <only_if><equal_to var1="_testnumber" value2="050503"/></only_if>
    </add>
    <add name="_keyvalue" value="1">
      <only_if><equal_to var1="_testnumber" value2="050504"/></only_if>
    </add>
    <add name="_markerhandle" value="1">
      <only_if><equal_to var1="_testnumber" value2="050505"/></only_if>
    </add>

    <!-- Invoke the function BioAPI_DbSetMarker to open the specified database. -->
    <invoke function="BioAPI_DbSetMarker">
      <input name="BSPHandle" var="_bsphandle"/>
      <input name="DbHandle" var="_dbhandle"/>
      <input name="KeyValue" var="_keyvalue"/>
      <input name="MarkerHandle" var="_markerhandle"/>
      <return setvar="return"/>
    </invoke>

    <!-- Extract the error code from the error value. -->
    <invoke activity="ExtractErrorCode" package="fb6ff5b0-7d5e-11de-8a39-0800200c9a66" break_on_break="true">
      <output name="Returnvalue" setvar="return"/>
    </invoke>

    <!-- Assertion -->
    <assert_condition response_if_false="fail">
      <description>
        The function BioAPI_DbSetMarker has returned the expected return value.
      </description>
      <or>
        <!-- If the "_expected_return_value" parameter is 0xFFFFFFFF,
             then check the "return" parameter is not __BioAPI_OK. (error check only) -->
        <and>
          <equal_to var1="_expected_return_value" var2="_indeterminate_error"/>
          <not_equal_to var1="return" var2="__BioAPI_OK"/>
        </and>
        <and>
          <not_equal_to var1="_expected_return_value" var2="_indeterminate_error"/>
          <equal_to var1="return" var2="_expected_return_value"/>
        </and>
      </or>
    </assert_condition>

    <!-- Set an valid value to the parameter. -->
    <subtract name="_bsphandle" value="1">
      <only_if><equal_to var1="_testnumber" value2="050502"/></only_if>
    </subtract>
    <subtract name="_dbhandle" value="1">
      <only_if><equal_to var1="_testnumber" value2="050503"/></only_if>
    </subtract>

    <!-- Invoke the activity DB Close the framework under test -->
    <invoke activity="DbClose" package="fb6ff5b0-7d5e-11de-8a39-0800200c9a66" break_on_break="true">
      <input name="Bsphandle" var="_bsphandle"/>
      <input name="Dbhandle" var="_dbhandle"/>
    </invoke>

    <!-- Invoke the functions Detach -->
    <invoke activity="BSPDetach" package="fb6ff5b0-7d5e-11de-8a39-0800200c9a66" break_on_break="true">
      <input name="Bsphandle" var="_bsphandle"/>
      <input name="Bspuuid" var="_bspuuid"/>
    </invoke>
  </activity>

  <!-- ************************************ -->
  <!-- Activity bound to BioSPI_DbSetMarker -->
  <!-- ************************************ -->
  <activity name="SPI_DbSetMarker">
    <input name="BSPHandle"/>
    <input name="DbHandle"/>
    <input name="KeyValue"/>
    <input name="MarkerHandle"/>
    <output name="return"/>

    <!-- check API=SPI -->
    <assert_condition response_if_false="fail">
      <equal_to var1="BSPHandle" var2="_bsphandle"/>
      <equal_to var1="DbHandle" var2="_dbhandle"/>
      <equal_to var1="KeyValue" var2="_keyvalue"/>
      <equal_to var1="MarkerHandle" var2="_markerhandle"/>
    </assert_condition>
  </activity>
</package>
```

8.37 Assertion 5.6 – BioAPI_DbFreeMarker_And_BioSPI_DbFreeMarker

Description: This assertion calls BioAPI_DbFreeMarker with the input parameters described in Default Input Table and Test Condition Table, and checks if the framework under test returns BioAPI_OK or an error value in accordance with the description in Expected Result Table. The assertion is according to the following statement in the subclauses from BioAPI 2.0 standard document (ISO/IEC19784-1:2006), which are described in References in this subclause in this part of ISO/IEC 24709.

Excerpts:

Subclause 8.5.6

