INTERNATIONAL STANDARD

ISO/IEC 24709-2

First edition
2007-02-15

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

Part 2:
Test assertions for biometric service providers

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

Partie 2: Assertions d'essai pour les fournisseurs de services biométriques

Reference number
ISO/IEC 24709-2:2007(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 24709-2 was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information technology*, Subcommittee SC 37, *Biometrics*.

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

— *Part 1: Methods and procedures*

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

The following parts are under preparation:

— *Part 3: Test assertions for BioAPI frameworks*

— *Part 4: Test assertions for biometric applications*

Introduction

This part of ISO/IEC 24709 defines a number of test assertions written in the assertion language specified in ISO/IEC 24709-1. These assertions enable a user of this part of ISO/IEC 24709 (such as a testing laboratory) to test the conformance to ISO/IEC 19784-1 (BioAPI 2.0) of any biometric service provider (BSP) 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 BSPs of all conformance subclasses. The assertions are further organized according to conformance subclasses (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 lists test assertions to be used in the conformance testing model for BioAPI BSPs, with specific provisions as follows:

Clause 7.1 contains descriptions of the BioAPI Conformity Statement and the test assertions.

Clause 7.2 contains specific provisions for BSPs of subclass "Verification BSP".

Clause 7.3 contains specific provisions for BSPs of subclass "Identification BSP".

Clause 7.4 contains specific provisions for BSPs of subclass "Capture BSP".

Clause 7.5 contains specific provisions for BSPs of subclass "Verification Engine".

Clause 7.6 contains specific provisions for BSPs of subclass "Identification Engine".

Clause 8 specifies the assertions to be used in the conformance testing model for BioAPI BSPs, for all conformance subclasses of BSPs (see ISO/IEC 19784-1:2006, A.4).

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

Part 2: Test assertions for biometric service providers

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.

This part of ISO/IEC 24709 specifies what subset of all the test assertions defined herein are to be executed for each of the five conformance subclasses of BSPs defined in ISO/IEC 19784-1 (BioAPI 2.0). It also specifies additional assertions that are to be executed depending on the optional features of BioAPI 2.0 that the implementation under test claims to support.

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). Biometric service provider implementations that are tested according to the methodology specified in ISO/IEC 24709-1 and with the 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 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 24709-1 apply.

5 Symbols and abbreviated terms

For the purposes of this document, the symbols and abbreviated terms defined in ISO/IEC 24709-1 apply.

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, 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, which is part of that methodology.

6.2 An important concept of the conformance testing methodology specified in ISO/IEC 24709-1 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; and

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 BSPs. This part of ISO/IEC 24709 is not concerned with the conformance testing models for BioAPI applications and frameworks and corresponding test assertions.

7 Testing the conformance of BioAPI BSPs

7.1 General

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

Table 1 — BioAPI Conformity Statement

Vendor 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	
BSP (this alternative shall be selected)	**X**
Conformance subclass of the BSP (one of the following):	■
verification BSP	
identification BSP	
capture BSP	
verification engine	
identification engine	
Optional functions supported by the BSP (zero or more of the following):	■
BioSPI_Capture	
BioSPI_CreateTemplate	
BioSPI_Process	
BioSPI_VerifyMatch	
BioSPI_Enroll	
BioSPI_Verify	
BioSPI_Identify	
Optional features supported by the BSP (zero or more of the following):	■
BSP-controlled database	
template adaptation	
generation of SOURCE PRESENT events	
setting of the Quality field in an intermediate BIR	
setting of the Quality field in a processed BIR	
Additional information:	■
UUID of the BSP	
BioAPI specification version	
Product version	
Supported BDB formats (one or more format owner / format type pairs)	
Supported biometric types (one or more)	
Maximum supported size for the payload	

7.1.2 The following table lists test assertions that shall be used to determine whether the implementation under test (a BioAPI BSP) fails to satisfy the requirements specified in ISO/IEC 19784-1:2006, A.4.

NOTE Successful processing of all applicable test assertions is prima facie evidence that the implementation satisfies the applicable requirements of ISO/IEC 19784-1, but does not establish this, as the assertions are not (and cannot be) an exhaustive test of conformance (see also ISO/IEC 24709-1:2007, Clause 6).

7.1.3 These assertions use the conformance testing model for BioAPI BSPs (see ISO 24709-1:2007, Clause 6), and are specified in Clause 8 of this Part of ISO/IEC 24709.

7.1.4 The following subclauses 7.2 to 7.6 contain specific provisions for each conformance subclass of BSP, and specify which of the test assertions specified below to be processed in order to perform conformance testing of a BSP of a given subclass.

Table 2 — Test assertions for BioAPI BSP

Number	Assertion name	References	Purpose
1a	*BioSPI_BSPLoad_InvalidUUID*	**9.3.1.1, 11.2.3**	Verify if calling the function BioSPI_BSPLoad with an invalid input parameter UUID returns BioAPIERR_H_FRAMEWORK_INVALID_UUID.
1b	*BioSPI_BSPLoad_ValidParam*	**9.3.1.1, A.4**	Verify if calling BioSPI_BSPLoad with valid input parameters returns BioAPI_OK.
2a	*BioSPI_BSPUnload_ValidParam*	**9.3.1.2, A.4**	Verify if calling BioSPI_BSPUnload with valid input parameters returns BioAPI_OK.
2b	*BioSPI_BSPUnload_InvalidUUID*	**9.3.1.2, 11.2.3**	Verify if calling BioSPI_BSPUnload with an invalid input parameter UUID returns BioAPIERR_INVALID_UUID.
2c	*BioSPI_BSPUnload_UnmatchedLoad*	**9.3.1.2, 8.1.6.1**	Verify if calling BioSPI_BSPUnload without a matching call to BioSPI_BSPLoad returns BioAPIERR_BSP_NOT_LOADED.
2d	*BioSPI_BSPUnload_Confirm*	**9.3.1.2**	Verify if calling BioSPI_BSPUnload truly unloads the BSP.

3a	*BioSPI_BSPAttach_ValidParam*	**9.3.1.3, A.4**	Verify if a call to the function BioSPI_BSPAttach with valid input parameters returns BioAPI_OK.
3b	*BioSPI_BSPAttach_InvalidUUID*	**9.3.1.3, 11.2.3**	Verify if a call to the function BioSPI_BSPAttach returns BioAPIERR_INVALID_UUID when the input UUID is invalid.
3c	*BioSPI_BSPAttach_InvalidVersion*	**9.3.1.3, 11.2.3**	Verify if a call to the function BioSPI_BSPAttach returns BioAPIERR_INCOMPATIBLE_VERSION when the caller specifies an incompatible version.
3d	*BioSPI_BSPAttach_InvalidBSPHandle*	**9.3.1.3**	Verify if the function BioSPI_BSPAttach returns an error when called with an invalid BSP handle.
4a	*BioSPI_BSPDetach_ValidParam*	**9.3.1.4, A.4**	Verify if a call to the function BioSPI_BSPDetach with a valid module handle returns BioAPI_OK.
4b	*BioSPI_BSPDetach_InvalidBSPHandle*	**9.3.1.4, 8.1.8.1**	Verify if a call to BioSPI_BSPDetach with an invalid module handle returns an error.
4c	*BioSPI_BSPDetach_Confirm*	**9.3.1.4, 8.1.8.4**	Verify if a call to the function BioSPI_BSPDetach truly terminates the attach session.
5a	*BioSPI_FreeBIRHandle_ValidParam*	**9.3.2.1, 8.2.1, A.4**	Verify if calling the function BioSPI_FreeBIRHandle with a valid BIR handle frees the BIR handle.
5b	*BioSPI_FreeBIRHandle_InvalidBSPHandle*	**9.3.2.1, 8.2.1**	Verify if calling BioSPI_FreeBIRHandle with an invalid BSP handle returns an error.
5c	*BioSPI_FreeBIRHandle_InvalidBIRHandle*	**9.3.2.1, 8.2.1**	Verify if calling BioSPI_FreeBIRHandle with an invalid BIR handle returns an error.

6a	*BioSPI_GetBIRFromHandle_ValidParam*	**9.3.2.2, 8.2.2, A.4**	Verify if calling BioSPI_GetBIRFromHandle with valid parameters returns BioAPI_OK.
6b	*BioSPI_GetBIRFromHandle_InvalidBSPHandle*	**9.3.2.2, 8.2.2**	Verify if calling BioSPI_GetBIRFromHandle with an invalid BSP handle returns an error.
6c	*BioSPI_GetBIRFromHandle_InvalidBIRHandle*	**9.3.2.2, 8.2.2**	Verify if calling the function BioSPI_GetBIRFromHandle with an invalid BIR handle returns an error.
7a	*BioSPI_GetHeaderFromHandle_ValidParam*	**9.3.2.3, 8.2.3, A.4**	Verify if a call to the function BioSPI_GetHeaderFromHandle with valid input parameters returns BioAPI_OK.
7b	*BioSPI_GetHeaderFromHandle_InvalidBSPHandle*	**9.3.2.3, 8.2.3**	Verify if invoking the function BioSPI_GetHeaderFromHandle with an invalid module handle returns an error.
7c	*BioSPI_GetHeaderFromHandle_InvalidBIRHandle*	**9.3.2.3, 8.2.3**	Verify if a call to the function BioSPI_GetHeaderFromHandle with an invalid BIR handle returns an error.
7d	*BioSPI_GetHeaderFromHandle_BIRHandleNotFreed*	**9.3.2.3, 8.2.3**	Verify that after a call to the function BioSPI_GetHeaderFromHandle, the BIR handle has not been freed.
8a	*BioSPI_EnableEvents_ValidParam*	**9.3.3.1, 8.3.1**	Verify BioSPI_EnableEvents with valid input parameters.
8b	*BioSPI_EnableEvents_InvalidBSPHandle*	**9.3.3.1, 8.3.1**	Verify BioSPI_EnableEvents with an invalid module handle.

9a	*BioSPI_Capture_AuditData*	**9.3.4.1, 8.4.1.1, 7.47, A.4.6.2.1**	Verify the function BioSPI_Capture with AuditData having a non-NULL value.
9b	*BioSPI_Capture_ReturnQuality*	**9.3.4.1, 8.4.1, 7.49, 7.47, A.4.6.2.2**	Verify if the header of the captured BIR contains a valid quality value (in the range 0-100).
9c	*BioSPI_Capture_ IntermediateProcessedBIR*	**9.3.4.1, 8.4.1.1, 8.4.1.2**	Verify if the BIR returned by the function BioSPI_Capture has a processed level of either INTERMEDIATE or PROCESSED.
9d	*BioSPI_Capture_InvalidBSPHandle*	**9.3.4.1, 8.4.1.2**	Verify if calling BioSPI_Capture with an invalid BSP handle returns an error BioAPIERR_INVALID_BSP_HANDLE.
10a	*BioSPI_CreateTemplate_PayloadSupported*	**9.3.4.2, 8.4.5.1, 7.47, A.4.6.2.6**	Verify BioSPI_CreateTemplate with valid parameters and payload.
10b	*BioSPI_CreateTemplate_BIRHeaderQuality*	**9.3.4.2, 7.49.3, 7.47, A.4.6.2.2**	Verify the function BioSPI_CreateTemplate with valid parameters and the returned quality value.
10c	*BioSPI_CreateTemplate_OutputBIRDataType*	**9.3.4.2, 8.4.2.1**	Verify BioSPI_CreateTemplate with valid parameters. The new template BIR is expected to have the processed level PROCESSED.
10d	*BioSPI_CreateTemplate_OutputBIRPurpose*	**9.3.4.2, 8.4.2.1**	Verify if the purpose of the created template is the same as the purpose of the captured BIR.
10e	*BioSPI_CreateTemplate_InputBIRDataType*	**9.3.4.2, 8.4.2.1**	Verify BioSPI_CreateTemplate with an input BIR that has an invalid processed level.
10f	*BioSPI_CreateTemplate_Inconsistent_Purpose*	**9.3.4.2, 8.4.2.1**	Verify function BioSPI_CreateTemplate with an invalid input BIR purpose.

11a	*BioSPI_Process_ValidParam*	**9.3.4.3, 8.4.3.1**	Verify BioSPI_Process with valid parameters and the returned processed level.
11b	*BioSPI_Process_BIRHeaderQuality*	**9.3.4.3, 8.4.3.1, 7.49.3, 7.47, A.4.6.2.2**	Verify BioSPI_Process with valid parameters and if the returned quality is valid.
11c	*BioSPI_Process_OutputBIRPurpose*	**9.3.4.3, 8.4.3.1, 7.12.3**	Verify BioSPI_Process with valid parameters and if the purpose of the processed BIR is the same as the purpose of the captured BIR.
11d	*BioSPI_Process_BuildsProcessedBIR*	**9.3.4.3, 8.4.3.1**	Verify BioSPI_Process with valid parameters and the returned processed level.
11e	*BioSPI_Process_InputBIRDataType*	**9.3.4.3, 8.4.3.1**	Verify BioSPI_Process with an input BIR having a processed level of PROCESSED and if the BioSPI_Process call fails.
12a	*BioSPI_VerifyMatch_ValidParam*	**9.3.4.5, 8.4.5.1**	Verify if calling BioSPI_VerifyMatch with valid input parameters returns BioAPI_OK.
12b	*BioSPI_VerifyMatch_Payload*	**9.3.4.5, 8.4.5.1, 7.47**	Verify the support of payload in the function BioSPI_VerifyMatch. The function is expected to return BioAPI_OK.
12c	*BioSPI_VerifyMatch_ Inconsistent_Purpose*	**9.3.4.5, 8.4.5.1, 11.2.3**	Verify BioSPI_VerifyMatch with a BIR whose purpose is invalid for the function. The function is expected to return BioAPIERR_BSP_INCONSISTEN T_PURPOSE.

13a	*BioSPI_Enroll_ValidParam*	**9.3.4.7, 8.4.7.1, A.4**	Verify if calling BioSPI_Enroll with valid input parameters returns BioAPI_OK.
13b	*BioSPI_Enroll_Payload*	**9.3.4.7, 8.4.7.1, A.4.6.2.6, 7.47**	Verify if calling BioSPI_Enroll with a payload returns BioAPI_OK.
13c	*BioSPI_Enroll_AuditData*	**9.3.4.7, 8.4.7.1, 7.47, A.4.6.2.1**	Verify BioSPI_Enroll with a non-NULL AuditData parameter. The function is expected to return audit data if the BSP supports audit data.
13d	*BioSPI_Enroll_BIRHeaderQuality*	**9.3.4.7, 8.4.7.1, 7.49.3, 7.47, A.4.6.2.2**	Verify if calling the function BioSPI_Enroll returns a valid quality value in the new template BIR's header.
14a	*BioSPI_Verify_ValidParam*	**9.3.4.8, 8.4.8.1, A.4**	Verify if calling the function BioSPI_Verify with valid input parameters returns BioAPI_OK.
14b	*BioSPI_Verify_Payload*	**9.3.4.8, 8.4.8.1, 7.47, A.4.6.2.6**	Verify if calling the function BioSPI_Verify with a payload returns BioAPI_OK.
14c	*BioSPI_Verify_AuditData*	**9.3.4.8, 8.4.8.1, 7.47, A.4.6.2.1**	Verify if calling BioSPI_Verify with audit data returns BioAPI_OK.
15a	*BioSPI_DbOpen_ValidParam*	**9.3.5.1, 8.5.1.1**	Verify BioSPI_DbOpen with valid input parameters
15b	*BioSPI_DbOpen_InvalidBSPHandle*	**9.3.5.1, 8.5.1.1**	Verify BioSPI_DbOpen with an invalid BSP handle
16a	*BioSPI_DbClose_ValidParam*	**9.3.5.2, 8.5.2.1**	Verify BioSPI_DbClose with valid input parameters
16b	*BioSPI_DbClose_InvalidBSPHandle*	**9.3.5.2, 8.5.2.1**	Verify BioSPI_DbClose with an invalid BSP handle

17a	*BioSPI_DbCreate_DbProtected*	**9.3.5.3, 8.5.3.1**	Invoke BioSPI_DbCreate twicw, and verify if the second invocation results in returning error code BioAPIERR_DATABASE_ALREADY_EXISTS.
17b	*BioSPI_DbCreate_ValidParam*	**9.3.5.3, 8.5.3.1**	Verify BioSPI_DbCreate with valid input parameters
17c	*BioSPI_DbCreate_InvalidBSPHandle*	**9.3.5.3, 8.5.3.1**	Verify BioSPI_DbCreate with invalid BSP handle
18a	*BioSPI_DbDelete_InvalidBSPHandle*	**9.3.5.4, 8.5.4.1**	Verify BioSPI_DbDelete with an invalid BSP handle
18b	*BioSPI_DbDelete_OpenDbProtected*	**9.3.5.4, 8.5.4.1**	Verify BioSPI_DbDelete when the database is open
18c	*BioSPI_DbDelete_ValidParam*	**9.3.5.4, 8.5.4.1**	Verify BioSPI_DbDelete with valid input parameters
19a	*BioSPI_DbSetMarker_ValidParam*	**9.3.5.5, 8.5.5.1**	Verify BioSPI_DbSetMarker with valid input parameters
19b	*BioSPI_DbSetMarker_InvalidBSPHandle*	**9.3.5.5, 8.5.5.1**	Verify BioSPI_DbSetMarker with invalid BSP handle
19c	*BioSPI_DbSetMarker_RecordNotFound*	**9.3.5.5, 8.5.5.1**	Verify BioSPI_DbSetMarker with a keyvalue that does not exist in the database
20a	*BioSPI_DbFreeMarker_ValidParam*	**9.3.5.6, 8.5.6.1**	Verify BioSPI_DbFreeMarker with valid input parameters
20b	*BioSPI_DbFreeMarker_InvalidBSPHandle*	**9.3.5.6, 8.5.6.1**	Verify BioSPI_DbFreeMarker with an invalid BSP handle
20c	*BioSPI_DbFreeMarker_InvalidMarker*	**9.3.5.6, 8.5.6.1**	Verify BioSPI_DbFreeMarker with invalid marker handle
21a	*BioSPI_DbStoreBIR_ValidParam*	**9.3.5.7, 8.5.7.1**	Verify BioSPI_DbStoreBIR with valid input parameters
21b	*BioSPI_DbStoreBIR_InvalidBSPHandle*	**9.3.5.7, 8.5.7.1**	Verify BioSPI_DbStoreBIR with invalid BSP handle
22a	*BioSPI_DbGetBIR_ValidParam*	**9.3.5.8, 8.5.8.1**	Verify BioSPI_DbGetBIR with valid input parameters
22b	*BioSPI_DbGetBIR_InvalidBSPHandle*	**9.3.5.8, 8.5.8.1**	Verify BioSPI_DbGetBIR with invalid BSP handle
22c	*BioSPI_DbGetBIR_RecordNotFound*	**9.3.5.8, 8.5.8.1**	Verify BioSPI_DbGetBIR with a key value that does not exist in the database

23a	*BioSPI_DbGetNextBIR_ValidParam*	**9.3.5.9, 8.5.9.1**	Verify BioSPI_DbGetNextBIR with valid input parameters
23b	*BioSPI_DbGetNextBIR_InvalidBSPHandle*	**9.3.5.9, 8.5.9.1**	Verify BioSPI_DbGetNextBIR with invalid BSP handle
24a	*BioSPI_DbDeleteBIR_ValidParam*	**9.3.5.10, 8.5.10.1**	Verify BioSPI_DbDeleteBIR with valid parameters
24b	*BioSPI_DbDeleteBIR_InvalidBSPHandle*	**9.3.5.10, 8.5.10.1**	Verify BioSPI_DbDeleteBIR with an invalid BSP handle

7.2 Testing BSPs of subclass "Verification BSP"

7.2.1 The following subclauses specify which test assertions shall be used to determine whether a BSP of the conformance category "Verification BSP" satisfies the requirements specified in ISO/IEC 19784-1:2006, A.4 and A.4.1.

7.2.2 All verification BSPs shall be tested by executing all of the following assertions (in order):

Table 3 — Assertions for Testing Verification BSP

Number	Assertion name	Package
1a	*BioSPI_BSPLoad_InvalidUUID*	020e90c8-0c19-1085-ab54-0002a5d5fd2e
1b	*BioSPI_BSPLoad_ValidParam*	01f6c6f0-0c19-1085-97fe-0002a5d5fd2e
2a	*BioSPI_BSPUnload_ValidParam*	01661010-0c22-1085-8688-0002a5d5fd2e
2b	*BioSPI_BSPUnload_InvalidUUID*	01c2e5b0-0c3b-1085-b31d-0002a5d5fd2e
2c	*BioSPI_BSPUnload_UnmatchedLoad*	02f6c618-0c23-1085-ba89-0002a5d5fd2e
2d	*BioSPI_BSPUnload_Confirm*	03daf040-0c3b-1085-a9fd-0002a5d5fd2e
3a	*BioSPI_BSPAttach_ValidParam*	00ae6488-0c3d-1085-9912-0002a5d5fd2e
3b	*BioSPI_BSPAttach_InvalidUUID*	049cc170-0c5f-1085-981f-0002a5d5fd2e
3c	*BioSPI_BSPAttach_InvalidVersion*	0052ac10-0c60-1085-9883-0002a5d5fd2e
3d	*BioSPI_BSPAttach_InvalidBSPHandle*	03826830-0c57-1085-bfb0-0002a5d5fd2e
4a	*BioSPI_BSPDetach_ValidParam*	00e0d2b0-0c7a-1085-b8ac-0002a5d5fd2e
4b	*BioSPI_BSPDetach_InvalidBSPHandle*	0434c458-0c79-1085-9f2c-0002a5d5fd2e
4c	*BioSPI_BSPDetach_Confirm*	002e7e58-0c78-1085-9e1d-0002a5d5fd2e
5a	*BioSPI_FreeBIRHandle_ValidParam*	0280a7d0-0c80-1085-a9a0-0002a5d5fd2e
5b	*BioSPI_FreeBIRHandle_InvalidBSPHandle*	047aed48-0c80-1085-898b-0002a5d5fd2e
5c	*BioSPI_FreeBIRHandle_InvalidBIRHandle*	018e6c18-0c9c-1085-afdf-0002a5d5fd2e