```
BioAPI_RETURN BioAPI BioAPI_DbFreeMarker
       (BioAPI_HANDLE                    BSPHandle,
       BioAPI_DB_MARKER_HANDLE           MarkerHandle);
```

Frees memory and resources associated with the specified marker and invalidates the MarkerHandle.

A BioAPI_RETURN value indicates success or specifies a particular error condition. The value BioAPI_OK indicates success. All other values represent an error condition.

Subclause 9.3.5.6

```
BioAPI_RETURN BioAPI BioSPI_DbFreeMarker
       (BioAPI_HANDLE                    BSPHandle,
       BioAPI_DB_MARKER_HANDLE           MarkerHandle);
```

References: 8.5.6, 9.3.5.6

Scenario:

a) The testing application does the followings:

1) Initialises the Framework under test, then installs, loads and attaches the testing BSP.
2) Opens the database.
3) Calls BioAPI_DbFreeMarker with the conditions given by the Default Input Table and by each row of the Test Condition Table.

b) The testing BSP does the followings:

1) Checks the parameters and the contents of the pointer variables as described in the column 'BioSPI function' in the Expected Result Table.
2) Sets the return value of the BioSPI_DbFreeMarker given by each row of the Test Condition Table, then returns to the Framework.

c) The testing application does the followings:

1) Checks the return value of the BioAPI_DbFreeMarker and if it is not as described in the Expected Result Table, then issues a fail conformity response.
2) Closes the database.

3) Detaches, unloads and uninstalls the testing BSP, then terminates the Framework under test.

Parameters: The input parameters described below table are used for this assertion.

Table 112 — Default Input Table for BioAPI_DbFreeMarker_And_BioSPI_DbFreeMarker

Number	Input parameter name	Input parameter value (underscore:invalid)
1	BSPHandle	*Valid BSPHandle*
2	MarkerHandle	*Valid MarkerHandle*

Table 113 — Test Condition Table for BioAPI_DbFreeMarker_And_BioSPI_DbFreeMarker

Test number	Input parameter name	Input parameter value (underscore:invalid)	Supported options in BSP Schema		Return value (from BioSPI)
			OPERATIONS_MASK	OPTIONS_MASK	
050601	BSPHandle	*Valid BSPHandle*	*DatabaseOperations*	-	*OK*
050602	BSPHandle	*Invalid BSPHandle*	*DatabaseOperations*	-	-
050603	MarkerHandle	*Invalid MarkerHandle*	*DatabaseOperations*	-	*MARKER_HANDLE_IS_INVALID*
050604	BSPHandle	*Valid BSPHandle*	-	-	-

Table 114 — Expected Result Table for BioAPI_DbFreeMarker_And_BioSPI_DbFreeMarker

Test number	BioSPI function (BioAPI/BioSPI parameter check)	BioAPI function			Other conditions
		Return value	Output parameter name	Output parameter value	
050601	X	*OK*	-	-	-
050602	-	*indeterminate error*	-	-	-
050603	X	*MARKER_HANDLE_IS_INVALID*	-	-	-
050604	-	*indeterminate error*	-	-	-

Assertion language package

```
<?xml version="1.0" encoding="UTF-8"?>
<package name="e4647960-792e-11de-8a39-0800200c9a66">
  <author>ISO/IEC JTC1 SC37</author>
  <description>This package contains the assertion "BioAPI_DbFreeMarker_And_BioSPI_DbFreeMarker".</description>