6a	*BioSPI_GetBIRFromHandle_ValidParam*	0460b658-0cb4-1085-a304-0002a5d5fd2e
6b	*BioSPI_GetBIRFromHandle_ InvalidBSPHandle*	02445668-0cc5-1085-a3ac-0002a5d5fd2e
6c	*BioSPI_GetBIRFromHandle_ InvalidBIRHandle*	0194a9c0-0cc7-1085-8780-0002a5d5fd2e
7a	*BioSPI_GetHeaderFromHandle_ ValidParam*	027a7db0-0cc7-1085-9391-0002a5d5fd2e
7b	*BioSPI_GetHeaderFromHandle_ InvalidBSPHandle*	057e0d38-0ccd-1085-83b8-0002a5d5fd2e
7c	*BioSPI_GetHeaderFromHandle_ InvalidBIRHandle*	02195e68-0cce-1085-a46f-0002a5d5fd2e
7d	*BioSPI_GetHeaderFromHandle_ BIRHandleNotFreed*	01cc0988-0ccf-1085-a367-0002a5d5fd2e
8a	*BioSPI_EnableEvents_ValidParam*	0333f628-0ccf-1085-aceb-0002a5d5fd2e
8b	*BioSPI_EnableEvents_InvalidBSPHandle*	04ed0838-0ccf-1085-b64e-0002a5d5fd2e
13a	*BioSPI_Enroll_ValidParam*	0b5ebb60-eefb-11d9-990c-0002a5d5c51b
13b	*BioSPI_Enroll_Payload*	e8969d40-ef05-11d9-9098-0002a5d5c51b
13c	*BioSPI_Enroll_AuditData*	b40a5260-ef14-11d9-a4fe-0002a5d5c51b
13d	*BioSPI_Enroll_BIRHeaderQuality*	6f727320-ef1a-11d9-9143-0002a5d5c51b
14a	*BioSPI_Verify_ValidParam*	b78e5be0-efcb-11d9-b2c7-0002a5d5c51b
14b	*BioSPI_Verify_Payload*	32969ec0-eff8-11d9-9831-0002a5d5c51b
14c	*BioSPI_Verify_AuditData*	89719700-f218-11d9-b028-0002a5d5c51b

7.2.3 If an implementation claims to support the function *BioSPI_Capture*, the implementation shall be tested by executing all of the following assertions (in order):

Table 4 — Assertions for Testing Verification BSP - BioSPI_Capture

Number	Assertion name	Package
9a	*BioSPI_Capture_AuditData*	02704c50-0cd8-1085-96cb-0002a5d5fd2e
9b	*BioSPI_Capture_ReturnQuality*	03f601f0-0cd8-1085-bd59-0002a5d5fd2e
9c	*BioSPI_Capture_ IntermediateProcessedBIR*	055ddb08-0cd6-1085-a6d3-0002a5d5fd2e
9d	*BioSPI_Capture_InvalidBSPHandle*	0244f2a8-0cf2-1085-8443-0002a5d5fd2e

7.2.4 If an implementation claims to support the function *BioSPI_CreateTemplate*, the implementation shall be tested by executing all of the following assertions (in order):

Table 5 — Assertions for Testing Verification BSP - BioSPI_CreateTemplate

Number	Assertion name	Package
10a	*BioSPI_CreateTemplate_PayloadSupported*	04a01118-0cf9-1085-96d4-0002a5d5fd2e
10b	*BioSPI_CreateTemplate_ BIRHeaderQuality*	00b5c728-0cfb-1085-8969-0002a5d5fd2e
10c	*BioSPI_CreateTemplate_ OutputBIRDataType*	0193c730-0cf9-1085-b0a3-0002a5d5fd2e
10d	*BioSPI_CreateTemplate_ OutputBIRPurpose*	03dbdaa0-0cf2-1085-99ed-0002a5d5fd2e
10e	*BioSPI_CreateTemplate_ InputBIRDataType*	6d543ea0-2ce9-11d9-9669-0800200c9a66
10f	*BioSPI_CreateTemplate_ Inconsistent_Purpose*	28ec1620-e995-11d9-b1d1-0002a5d5c51b

7.2.5 If an implementation claims to support the function *BioSPI_Process*, the implementation shall be tested by executing all of the following assertions (in order):

Table 6 — Assertions for Testing Verification BSP - BioSPI_Process

Number	Assertion name	Package
11a	*BioSPI_Process_ValidParam*	4ec34700-e9a0-11d9-8fc8-0002a5d5c51b
11b	*BioSPI_Process_BIRHeaderQuality*	211668e0-e9a6-11d9-bcc8-0002a5d5c51b
11c	*BioSPI_Process_OutputBIRPurpose*	e1bb4f20-ed61-11d9-9344-0002a5d5c51b
11d	*BioSPI_Process_BuildsProcessedBIR*	f2ce6540-ed66-11d9-9618-0002a5d5c51b
11e	*BioSPI_Process_InputBIRDataType*	3cf96080-ed6b-11d9-9acf-0002a5d5c51b

7.2.6 If an implementation claims to support the function BioSPI_VerifyMatch, the implementation shall be tested by executing all of the following assertions (in order):

Table 7 — Assertions for Testing Verification BSP - BioSPI_VerifyMatch

Number	Assertion name	Package
12a	*BioSPI_VerifyMatch_ValidParam*	688aad60-ee30-11d9-a62c-0002a5d5c51b
12b	*BioSPI_VerifyMatch_Payload*	692ebe20-ee47-11d9-bd34-0002a5d5c51b
12c	*BioSPI_VerifyMatch_ Inconsistent_Purpose*	9108ec70-2e9b-11d9-9669-0800200c9a66

7.2.7 If an implementation claims to support a BSP-controlled database (see 19784-1:2006, A.4.6.1.2), the implementation shall be tested by executing all of the following assertions (in order):

Table 8 — Assertions for Testing Verification BSP - Database Functions

Number	Assertion name	Package
15a	*BioSPI_DbOpen_ValidParam*	e68ff9a0-e506-11d9-a6a1-0002a5d5c51b
15b	*BioSPI_DbOpen_InvalidBSPHandle*	bfd44400-e5de-11d9-bdb9-0002a5d5c51b
16a	*BioSPI_DbClose_ValidParam*	39aa9560-e5f1-11d9-89f3-0002a5d5c51b
16b	*BioSPI_DbClose_InvalidBSPHandle*	6e3f5c00-e5f3-11d9-b663-0002a5d5c51b
17a	*BioSPI_DbCreate_DbProtected*	7b6c2f40-e650-11d9-812f-0002a5d5c51b
17b	*BioSPI_DbCreate_ValidParam*	1421ec38-1db6-49d4-873d-03e2de17598b
17c	*BioSPI_DbCreate_InvalidBSPHandle*	ef4bb862-79f6-4f01-8f5d-af5c3abf23c0
18a	*BioSPI_DbDelete_InvalidBSPHandle*	678e5d12-3d51-41ec-a672-13f34ea24545
18b	*BioSPI_DbDelete_OpenDbProtected*	9e4d0c6d-4d59-479c-9f58-160ecde99aad
18c	*BioSPI_DbDelete_ValidParam*	499d9cc3-4269-4671-9d69-29a31bc1a08f
19a	*BioSPI_DbSetMarker_ValidParam*	94271080-e723-11d9-898c-0002a5d5c51b
19b	*BioSPI_DbSetMarker_InvalidBSPHandle*	69f7ce20-e72e-11d9-b4e0-0002a5d5c51b
19c	*BioSPI_DbSetMarker_RecordNotFound*	b9c90f40-e7ec-11d9-a435-0002a5d5c51b
20a	*BioSPI_DbFreeMarker_ValidParam*	2e1c9520-e7f1-11d9-a011-0002a5d5c51b
20b	*BioSPI_DbFreeMarker_InvalidBSPHandle*	0f735140-e800-11d9-8a8a-0002a5d5c51b
20c	*BioSPI_DbFreeMarker_InvalidMarker*	a9007b60-e802-11d9-8a5d-0002a5d5c51b
21a	*BioSPI_DbStoreBIR_ValidParam*	da953680-e806-11d9-90d3-0002a5d5c51b
21b	*BioSPI_DbStoreBIR_InvalidBSPHandle*	e39027e0-e80b-11d9-85eb-0002a5d5c51b
22a	*BioSPI_DbGetBIR_ValidParam*	67512700-e811-11d9-a5e0-0002a5d5c51b
22b	*BioSPI_DbGetBIR_InvalidBSPHandle*	f62947e0-e8b7-11d9-9dad-0002a5d5c51b
22c	*BioSPI_DbGetBIR_RecordNotFound*	37457440-e8ba-11d9-87da-0002a5d5c51b
23a	*BioSPI_DbGetNextBIR_ValidParam*	e3396400-e8c9-11d9-990f-0002a5d5c51b
23b	*BioSPI_DbGetNextBIR_InvalidBSPHandle*	f0ef5320-e8d3-11d9-bbc1-0002a5d5c51b

24a	*BioSPI_DbDeleteBIR_ValidParam*	ed4afbe0-e8d6-11d9-9ed0-0002a5d5c51b
24b	*BioSPI_DbDeleteBIR_InvalidBSPHandle*	72eca940-e8d9-11d9-aa0d-0002a5d5c51b

7.3 Testing BSPs of subclass "Identification BSP"

7.3.1 The following subclauses specify which test assertions shall be used to determine whether a BSP of the conformance subclass "Identification BSP" satisfies the requirements specified in ISO/IEC 19784-1:2006, A.4 and A.4.2

7.3.2 All identification BSPs shall be tested by executing all of the following assertions (in order):

Table 9 — Assertions for Testing Identification BSP

Number	Assertion name	Package
1a	*BioSPI_BSPLoad_InvalidUUID*	020e90c8-0c19-1085-ab54-0002a5d5fd2e
1b	*BioSPI_BSPLoad_ValidParam*	01f6c6f0-0c19-1085-97fe-0002a5d5fd2e
2a	*BioSPI_BSPUnload_ValidParam*	01661010-0c22-1085-8688-0002a5d5fd2e
2b	*BioSPI_BSPUnload_InvalidUUID*	01c2e5b0-0c3b-1085-b31d-0002a5d5fd2e
2c	*BioSPI_BSPUnload_UnmatchedLoad*	02f6c618-0c23-1085-ba89-0002a5d5fd2e
2d	*BioSPI_BSPUnload_Confirm*	03daf040-0c3b-1085-a9fd-0002a5d5fd2e
3a	*BioSPI_BSPAttach_ValidParam*	00ae6488-0c3d-1085-9912-0002a5d5fd2e
3b	*BioSPI_BSPAttach_InvalidUUID*	049cc170-0c5f-1085-981f-0002a5d5fd2e
3c	*BioSPI_BSPAttach_InvalidVersion*	0052ac10-0c60-1085-9883-0002a5d5fd2e
3d	*BioSPI_BSPAttach_InvalidBSPHandle*	03826830-0c57-1085-bfb0-0002a5d5fd2e
4a	*BioSPI_BSPDetach_ValidParam*	00e0d2b0-0c7a-1085-b8ac-0002a5d5fd2e
4b	*BioSPI_BSPDetach_InvalidBSPHandle*	0434c458-0c79-1085-9f2c-0002a5d5fd2e
4c	*BioSPI_BSPDetach_Confirm*	002e7e58-0c78-1085-9e1d-0002a5d5fd2e
5a	*BioSPI_FreeBIRHandle_ValidParam*	0280a7d0-0c80-1085-a9a0-0002a5d5fd2e
5b	*BioSPI_FreeBIRHandle_InvalidBSPHandle*	047aed48-0c80-1085-898b-0002a5d5fd2e
5c	*BioSPI_FreeBIRHandle_InvalidBIRHandle*	018e6c18-0c9c-1085-afdf-0002a5d5fd2e

6a	*BioSPI_GetBIRFromHandle_ValidParam*	0460b658-0cb4-1085-a304-0002a5d5fd2e
6b	*BioSPI_GetBIRFromHandle_ InvalidBSPHandle*	02445668-0cc5-1085-a3ac-0002a5d5fd2e
6c	*BioSPI_GetBIRFromHandle_ InvalidBIRHandle*	0194a9c0-0cc7-1085-8780-0002a5d5fd2e
7a	*BioSPI_GetHeaderFromHandle_ ValidParam*	027a7db0-0cc7-1085-9391-0002a5d5fd2e
7b	*BioSPI_GetHeaderFromHandle_ InvalidBSPHandle*	057e0d38-0ccd-1085-83b8-0002a5d5fd2e
7c	*BioSPI_GetHeaderFromHandle_ InvalidBIRHandle*	02195e68-0cce-1085-a46f-0002a5d5fd2e
7d	*BioSPI_GetHeaderFromHandle_ BIRHandleNotFreed*	01cc0988-0ccf-1085-a367-0002a5d5fd2e
8a	*BioSPI_EnableEvents_ValidParam*	0333f628-0ccf-1085-aceb-0002a5d5fd2e
8b	*BioSPI_EnableEvents_InvalidBSPHandle*	04ed0838-0ccf-1085-b64e-0002a5d5fd2e
13a	*BioSPI_Enroll_ValidParam*	0b5ebb60-eefb-11d9-990c-0002a5d5c51b
13b	*BioSPI_Enroll_Payload*	e8969d40-ef05-11d9-9098-0002a5d5c51b
13c	*BioSPI_Enroll_AuditData*	b40a5260-ef14-11d9-a4fe-0002a5d5c51b
13d	*BioSPI_Enroll_BIRHeaderQuality*	6f727320-ef1a-11d9-9143-0002a5d5c51b
14a	*BioSPI_Verify_ValidParam*	b78e5be0-efcb-11d9-b2c7-0002a5d5c51b
14b	*BioSPI_Verify_Payload*	32969ec0-eff8-11d9-9831-0002a5d5c51b
14c	*BioSPI_Verify_AuditData*	89719700-f218-11d9-b028-0002a5d5c51b

7.3.3 If an implementation claims to support the function *BioSPI_Capture*, the implementation shall be tested by executing all of the following assertions (in order):

Table 10 — Assertions for Testing Identification BSP - BioSPI_Capture

Number	Assertion name	Package
9a	*BioSPI_Capture_AuditData*	02704c50-0cd8-1085-96cb-0002a5d5fd2e
9b	*BioSPI_Capture_ReturnQuality*	03f601f0-0cd8-1085-bd59-0002a5d5fd2e
9c	*BioSPI_Capture_ IntermediateProcessedBIR*	055ddb08-0cd6-1085-a6d3-0002a5d5fd2e
9d	*BioSPI_Capture_InvalidBSPHandle*	0244f2a8-0cf2-1085-8443-0002a5d5fd2e

7.3.4 If an implementation claims to support the function *BioSPI_CreateTemplate*, the implementation shall be tested by executing all of the following assertions (in order):

Table 11 — Assertions for Testing Identification BSP - BioSPI_CreateTemplate

Number	Assertion name	Package
10a	*BioSPI_CreateTemplate_PayloadSupported*	04a01118-0cf9-1085-96d4-0002a5d5fd2e
10b	*BioSPI_CreateTemplate_ BIRHeaderQuality*	00b5c728-0cfb-1085-8969-0002a5d5fd2e
10c	*BioSPI_CreateTemplate_ OutputBIRDataType*	0193c730-0cf9-1085-b0a3-0002a5d5fd2e
10d	*BioSPI_CreateTemplate_ OutputBIRPurpose*	03dbdaa0-0cf2-1085-99ed-0002a5d5fd2e
10e	*BioSPI_CreateTemplate_ InputBIRDataType*	6d543ea0-2ce9-11d9-9669-0800200c9a66
10f	*BioSPI_CreateTemplate_ Inconsistent_Purpose*	28ec1620-e995-11d9-b1d1-0002a5d5c51b

7.3.5 If an implementation claims to support the function *BioSPI_Process*, the implementation shall be tested by executing all of the following assertions (in order):

Table 12 — Assertions for Testing Identification BSP - BioSPI_Process

Number	Assertion name	Package
11a	*BioSPI_Process_ValidParam*	4ec34700-e9a0-11d9-8fc8-0002a5d5c51b
11b	*BioSPI_Process_BIRHeaderQuality*	211668e0-e9a6-11d9-bcc8-0002a5d5c51b
11c	*BioSPI_Process_OutputBIRPurpose*	e1bb4f20-ed61-11d9-9344-0002a5d5c51b
11d	*BioSPI_Process_BuildsProcessedBIR*	f2ce6540-ed66-11d9-9618-0002a5d5c51b
11e	*BioSPI_Process_InputBIRDataType*	3cf96080-ed6b-11d9-9acf-0002a5d5c51b

7.3.6 If an implementation claims to support the function *BioSPI_VerifyMatch*, the implementation shall be tested by executing all of the following assertions (in order):

Table 13 — Assertions for Testing Identification BSP - BioSPI_VerifyMatch

Number	Assertion name	Package
12a	*BioSPI_VerifyMatch_ValidParam*	688aad60-ee30-11d9-a62c-0002a5d5c51b
12b	*BioSPI_VerifyMatch_Payload*	692ebe20-ee47-11d9-bd34-0002a5d5c51b
12c	*BioSPI_VerifyMatch_ Inconsistent_Purpose*	9108ec70-2e9b-11d9-9669-0800200c9a66

7.3.7 If an implementation claims to support a BSP-controlled database (see 19784-1:2006, A.4.6.1.2), the implementation shall be tested by executing all of the following assertions (in order):

Table 14 — Assertions for Testing Identification BSP - Database Functions

Number	Assertion name	Package
15a	*BioSPI_DbOpen_ValidParam*	e68ff9a0-e506-11d9-a6a1-0002a5d5c51b
15b	*BioSPI_DbOpen_InvalidBSPHandle*	bfd44400-e5de-11d9-bdb9-0002a5d5c51b
16a	*BioSPI_DbClose_ValidParam*	39aa9560-e5f1-11d9-89f3-0002a5d5c51b
16b	*BioSPI_DbClose_InvalidBSPHandle*	6e3f5c00-e5f3-11d9-b663-0002a5d5c51b
17a	*BioSPI_DbCreate_DbProtected*	7b6c2f40-e650-11d9-812f-0002a5d5c51b
17b	*BioSPI_DbCreate_ValidParam*	1421ec38-1db6-49d4-873d-03e2de17598b
17c	*BioSPI_DbCreate_InvalidBSPHandle*	ef4bb862-79f6-4f01-8f5d-af5c3abf23c0
18a	*BioSPI_DbDelete_InvalidBSPHandle*	678e5d12-3d51-41ec-a672-13f34ea24545
18b	*BioSPI_DbDelete_OpenDbProtected*	9e4d0c6d-4d59-479c-9f58-160ecde99aad
18c	*BioSPI_DbDelete_ValidParam*	499d9cc3-4269-4671-9d69-29a31bc1a08f
19a	*BioSPI_DbSetMarker_ValidParam*	94271080-e723-11d9-898c-0002a5d5c51b
19b	*BioSPI_DbSetMarker_InvalidBSPHandle*	69f7ce20-e72e-11d9-b4e0-0002a5d5c51b
19c	*BioSPI_DbSetMarker_RecordNotFound*	b9c90f40-e7ec-11d9-a435-0002a5d5c51b
20a	*BioSPI_DbFreeMarker_ValidParam*	2e1c9520-e7f1-11d9-a011-0002a5d5c51b
20b	*BioSPI_DbFreeMarker_InvalidBSPHandle*	0f735140-e800-11d9-8a8a-0002a5d5c51b
20c	*BioSPI_DbFreeMarker_InvalidMarker*	a9007b60-e802-11d9-8a5d-0002a5d5c51b
21a	*BioSPI_DbStoreBIR_ValidParam*	da953680-e806-11d9-90d3-0002a5d5c51b
21b	*BioSPI_DbStoreBIR_InvalidBSPHandle*	e39027e0-e80b-11d9-85eb-0002a5d5c51b
22a	*BioSPI_DbGetBIR_ValidParam*	67512700-e811-11d9-a5e0-0002a5d5c51b
22b	*BioSPI_DbGetBIR_InvalidBSPHandle*	f62947e0-e8b7-11d9-9dad-0002a5d5c51b
22c	*BioSPI_DbGetBIR_RecordNotFound*	37457440-e8ba-11d9-87da-0002a5d5c51b

23a	*BioSPI_DbGetNextBIR_ValidParam*	e3396400-e8c9-11d9-990f-0002a5d5c51b
23b	*BioSPI_DbGetNextBIR_InvalidBSPHandle*	f0ef5320-e8d3-11d9-bbc1-0002a5d5c51b
24a	*BioSPI_DbDeleteBIR_ValidParam*	ed4afbe0-e8d6-11d9-9ed0-0002a5d5c51b
24b	*BioSPI_DbDeleteBIR_InvalidBSPHandle*	72eca940-e8d9-11d9-aa0d-0002a5d5c51b

7.4 Testing BSPs of subclass "Capture BSP"

7.4.1 The following subclauses specify which test assertions shall be used to determine whether a BSP of the conformance subclass "Capture BSP" satisfies the requirements specified in ISO/IEC 19784-1:2006, A.4 and A.4.3.