  <!-- ************** -->
  <!-- Test assertion -->
  <!-- ************** -->
  <assertion name="BioAPI_DbFreeMarker_And_BioSPI_DbFreeMarker" model="frameworkTesting">
    <description>
      This assertion checks if BioAPI_DbFreeMarker is called with input parameters as described in Test Condition
      Table, the framework under test returns BioAPI_OK or error value in accordance with the description in
      Expected Result Table.
    </description>

    <!-- Parameter given by the CTS by referring to the Test Condition Table. -->
    <input name="_testnumber"/>
    <input name="_operationsmask"/>
    <input name="_optionsmask"/>

    <!-- Parameter given by the CTS by referring to the Expected Result Table. -->
    <input name="_expected_return_value"/>

    <!-- Parameter given by the CTS. -->
    <input name="_bspuuid"/>
    <input name="_dbuuid"/>

    <!-- Invocation of the primary activity of this assertion. -->
    <invoke activity="BioAPI_DbFreeMarker_And_BioSPI_DbFreeMarker"/>

    <!-- Bind activities to check the BioSPI function invoked by the framework under test -->
    <bind activity="SPI_DbFreeMarker" function="BioSPI_DbFreeMarker"/>
  </assertion>

  <!-- **************** -->
  <!-- Primary activity -->
  <!-- **************** -->
  <activity name="BioAPI_DbFreeMarker_And_BioSPI_DbFreeMarker">

    <!-- Invoke the activity BSP Attach the framework under test -->
    <invoke activity="BSPAttach" package="fb6ff5b0-7d5e-11de-8a39-0800200c9a66" break_on_break="true">
      <input name="Operationsmask" var="_operationsmask"/>
      <input name="Optionsmask" var="_optionsmask"/>
      <input name="Bspuuid" var="_bspuuid"/>
      <output name="Newbsphandle" setvar="_bsphandle"/>
    </invoke>

    <!-- Invoke the activity DB Open the framework under test -->
    <invoke activity="DbOpen" package="fb6ff5b0-7d5e-11de-8a39-0800200c9a66" break_on_break="true">
      <input name="Bsphandle" var="_bsphandle"/>
      <input name="Dbuuid" var="_dbuuid"/>
      <output name="Dbhandle" setvar="_dbhandle"/>
      <output name="Markerhandle" setvar="_markerhandle"/>
    </invoke>

    <!-- Set an illegal value to the parameter. -->
    <add name="_bsphandle" value="1">
      <only_if><equal_to var1="_testnumber" value2="050602"/></only_if>
    </add>
    <add name="_markerhandle" value="1">
      <only_if><equal_to var1="_testnumber" value2="050603"/></only_if>
    </add>

    <!-- Invoke the function BioAPI_DbFreeMarker to open the specified database. -->
    <invoke function="BioAPI_DbFreeMarker">
      <input name="BSPHandle" var="_bsphandle"/>
      <input name="MarkerHandle" var="_markerhandle"/>
      <return setvar="return"/>
    </invoke>

    <!-- Extract the error code from the error value. -->
    <invoke activity="ExtractErrorCode" package="fb6ff5b0-7d5e-11de-8a39-0800200c9a66" break_on_break="true">
      <output name="Returnvalue" setvar="return"/>
    </invoke>

    <!-- Assertion -->
    <assert_condition response_if_false="fail">
      <description>
        The function BioAPI_DbFreeMarker has returned the expected return value.
      </description>
      <or>
        <!-- If the "_expected_return_value" parameter is 0xFFFFFFFF,
             then check the "return" parameter is not __BioAPI_OK. (error check only) -->
        <and>
          <equal_to var1="_expected_return_value" var2="_indeterminate_error"/>
          <not_equal_to var1="return" var2="__BioAPI_OK"/>
        </and>
        <and>
          <not_equal_to var1="_expected_return_value" var2="_indeterminate_error"/>
          <equal_to var1="return" var2="_expected_return_value"/>
        </and>
      </or>
    </assert_condition>

    <!-- Set an valid value to the parameter. -->
    <subtract name="_bsphandle" value="1">
      <only_if><equal_to var1="_testnumber" value2="050602"/></only_if>
    </subtract>