7.4.2 All BSPs of this subclass shall be tested by executing all of the following assertions (in order):

Table 15 — Assertions for Testing Capture BSP

Number	Assertion name	Package
1a	*BioSPI_BSPLoad_InvalidUUID*	020e90c8-0c19-1085-ab54-0002a5d5fd2e
1b	*BioSPI_BSPLoad_ValidParam*	01f6c6f0-0c19-1085-97fe-0002a5d5fd2e
2a	*BioSPI_BSPUnload_ValidParam*	01661010-0c22-1085-8688-0002a5d5fd2e
2b	*BioSPI_BSPUnload_InvalidUUID*	01c2e5b0-0c3b-1085-b31d-0002a5d5fd2e
2c	*BioSPI_BSPUnload_UnmatchedLoad*	02f6c618-0c23-1085-ba89-0002a5d5fd2e
2d	*BioSPI_BSPUnload_Confirm*	03daf040-0c3b-1085-a9fd-0002a5d5fd2e
3a	*BioSPI_BSPAttach_ValidParam*	00ae6488-0c3d-1085-9912-0002a5d5fd2e
3b	*BioSPI_BSPAttach_InvalidUUID*	049cc170-0c5f-1085-981f-0002a5d5fd2e
3c	*BioSPI_BSPAttach_InvalidVersion*	0052ac10-0c60-1085-9883-0002a5d5fd2e
3d	*BioSPI_BSPAttach_InvalidBSPHandle*	03826830-0c57-1085-bfb0-0002a5d5fd2e
4a	*BioSPI_BSPDetach_ValidParam*	00e0d2b0-0c7a-1085-b8ac-0002a5d5fd2e
4b	*BioSPI_BSPDetach_InvalidBSPHandle*	0434c458-0c79-1085-9f2c-0002a5d5fd2e
4c	*BioSPI_BSPDetach_Confirm*	002e7e58-0c78-1085-9e1d-0002a5d5fd2e

5a	*BioSPI_FreeBIRHandle_ValidParam*	0280a7d0-0c80-1085-a9a0-0002a5d5fd2e
5b	*BioSPI_FreeBIRHandle_InvalidBSPHandle*	047aed48-0c80-1085-898b-0002a5d5fd2e
5c	*BioSPI_FreeBIRHandle_InvalidBIRHandle*	018e6c18-0c9c-1085-afdf-0002a5d5fd2e
6a	*BioSPI_GetBIRFromHandle_ValidParam*	0460b658-0cb4-1085-a304-0002a5d5fd2e
6b	*BioSPI_GetBIRFromHandle_ InvalidBSPHandle*	02445668-0cc5-1085-a3ac-0002a5d5fd2e
6c	*BioSPI_GetBIRFromHandle_ InvalidBIRHandle*	0194a9c0-0cc7-1085-8780-0002a5d5fd2e
7a	*BioSPI_GetHeaderFromHandle_ ValidParam*	027a7db0-0cc7-1085-9391-0002a5d5fd2e
7b	*BioSPI_GetHeaderFromHandle_ InvalidBSPHandle*	057e0d38-0ccd-1085-83b8-0002a5d5fd2e
7c	*BioSPI_GetHeaderFromHandle_ InvalidBIRHandle*	02195e68-0cce-1085-a46f-0002a5d5fd2e
7d	*BioSPI_GetHeaderFromHandle_ BIRHandleNotFreed*	01cc0988-0ccf-1085-a367-0002a5d5fd2e
8a	*BioSPI_EnableEvents_ValidParam*	0333f628-0ccf-1085-aceb-0002a5d5fd2e
8b	*BioSPI_EnableEvents_InvalidBSPHandle*	04ed0838-0ccf-1085-b64e-0002a5d5fd2e
9a	*BioSPI_Capture_AuditData*	02704c50-0cd8-1085-96cb-0002a5d5fd2e
9b	*BioSPI_Capture_ReturnQuality*	03f601f0-0cd8-1085-bd59-0002a5d5fd2e
9c	*BioSPI_Capture_ IntermediateProcessedBIR*	055ddb08-0cd6-1085-a6d3-0002a5d5fd2e
9d	*BioSPI_Capture_InvalidBSPHandle*	0244f2a8-0cf2-1085-8443-0002a5d5fd2e

7.4.3 If an implementation claims to support the function *BioSPI_CreateTemplate*, the implementation shall be tested by executing all of the following assertions (in order):

Table 16 — Assertions for Testing Capture BSP - BioSPI_CreateTemplate

Number	Assertion name	Package
10a	*BioSPI_CreateTemplate_PayloadSupported*	04a01118-0cf9-1085-96d4-0002a5d5fd2e
10b	*BioSPI_CreateTemplate_ BIRHeaderQuality*	00b5c728-0cfb-1085-8969-0002a5d5fd2e
10c	*BioSPI_CreateTemplate_ OutputBIRDataType*	0193c730-0cf9-1085-b0a3-0002a5d5fd2e
10d	*BioSPI_CreateTemplate_ OutputBIRPurpose*	03dbdaa0-0cf2-1085-99ed-0002a5d5fd2e
10e	*BioSPI_CreateTemplate_ InputBIRDataType*	6d543ea0-2ce9-11d9-9669-0800200c9a66
10f	*BioSPI_CreateTemplate_ Inconsistent_Purpose*	28ec1620-e995-11d9-b1d1-0002a5d5c51b

7.4.4 If an implementation claims to support the function *BioSPI_Process*, the implementation shall be tested by executing all of the following assertions (in order):

Table 17 — Assertions for Testing Capture BSP - BioSPI_Process

Number	Assertion name	Package
11a	*BioSPI_Process_ValidParam*	4ec34700-e9a0-11d9-8fc8-0002a5d5c51b
11b	*BioSPI_Process_BIRHeaderQuality*	211668e0-e9a6-11d9-bcc8-0002a5d5c51b
11c	*BioSPI_Process_OutputBIRPurpose*	e1bb4f20-ed61-11d9-9344-0002a5d5c51b
11d	*BioSPI_Process_BuildsProcessedBIR*	f2ce6540-ed66-11d9-9618-0002a5d5c51b
11e	*BioSPI_Process_InputBIRDataType*	3cf96080-ed6b-11d9-9acf-0002a5d5c51b

7.4.5 If an implementation claims to support the function *BioSPI_VerifyMatch*, the implementation shall be tested by executing all of the following assertions (in order):

Table 18 — Assertions for Testing Capture BSP - BioSPI_VerifyMatch

Number	Assertion name	Package
12a	*BioSPI_VerifyMatch_ValidParam*	688aad60-ee30-11d9-a62c-0002a5d5c51b
12b	*BioSPI_VerifyMatch_Payload*	692ebe20-ee47-11d9-bd34-0002a5d5c51b
12c	*BioSPI_VerifyMatch_ Inconsistent_Purpose*	9108ec70-2e9b-11d9-9669-0800200c9a66

7.4.6 If an implementation claims to support the function *BioSPI_Enroll*, the implementation shall be tested by executing all of the following assertions (in order) to verify the claim:

Table 19 — Assertions for Testing Capture BSP - BioSPI_Enroll

Number	Assertion name	Package
13a	*BioSPI_Enroll_ValidParam*	0b5ebb60-eefb-11d9-990c-0002a5d5c51b
13b	*BioSPI_Enroll_Payload*	e8969d40-ef05-11d9-9098-0002a5d5c51b
13c	*BioSPI_Enroll_AuditData*	b40a5260-ef14-11d9-a4fe-0002a5d5c51b
13d	*BioSPI_Enroll_BIRHeaderQuality*	6f727320-ef1a-11d9-9143-0002a5d5c51b

7.4.7 If an implementation claims to support the function *BioSPI_Verify*, the implementation shall be tested by executing all of the following assertions (in order):

Table 20 — Assertions for Testing Capture BSP - BioSPI_Verify

Number	Assertion name	Package
14a	*BioSPI_Verify_ValidParam*	b78e5be0-efcb-11d9-b2c7-0002a5d5c51b
14b	*BioSPI_Verify_Payload*	32969ec0-eff8-11d9-9831-0002a5d5c51b
14c	*BioSPI_Verify_AuditData*	89719700-f218-11d9-b028-0002a5d5c51b

7.4.8 If an implementation claims to support a BSP-controlled database (see 19784-1:2006, A.4.6.1.2), the implementation shall be tested by executing all of the following assertions (in order):

Table 21 — Assertions for Testing Capture BSP - Database Functions

Number	Assertion name	Package
15a	*BioSPI_DbOpen_ValidParam*	e68ff9a0-e506-11d9-a6a1-0002a5d5c51b
15b	*BioSPI_DbOpen_InvalidBSPHandle*	bfd44400-e5de-11d9-bdb9-0002a5d5c51b
16a	*BioSPI_DbClose_ValidParam*	39aa9560-e5f1-11d9-89f3-0002a5d5c51b
16b	*BioSPI_DbClose_InvalidBSPHandle*	6e3f5c00-e5f3-11d9-b663-0002a5d5c51b
17a	*BioSPI_DbCreate_DbProtected*	7b6c2f40-e650-11d9-812f-0002a5d5c51b
17b	*BioSPI_DbCreate_ValidParam*	1421ec38-1db6-49d4-873d-03e2de17598b
17c	*BioSPI_DbCreate_InvalidBSPHandle*	ef4bb862-79f6-4f01-8f5d-af5c3abf23c0

18a	*BioSPI_DbDelete_InvalidBSPHandle*	678e5d12-3d51-41ec-a672-13f34ea24545
18b	*BioSPI_DbDelete_OpenDbProtected*	9e4d0c6d-4d59-479c-9f58-160ecde99aad
18c	*BioSPI_DbDelete_ValidParam*	499d9cc3-4269-4671-9d69-29a31bc1a08f
19a	*BioSPI_DbSetMarker_ValidParam*	94271080-e723-11d9-898c-0002a5d5c51b
19b	*BioSPI_DbSetMarker_InvalidBSPHandle*	69f7ce20-e72e-11d9-b4e0-0002a5d5c51b
19c	*BioSPI_DbSetMarker_RecordNotFound*	b9c90f40-e7ec-11d9-a435-0002a5d5c51b
20a	*BioSPI_DbFreeMarker_ValidParam*	2e1c9520-e7f1-11d9-a011-0002a5d5c51b
20b	*BioSPI_DbFreeMarker_InvalidBSPHandle*	0f735140-e800-11d9-8a8a-0002a5d5c51b
20c	*BioSPI_DbFreeMarker_InvalidMarker*	a9007b60-e802-11d9-8a5d-0002a5d5c51b
21a	*BioSPI_DbStoreBIR_ValidParam*	da953680-e806-11d9-90d3-0002a5d5c51b
21b	*BioSPI_DbStoreBIR_InvalidBSPHandle*	e39027e0-e80b-11d9-85eb-0002a5d5c51b
22a	*BioSPI_DbGetBIR_ValidParam*	67512700-e811-11d9-a5e0-0002a5d5c51b
22b	*BioSPI_DbGetBIR_InvalidBSPHandle*	f62947e0-e8b7-11d9-9dad-0002a5d5c51b
22c	*BioSPI_DbGetBIR_RecordNotFound*	37457440-e8ba-11d9-87da-0002a5d5c51b
23a	*BioSPI_DbGetNextBIR_ValidParam*	e3396400-e8c9-11d9-990f-0002a5d5c51b
23b	*BioSPI_DbGetNextBIR_InvalidBSPHandle*	f0ef5320-e8d3-11d9-bbc1-0002a5d5c51b
24a	*BioSPI_DbDeleteBIR_ValidParam*	ed4afbe0-e8d6-11d9-9ed0-0002a5d5c51b
24b	*BioSPI_DbDeleteBIR_InvalidBSPHandle*	72eca940-e8d9-11d9-aa0d-0002a5d5c51b

7.5 Testing BSPs of subclass "Verification Engine"

7.5.1 The following subclauses specify which test assertions shall be used to determine whether a BSP of the conformance subclass "Verification Engine" satisfies the requirements specified in ISO/IEC 19784-1:2006, A.4 and A.4.4.