    <!-- Invoke the activity DB Close the framework under test -->
    <invoke activity="DbClose" package="fb6ff5b0-7d5e-11de-8a39-0800200c9a66" break_on_break="true">
      <input name="Bsphandle" var="_bsphandle"/>
      <input name="Dbhandle" var="_dbhandle"/>
    </invoke>

    <!-- Invoke the functions Detach -->
    <invoke activity="BSPDetach" package="fb6ff5b0-7d5e-11de-8a39-0800200c9a66" break_on_break="true">
      <input name="Bsphandle" var="_bsphandle"/>
      <input name="Bspuuid" var="_bspuuid"/>
    </invoke>
  </activity>

  <!-- ************************************ -->
  <!-- Activity bound to BioSPI_DbFreeMarker -->
  <!-- ************************************ -->
  <activity name="SPI_DbFreeMarker">
    <input name="BSPHandle"/>
    <input name="MarkerHandle"/>
    <output name="return"/>

    <!-- check API=SPI -->
    <assert_condition response_if_false="fail">
      <equal_to var1="BSPHandle" var2="_bsphandle"/>
      <equal_to var1="MarkerHandle" var2="_markerhandle"/>
    </assert_condition>
  </activity>
</package>
```

8.38 Assertion 5.7 – BioAPI_DbStoreBIR_And_BioSPI_DbStoreBIR

Description: This assertion calls BioAPI_DbStoreBIR with the input parameters described in Default Input Table and Test Condition Table, and checks if the framework under test returns BioAPI_OK or an error value in accordance with the description in Expected Result Table. The assertion is according to the following statement in the subclauses from BioAPI 2.0 standard document (ISO/IEC19784-1:2006), which are described in References in this subclause in this part of ISO/IEC 24709.

Excerpts:

Subclause 8.5.7

```
BioAPI_RETURN BioAPI BioAPI_DbStoreBIR
        (BioAPI_HANDLE                  BSPHandle,
        const BioAPI_INPUT_BIR          *BIRToStore,
        BioAPI_DB_HANDLE                DbHandle,
        BioAPI_UUID                     *BirUuid);
```

The BIR identified by the BIRToStore parameter is stored in the open BIR database identified by the DbHandle parameter. If the BIRToStore is identified by a BIR Handle, the input BIR Handle is freed. If the BIRToStore is identified by a database key value, the BIR is retrieved and stored in (copied to) the open database. A new UUID is assigned to the new BIR in the database, and this UUID can be used as a key value to access the BIR later.

A BioAPI_RETURN value indicates success or specifies a particular error condition. The value BioAPI_OK indicates success. All other values represent an error condition.

Subclause 9.3.5.7