7.5.2 All BSPs of this subclass shall be tested by executing all of the following assertions (in order):

Table 22 — Assertions for Testing Verification Engine BSP

Number	Assertion name	Package
1a	*BioSPI_BSPLoad_InvalidUUID*	020e90c8-0c19-1085-ab54-0002a5d5fd2e
1b	*BioSPI_BSPLoad_ValidParam*	01f6c6f0-0c19-1085-97fe-0002a5d5fd2e
2a	*BioSPI_BSPUnload_ValidParam*	01661010-0c22-1085-8688-0002a5d5fd2e
2b	*BioSPI_BSPUnload_InvalidUUID*	01c2e5b0-0c3b-1085-b31d-0002a5d5fd2e
2c	*BioSPI_BSPUnload_UnmatchedLoad*	02f6c618-0c23-1085-ba89-0002a5d5fd2e
2d	*BioSPI_BSPUnload_Confirm*	03daf040-0c3b-1085-a9fd-0002a5d5fd2e
3a	*BioSPI_BSPAttach_ValidParam*	00ae6488-0c3d-1085-9912-0002a5d5fd2e
3b	*BioSPI_BSPAttach_InvalidUUID*	049cc170-0c5f-1085-981f-0002a5d5fd2e
3c	*BioSPI_BSPAttach_InvalidVersion*	0052ac10-0c60-1085-9883-0002a5d5fd2e
3d	*BioSPI_BSPAttach_InvalidBSPHandle*	03826830-0c57-1085-bfb0-0002a5d5fd2e
4a	*BioSPI_BSPDetach_ValidParam*	00e0d2b0-0c7a-1085-b8ac-0002a5d5fd2e
4b	*BioSPI_BSPDetach_InvalidBSPHandle*	0434c458-0c79-1085-9f2c-0002a5d5fd2e
4c	*BioSPI_BSPDetach_Confirm*	002e7e58-0c78-1085-9e1d-0002a5d5fd2e
5a	*BioSPI_FreeBIRHandle_ValidParam*	0280a7d0-0c80-1085-a9a0-0002a5d5fd2e
5b	*BioSPI_FreeBIRHandle_InvalidBSPHandle*	047aed48-0c80-1085-898b-0002a5d5fd2e
5c	*BioSPI_FreeBIRHandle_InvalidBIRHandle*	018e6c18-0c9c-1085-afdf-0002a5d5fd2e
6a	*BioSPI_GetBIRFromHandle_ValidParam*	0460b658-0cb4-1085-a304-0002a5d5fd2e
6b	*BioSPI_GetBIRFromHandle_ InvalidBSPHandle*	02445668-0cc5-1085-a3ac-0002a5d5fd2e
6c	*BioSPI_GetBIRFromHandle_ InvalidBIRHandle*	0194a9c0-0cc7-1085-8780-0002a5d5fd2e

7a	*BioSPI_GetHeaderFromHandle_ ValidParam*	027a7db0-0cc7-1085-9391-0002a5d5fd2e
7b	*BioSPI_GetHeaderFromHandle_ InvalidBSPHandle*	057e0d38-0ccd-1085-83b8-0002a5d5fd2e
7c	*BioSPI_GetHeaderFromHandle_ InvalidBIRHandle*	02195e68-0cce-1085-a46f-0002a5d5fd2e
7d	*BioSPI_GetHeaderFromHandle_ BIRHandleNotFreed*	01cc0988-0ccf-1085-a367-0002a5d5fd2e
8a	*BioSPI_EnableEvents_ValidParam*	0333f628-0ccf-1085-aceb-0002a5d5fd2e
8b	*BioSPI_EnableEvents_InvalidBSPHandle*	04ed0838-0ccf-1085-b64e-0002a5d5fd2e
10a	*BioSPI_CreateTemplate_PayloadSupported*	04a01118-0cf9-1085-96d4-0002a5d5fd2e
10b	*BioSPI_CreateTemplate_ BIRHeaderQuality*	00b5c728-0cfb-1085-8969-0002a5d5fd2e
10c	*BioSPI_CreateTemplate_ OutputBIRDataType*	0193c730-0cf9-1085-b0a3-0002a5d5fd2e
10d	*BioSPI_CreateTemplate_ OutputBIRPurpose*	03dbdaa0-0cf2-1085-99ed-0002a5d5fd2e
10e	*BioSPI_CreateTemplate_ InputBIRDataType*	6d543ea0-2ce9-11d9-9669-0800200c9a66
10f	*BioSPI_CreateTemplate_ Inconsistent_Purpose*	28ec1620-e995-11d9-b1d1-0002a5d5c51b
11a	*BioSPI_Process_ValidParam*	4ec34700-e9a0-11d9-8fc8-0002a5d5c51b
11b	*BioSPI_Process_BIRHeaderQuality*	211668e0-e9a6-11d9-bcc8-0002a5d5c51b
11c	*BioSPI_Process_OutputBIRPurpose*	e1bb4f20-ed61-11d9-9344-0002a5d5c51b
11d	*BioSPI_Process_BuildsProcessedBIR*	f2ce6540-ed66-11d9-9618-0002a5d5c51b
11e	*BioSPI_Process_InputBIRDataType*	3cf96080-ed6b-11d9-9acf-0002a5d5c51b
12a	*BioSPI_VerifyMatch_ValidParam*	688aad60-ee30-11d9-a62c-0002a5d5c51b
12b	*BioSPI_VerifyMatch_Payload*	692ebe20-ee47-11d9-bd34-0002a5d5c51b
12c	*BioSPI_VerifyMatch_ Inconsistent_Purpose*	9108ec70-2e9b-11d9-9669-0800200c9a66

7.5.3 If an implementation claims to support the function *BioSPI_Capture*, the implementation shall be tested by executing all of the following assertions (in order):

Table 23 — Assertions for Testing Verification Engine BSP - BioSPI_Capture

Number	Assertion name	Package
9a	*BioSPI_Capture_AuditData*	02704c50-0cd8-1085-96cb-0002a5d5fd2e
9b	*BioSPI_Capture_ReturnQuality*	03f601f0-0cd8-1085-bd59-0002a5d5fd2e
9c	*BioSPI_Capture_ IntermediateProcessedBIR*	055ddb08-0cd6-1085-a6d3-0002a5d5fd2e
9d	*BioSPI_Capture_InvalidBSPHandle*	0244f2a8-0cf2-1085-8443-0002a5d5fd2e

7.5.4 If an implementation claims to support the function *BioSPI_Enroll*, the implementation shall be tested by executing all of the following assertions (in order):

Table 24 — Assertions for Testing Verification Engine BSP - BioSPI_Enroll

Number	Assertion name	Package
13a	*BioSPI_Enroll_ValidParam*	0b5ebb60-eefb-11d9-990c-0002a5d5c51b
13b	*BioSPI_Enroll_Payload*	e8969d40-ef05-11d9-9098-0002a5d5c51b
13c	*BioSPI_Enroll_AuditData*	b40a5260-ef14-11d9-a4fe-0002a5d5c51b
13d	*BioSPI_Enroll_BIRHeaderQuality*	6f727320-ef1a-11d9-9143-0002a5d5c51b

7.5.5 If an implementation claims to support the function *BioSPI_Verify*, the implementation shall be tested by executing all of the following assertions (in order):

Table 25 — Assertions for Testing Verification Engine BSP - BioSPI_Verify

Number	Assertion name	Package
14a	*BioSPI_Verify_ValidParam*	b78e5be0-efcb-11d9-b2c7-0002a5d5c51b
14b	*BioSPI_Verify_Payload*	32969ec0-eff8-11d9-9831-0002a5d5c51b
14c	*BioSPI_Verify_AuditData*	89719700-f218-11d9-b028-0002a5d5c51b

7.5.6 If an implementation claims to support a BSP-controlled database (see 19784-1:2006, A.4.6.1.2), the implementation shall be tested by executing all of the following assertions (in order):

Table 26 — Assertions for Testing Verification Engine BSP - Database Functions

Number	Assertion name	Package
15a	*BioSPI_DbOpen_ValidParam*	e68ff9a0-e506-11d9-a6a1-0002a5d5c51b
15b	*BioSPI_DbOpen_InvalidBSPHandle*	bfd44400-e5de-11d9-bdb9-0002a5d5c51b
16a	*BioSPI_DbClose_ValidParam*	39aa9560-e5f1-11d9-89f3-0002a5d5c51b
16b	*BioSPI_DbClose_InvalidBSPHandle*	6e3f5c00-e5f3-11d9-b663-0002a5d5c51b
17a	*BioSPI_DbCreate_DbProtected*	7b6c2f40-e650-11d9-812f-0002a5d5c51b
17b	*BioSPI_DbCreate_ValidParam*	1421ec38-1db6-49d4-873d-03e2de17598b
17c	*BioSPI_DbCreate_InvalidBSPHandle*	ef4bb862-79f6-4f01-8f5d-af5c3abf23c0
18a	*BioSPI_DbDelete_InvalidBSPHandle*	678e5d12-3d51-41ec-a672-13f34ea24545
18b	*BioSPI_DbDelete_OpenDbProtected*	9e4d0c6d-4d59-479c-9f58-160ecde99aad
18c	*BioSPI_DbDelete_ValidParam*	499d9cc3-4269-4671-9d69-29a31bc1a08f
19a	*BioSPI_DbSetMarker_ValidParam*	94271080-e723-11d9-898c-0002a5d5c51b
19b	*BioSPI_DbSetMarker_InvalidBSPHandle*	69f7ce20-e72e-11d9-b4e0-0002a5d5c51b
19c	*BioSPI_DbSetMarker_RecordNotFound*	b9c90f40-e7ec-11d9-a435-0002a5d5c51b
20a	*BioSPI_DbFreeMarker_ValidParam*	2e1c9520-e7f1-11d9-a011-0002a5d5c51b
20b	*BioSPI_DbFreeMarker_InvalidBSPHandle*	0f735140-e800-11d9-8a8a-0002a5d5c51b
20c	*BioSPI_DbFreeMarker_InvalidMarker*	a9007b60-e802-11d9-8a5d-0002a5d5c51b
21a	*BioSPI_DbStoreBIR_ValidParam*	da953680-e806-11d9-90d3-0002a5d5c51b
21b	*BioSPI_DbStoreBIR_InvalidBSPHandle*	e39027e0-e80b-11d9-85eb-0002a5d5c51b

22a	*BioSPI_DbGetBIR_ValidParam*	67512700-e811-11d9-a5e0-0002a5d5c51b
22b	*BioSPI_DbGetBIR_InvalidBSPHandle*	f62947e0-e8b7-11d9-9dad-0002a5d5c51b
22c	*BioSPI_DbGetBIR_RecordNotFound*	37457440-e8ba-11d9-87da-0002a5d5c51b
23a	*BioSPI_DbGetNextBIR_ValidParam*	e3396400-e8c9-11d9-990f-0002a5d5c51b
23b	*BioSPI_DbGetNextBIR_InvalidBSPHandle*	f0ef5320-e8d3-11d9-bbc1-0002a5d5c51b
24a	*BioSPI_DbDeleteBIR_ValidParam*	ed4afbe0-e8d6-11d9-9ed0-0002a5d5c51b
24b	*BioSPI_DbDeleteBIR_InvalidBSPHandle*	72eca940-e8d9-11d9-aa0d-0002a5d5c51b

7.6 Testing BSPs of subclass "Identification Engine"

7.6.1 The following subclauses specify which test assertions shall be used to determine whether a BSP of the conformance subclass "Identification Engine" satisfies the requirements specified in ISO/IEC 19784-1:2006, A.4 and A.4.5.

7.6.2 All BSPs of this subclass shall be tested by executing all of the following assertions (in order):

Table 27 — Assertions for Testing Identification Engine BSP

Number	Assertion name	Package
1a	*BioSPI_BSPLoad_InvalidUUID*	020e90c8-0c19-1085-ab54-0002a5d5fd2e
1b	*BioSPI_BSPLoad_ValidParam*	01f6c6f0-0c19-1085-97fe-0002a5d5fd2e
2a	*BioSPI_BSPUnload_ValidParam*	01661010-0c22-1085-8688-0002a5d5fd2e
2b	*BioSPI_BSPUnload_InvalidUUID*	01c2e5b0-0c3b-1085-b31d-0002a5d5fd2e
2c	*BioSPI_BSPUnload_UnmatchedLoad*	02f6c618-0c23-1085-ba89-0002a5d5fd2e
2d	*BioSPI_BSPUnload_Confirm*	03daf040-0c3b-1085-a9fd-0002a5d5fd2e
3a	*BioSPI_BSPAttach_ValidParam*	00ae6488-0c3d-1085-9912-0002a5d5fd2e
3b	*BioSPI_BSPAttach_InvalidUUID*	049cc170-0c5f-1085-981f-0002a5d5fd2e
3c	*BioSPI_BSPAttach_InvalidVersion*	0052ac10-0c60-1085-9883-0002a5d5fd2e
3d	*BioSPI_BSPAttach_InvalidBSPHandle*	03826830-0c57-1085-bfb0-0002a5d5fd2e

4a	*BioSPI_BSPDetach_ValidParam*	00e0d2b0-0c7a-1085-b8ac-0002a5d5fd2e
4b	*BioSPI_BSPDetach_InvalidBSPHandle*	0434c458-0c79-1085-9f2c-0002a5d5fd2e
4c	*BioSPI_BSPDetach_Confirm*	002e7e58-0c78-1085-9e1d-0002a5d5fd2e
5a	*BioSPI_FreeBIRHandle_ValidParam*	0280a7d0-0c80-1085-a9a0-0002a5d5fd2e
5b	*BioSPI_FreeBIRHandle_InvalidBSPHandle*	047aed48-0c80-1085-898b-0002a5d5fd2e
5c	*BioSPI_FreeBIRHandle_InvalidBIRHandle*	018e6c18-0c9c-1085-afdf-0002a5d5fd2e
6a	*BioSPI_GetBIRFromHandle_ValidParam*	0460b658-0cb4-1085-a304-0002a5d5fd2e
6b	*BioSPI_GetBIRFromHandle_ InvalidBSPHandle*	02445668-0cc5-1085-a3ac-0002a5d5fd2e
6c	*BioSPI_GetBIRFromHandle_ InvalidBIRHandle*	0194a9c0-0cc7-1085-8780-0002a5d5fd2e
7a	*BioSPI_GetHeaderFromHandle_ ValidParam*	027a7db0-0cc7-1085-9391-0002a5d5fd2e
7b	*BioSPI_GetHeaderFromHandle_ InvalidBSPHandle*	057e0d38-0ccd-1085-83b8-0002a5d5fd2e
7c	*BioSPI_GetHeaderFromHandle_ InvalidBIRHandle*	02195e68-0cce-1085-a46f-0002a5d5fd2e
7d	*BioSPI_GetHeaderFromHandle_ BIRHandleNotFreed*	01cc0988-0ccf-1085-a367-0002a5d5fd2e
8a	*BioSPI_EnableEvents_ValidParam*	0333f628-0ccf-1085-aceb-0002a5d5fd2e
8b	*BioSPI_EnableEvents_InvalidBSPHandle*	04ed0838-0ccf-1085-b64e-0002a5d5fd2e

10a	*BioSPI_CreateTemplate_PayloadSupported*	04a01118-0cf9-1085-96d4-0002a5d5fd2e
10b	*BioSPI_CreateTemplate_ BIRHeaderQuality*	00b5c728-0cfb-1085-8969-0002a5d5fd2e
10c	*BioSPI_CreateTemplate_ OutputBIRDataType*	0193c730-0cf9-1085-b0a3-0002a5d5fd2e
10d	*BioSPI_CreateTemplate_ OutputBIRPurpose*	03dbdaa0-0cf2-1085-99ed-0002a5d5fd2e
10e	*BioSPI_CreateTemplate_ InputBIRDataType*	6d543ea0-2ce9-11d9-9669-0800200c9a66
10f	*BioSPI_CreateTemplate_ Inconsistent_Purpose*	28ec1620-e995-11d9-b1d1-0002a5d5c51b
11a	*BioSPI_Process_ValidParam*	4ec34700-e9a0-11d9-8fc8-0002a5d5c51b
11b	*BioSPI_Process_BIRHeaderQuality*	211668e0-e9a6-11d9-bcc8-0002a5d5c51b
11c	*BioSPI_Process_OutputBIRPurpose*	e1bb4f20-ed61-11d9-9344-0002a5d5c51b
11d	*BioSPI_Process_BuildsProcessedBIR*	f2ce6540-ed66-11d9-9618-0002a5d5c51b
11e	*BioSPI_Process_InputBIRDataType*	3cf96080-ed6b-11d9-9acf-0002a5d5c51b
12a	*BioSPI_VerifyMatch_ValidParam*	688aad60-ee30-11d9-a62c-0002a5d5c51b
12b	*BioSPI_VerifyMatch_Payload*	692ebe20-ee47-11d9-bd34-0002a5d5c51b
12c	*BioSPI_VerifyMatch_ Inconsistent_Purpose*	9108ec70-2e9b-11d9-9669-0800200c9a66

7.6.3 If an implementation claims to support the function *BioSPI_Capture*, the implementation shall be tested by executing all of the following assertions (in order):

Table 28 — Assertions for Testing Identification Engine BSP - BioSPI_Capture

Number	Assertion name	Package
9a	*BioSPI_Capture_AuditData*	02704c50-0cd8-1085-96cb-0002a5d5fd2e
9b	*BioSPI_Capture_ReturnQuality*	03f601f0-0cd8-1085-bd59-0002a5d5fd2e
9c	*BioSPI_Capture_ IntermediateProcessedBIR*	055ddb08-0cd6-1085-a6d3-0002a5d5fd2e
9d	*BioSPI_Capture_InvalidBSPHandle*	0244f2a8-0cf2-1085-8443-0002a5d5fd2e

7.6.4 If an implementation claims to support the function *BioSPI_Enroll*, the implementation shall be tested by executing all of the following assertions (in order):

Table 29 — Assertions for Testing Identification Engine BSP - BioSPI_Enroll

Number	Assertion name	Package
13a	*BioSPI_Enroll_ValidParam*	0b5ebb60-eefb-11d9-990c-0002a5d5c51b
13b	*BioSPI_Enroll_Payload*	e8969d40-ef05-11d9-9098-0002a5d5c51b
13c	*BioSPI_Enroll_AuditData*	b40a5260-ef14-11d9-a4fe-0002a5d5c51b
13d	*BioSPI_Enroll_BIRHeaderQuality*	6f727320-ef1a-11d9-9143-0002a5d5c51b

7.6.5 If an implementation claims to support the function *BioSPI_Verify*, the implementation shall be tested by executing all of the following assertions (in order):

Table 30 — Assertions for Testing Identification Engine BSP - BioSPI_Verify

Number	Assertion name	Package
14a	*BioSPI_Verify_ValidParam*	b78e5be0-efcb-11d9-b2c7-0002a5d5c51b
14b	*BioSPI_Verify_Payload*	32969ec0-eff8-11d9-9831-0002a5d5c51b
14c	*BioSPI_Verify_AuditData*	89719700-f218-11d9-b028-0002a5d5c51b

7.6.6 If an implementation claims to support a BSP-controlled database (see 19784-1:2006, A.4.6.1.2), the implementation shall be tested by executing all of the following assertions (in order):

Table 31 — Assertions for Testing Identification Engine BSP - Database Functions

Number	Assertion name	Package
15a	*BioSPI_DbOpen_ValidParam*	e68ff9a0-e506-11d9-a6a1-0002a5d5c51b
15b	*BioSPI_DbOpen_InvalidBSPHandle*	bfd44400-e5de-11d9-bdb9-0002a5d5c51b
16a	*BioSPI_DbClose_ValidParam*	39aa9560-e5f1-11d9-89f3-0002a5d5c51b
16b	*BioSPI_DbClose_InvalidBSPHandle*	6e3f5c00-e5f3-11d9-b663-0002a5d5c51b
17a	*BioSPI_DbCreate_DbProtected*	7b6c2f40-e650-11d9-812f-0002a5d5c51b
17b	*BioSPI_DbCreate_ValidParam*	1421ec38-1db6-49d4-873d-03e2de17598b
17c	*BioSPI_DbCreate_InvalidBSPHandle*	ef4bb862-79f6-4f01-8f5d-af5c3abf23c0