```
BioAPI_RETURN BioAPI BioSPI_DbStoreBIR
        (BioAPI_HANDLE                  BSPHandle,
        const BioAPI_INPUT_BIR          *BIRToStore,
        BioAPI_DB_HANDLE                DbHandle,
        BioAPI_UUID                     *BirUuid);
```

References: 8.5.7, 9.3.5.7

Scenario:

a) The testing application does the followings:
   1) Initialises the Framework under test, then installs, loads and attaches the testing BSP.
   2) Opens the database.
   3) Enrolls to obtain a BIR handle.
   4) Calls BioAPI_DbStoreBIR with the conditions given by the Default Input Table and by each row of the Test Condition Table.

b) The testing BSP does the followings:
   1) Checks the parameters and the contents of the pointer variables as described in the column 'BioSPI function' in the Expected Result Table.
   2) Sets the return value of the BioSPI_DbStoreBIR given by each row of the Test Condition Table, then returns to the Framework.

c) The testing application does the followings:
   1) Checks the return value of the BioAPI_DbStoreBIR and if it is not as described in the Expected Result Table, then issues a fail conformity response.
   2) Closes the database.
   3) Detaches, unloads and uninstalls the testing BSP, then terminates the Framework under test.

Parameters: The input parameters described below table are used for this assertion.

Table 115 — Default Input Table for BioAPI_DbStoreBIR_And_BioSPI_DbStoreBIR

Number	Input parameter name	Input parameter value (underscore:invalid)
1	BSPHandle	*Valid BSPHandle*
2	BIRToStore_Form	2
3	BIRToStore_BIRHandle	*Valid BIRHandle*
4	DbHandle	*Valid DBHandle*
5	no_BirUuid	false

Table 116 — Test Condition Table for BioAPI_DbStoreBIR_And_BioSPI_DbStoreBIR

Test number	Input parameter name	Input parameter value (underscore:invalid)	Supported options in BSP Schema		Return value (from BioSPI)
			OPERATIONS_MASK	OPTIONS_MASK	
050701	BSPHandle	*Valid BSPHandle*	*DatabaseOperations Enroll*	-	*OK*
050702	BSPHandle	*Invalid BSPHandle*	*DatabaseOperations Enroll*	-	-
050703	BIRToStore_Form	0	*DatabaseOperations Enroll*	-	*INVALID_INPUT_POINTER*
050704	BIRToStore_BIRHandle	*Valid BIRHandle*	*DatabaseOperations Enroll*	-	*OK*
050705	BIRToStore_BIRHandle	*Invalid BIRHandle*	*DatabaseOperations Enroll*	-	*INVALID_BIR_HANDLE*
050706	DbHandle	*Invalid DBHandle*	*DatabaseOperations Enroll*	-	*INVALID_DB_HANDLE*
050707	no_BirUuid	true	*DatabaseOperations Enroll*	-	*INVALID_OUTPUT_POINTER*
050708	BSPHandle	*Valid BSPHandle*	*Enroll*	-	-

Table 117 — Expected Result Table for BioAPI_DbStoreBIR_And_BioSPI_DbStoreBIR

Test number	BioSPI function (BioAPI/BioSPI parameter check)	BioAPI function			Other conditions
		Return value	Output parameter name	Output parameter value	
050701	X	*OK*	-	-	-
050702	-	*indeterminate error*	-	-	-
050703	X	*indeterminate error*	-	-	-
050704	X	*OK*	-	-	-
050705	X	*indeterminate error*	-	-	-
050706	X	*INVALID_DB_HANDLE*	-	-	-
050707	X	*indeterminate error*	-	-	-
050708	-	*indeterminate error*	-	-	-

Assertion language package

```
<?xml version="1.0" encoding="UTF-8"?>
<package name="f2614110-792e-11de-8a39-0800200c9a66">
  <author>ISO/IEC JTC1 SC37</author>
  <description>This package contains the assertion "BioAPI_DbStoreBIR_And_BioSPI_DbStoreBIR".</description>

  <!-- ************** -->
  <!-- Test assertion -->
  <!-- ************** -->
  <assertion name="BioAPI_DbStoreBIR_And_BioSPI_DbStoreBIR" model="frameworkTesting">
    <description>
      This assertion checks if BioAPI_DbStoreBIR is called with input parameters as described in Test Condition
      Table, the framework under test returns BioAPI_OK or error value in accordance with the description in
      Expected Result Table.
    </description>

    <!-- Parameter given by the CTS by referring to the Test Condition Table. -->
    <input name="_testnumber"/>
    <input name="_operationsmask1"/>
    <input name="_optionsmask1"/>
    <input name="_operationsmask2"/>
    <input name="_optionsmask2"/>
    <input name="_operationsmask3"/>
    <input name="_optionsmask3"/>
    <input name="_operationsmask4"/>
    <input name="_optionsmask4"/>
    <input name="_no_biruuid"/>

    <!-- Parameter given by the CTS by referring to the Expected Result Table. -->
    <input name="_expected_return_value"/>
```