18a	*BioSPI_DbDelete_InvalidBSPHandle*	678e5d12-3d51-41ec-a672-13f34ea24545
18b	*BioSPI_DbDelete_OpenDbProtected*	9e4d0c6d-4d59-479c-9f58-160ecde99aad
18c	*BioSPI_DbDelete_ValidParam*	499d9cc3-4269-4671-9d69-29a31bc1a08f
19a	*BioSPI_DbSetMarker_ValidParam*	94271080-e723-11d9-898c-0002a5d5c51b
19b	*BioSPI_DbSetMarker_InvalidBSPHandle*	69f7ce20-e72e-11d9-b4e0-0002a5d5c51b
19c	*BioSPI_DbSetMarker_RecordNotFound*	b9c90f40-e7ec-11d9-a435-0002a5d5c51b
20a	*BioSPI_DbFreeMarker_ValidParam*	2e1c9520-e7f1-11d9-a011-0002a5d5c51b
20b	*BioSPI_DbFreeMarker_InvalidBSPHandle*	0f735140-e800-11d9-8a8a-0002a5d5c51b
20c	*BioSPI_DbFreeMarker_InvalidMarker*	a9007b60-e802-11d9-8a5d-0002a5d5c51b
21a	*BioSPI_DbStoreBIR_ValidParam*	da953680-e806-11d9-90d3-0002a5d5c51b
21b	*BioSPI_DbStoreBIR_InvalidBSPHandle*	e39027e0-e80b-11d9-85eb-0002a5d5c51b
22a	*BioSPI_DbGetBIR_ValidParam*	67512700-e811-11d9-a5e0-0002a5d5c51b
22b	*BioSPI_DbGetBIR_InvalidBSPHandle*	f62947e0-e8b7-11d9-9dad-0002a5d5c51b
22c	*BioSPI_DbGetBIR_RecordNotFound*	37457440-e8ba-11d9-87da-0002a5d5c51b
23a	*BioSPI_DbGetNextBIR_ValidParam*	e3396400-e8c9-11d9-990f-0002a5d5c51b
23b	*BioSPI_DbGetNextBIR_InvalidBSPHandle*	f0ef5320-e8d3-11d9-bbc1-0002a5d5c51b
24a	*BioSPI_DbDeleteBIR_ValidParam*	ed4afbe0-e8d6-11d9-9ed0-0002a5d5c51b
24b	*BioSPI_DbDeleteBIR_InvalidBSPHandle*	72eca940-e8d9-11d9-aa0d-0002a5d5c51b

8 Test assertions

8.1 General

8.1.1 This clause 8 specifies all the test assertions relative to BSPs, and includes assertions for BSPs of all conformance subclasses.

8.1.2 A package containing activities that are common to most of the test assertions, is specified in the following subclause 8.2.

8.2 Common activities

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

```
<package name="02c59458-0c46-1085-95d7-0002a5d5fd2e">
      <author>
            ISO/IEC JTC1 SC37
      </author>

      <description>
            This package contains several useful activities that are invoked by activities in other
                              packages.
      </description>

      <activity name="LoadAndAttach">
            <input name="bspUuid"/>
            <input name="bspVersion"/>
            <input name="unitIDOrNull"/>
            <input name="bspHandle"/>
            <input name="eventtimeouttime"/>

            <!-- Initialize the global variable "_unitIDOrNull" to make it available to the activity
                              "EventHandler" -->
            <set name="_unitIDOrNull" var="unitIDOrNull"/>

            <!-- Initialize the global variable "_insert" to "false".  The activity "EventHandler"
                              will set this variable to "true" when a BioAPI_NOTIFY_INSERT event
                              notification is received, and will set it to "false" when a
                              BioAPI_NOTIFY_REMOVE event notification is received. -->
            <set name="_insert" value="false"/>

            <!-- Initialize the "_sourcePresent" global variable to "false".  The activity
                              "EventHandler" will set this variable to "true" when a
                              BioAPI_NOTIFY_SOURCE_PRESENT event notification is received, and will
                              set it to "false" when a BioAPI_NOTIFY_SOURCE_REMOVED event notification
                              is received. -->
            <set name="_sourcePresent" value="false"/>

            <set name="eventtimeoutflag" value="false"/>

            <!-- Invoke the function BioSPI_BSPLoad. -->
            <invoke function="BioSPI_BSPLoad">
                  <input name="BSPUuid" var="bspUuid"/>
                  <input name="BioAPINotifyCallback" value="*"/>
                  <input name="BFPEnumerationHandler" value="*"/>
                  <input name="MemoryFreeHandler" value="*" />
                  <return setvar="return"/>
            </invoke>

            <!-- Issue a conformity response.
                        If the condition specified in the <description> below is false, an UNDECIDED
                              conformity response is issued and the execution of the activity is
                              interrupted, otherwise a PASS conformity response is issued.-->
            <assert_condition response_if_false="undecided"
                        break_if_false="true">
                  <description>
                        The function BioSPI_BSPLoad has returned BioAPI_OK.
                  </description>
                  <equal_to var1="return" var2="__BioAPI_OK"/>
            </assert_condition>
```

```
        <!-- Wait until the BioAPI_NOTIFY_INSERT event notification has been received, but no
                         longer than the specified maximum duration.-->
        <wait_until timeout_var="eventtimeouttime"
                   setvar="eventtimeoutflag" var="_insert"/>
        <!-- Issue a conformity response.
                   If the condition specified in the <description> below is false, an UNDECIDED
                         conformity response is issued and the execution of the activity is
                         interrupted, otherwise a PASS conformity response is issued.-->
        <assert_condition response_if_false="undecided"
                   break_if_false="true">
              <description>
                   The BioAPI_NOTIFY_INSERT event notification has been received within the
                         specified maximum duration
              </description>
              <not var="eventtimeoutflag"/>
        </assert_condition>

        <!-- Invoke the function BioSPI_BSPAttach. -->
        <invoke function="BioSPI_BSPAttach">
              <input name="BSPUuid" var="bspUuid"/>
              <input name="Version" var="bspVersion"/>
              <input name="Unit_1_UnitCategory" var="_unitCategory" />
              <input name="Unit_1_UnitID" var="_unitID"/>
              <input name="NumUnits" value="1" />
              <input name="BSPHandle" var="bspHandle"/>
              <return setvar="return"/>
        </invoke>

        <!-- Issue a conformity response.
                   If the condition specified in the <description> below is false, an UNDECIDED
                         conformity response is issued and the execution of the activity is
                         interrupted, otherwise a PASS conformity response is issued.-->
        <assert_condition response_if_false="undecided"
                   break_if_false="true">
              <description>
                   The function BioSPI_BSPAttach has returned BioAPI_OK.
              </description>
              <equal_to var1="return" var2="__BioAPI_OK"/>
        </assert_condition>
  </activity>

  <!-- This activity will be invoked on incoming calls to the function BioSPI_ModuleEventHandler
                         exposed by the testing component.  In this activity, the global
                         variables "_unitID", "_insert", "_sourcePresent" and "_eventtype" are
                         set depending on the input parameter values. -->
  <activity name="EventHandler" atomic="true">
        <input name="BSPUuid"/>
        <input name="UnitID"/>
        <input name="UnitSchema_BspUuid" />
        <input name="UnitSchema_UnitManagerUuid" />
        <input name="UnitSchema_UnitId" />
        <input name="UnitSchema_UnitCategory" />
        <input name="UnitSchema_UnitProperties" />
        <input name="UnitSchema_VendorInformation" />
        <input name="UnitSchema_EventNotifyInsert" />
        <input name="UnitSchema_EventNotifyRemove" />
        <input name="UnitSchema_EventNotifyFault" />
        <input name="UnitSchema_EventNotifySourcePresent" />
        <input name="UnitSchema_EventNotifySourceRemoved" />
        <input name="UnitSchema_UnitPropertyID" />
        <input name="UnitSchema_UnitProperty" />
        <input name="UnitSchema_HardwareVersion" />
        <input name="UnitSchema_FirmwareVersion" />
        <input name="UnitSchema_SoftwareVersion" />
        <input name="UnitSchema_HardwareSerialNumber" />
        <input name="UnitSchema_AuthenticatedHardware" />
        <input name="UnitSchema_MaxBspDbSize" />
        <input name="UnitSchema_MaxIdentify" />
        <input name="EventType" />
        <output name="return"/>
```

```
      <!-- Set the global variable "_unitID" if:
                  - it is not set; and
                  - the event notification is either BioAPI_NOTIFY_INSERT or
                        BioAPI_NOTIFY_SOURCE_PRESENT; and
                  - the event is related to the expected unit, as specified by the parameter
                        "unitIDOrNULL" of the "LoadAndAttach" activity invocation. -->
      <set name="_unitID" var="UnitID">
            <only_if>
                  <not>
                        <existing var="_unitID"/>
                  </not>
                  <or>
                        <equal_to var1="EventType"
                                    var2="__BioAPI_NOTIFY_INSERT"/>
                        <equal_to var1="EventType"
                                    var2="__BioAPI_NOTIFY_SOURCE_PRESENT"/>
                  </or>
                  <or>
                        <equal_to var1="_unitIDOrNull" value2="0"/>
                        <equal_to var1="_unitIDOrNull" var2="UnitID"/>
                  </or>
            </only_if>
      </set>

      <!-- set the unit category -->
      <set name="_unitCategory" var="UnitSchema_UnitCategory">
            <only_if>
                  <not>
                        <existing var="_unitCategory"/>
                  </not>
                  <equal_to var1="EventType"
                              var2="__BioAPI_NOTIFY_INSERT"/>
                  <or>
                        <equal_to var1="_unitIDOrNull" value2="0"/>
                        <equal_to var1="_unitIDOrNull" var2="UnitID"/>
                  </or>
            </only_if>
      </set>

      <invoke activity="EventHandlerSetGlobalData">
            <only_if>
                  <existing var="_unitID"/>
            </only_if>
            <input name="UnitID" var="UnitID"/>
            <input name="EventType" var="EventType"/>
      </invoke>

      <set name="return" var="__BioAPI_OK"/>
</activity>

<activity name="EventHandlerSetGlobalData" atomic="true">
      <input name="UnitID"/>
      <input name="EventType"/>

      <!-- Set the global variable "_insert" to "true" if:
                  - the event notification is BioAPI_NOTIFY_INSERT; and
                  - the event is related to the expected unit, as specified by the parameter
                        "unitIDOrNULL" of the "LoadAndAttach" activity invocation. -->
      <set name="_insert" value="true">
            <only_if>
                  <equal_to var1="EventType" var2="__BioAPI_NOTIFY_INSERT"/>
                  <equal_to var1="_unitID" var2="UnitID"/>
            </only_if>
      </set>

      <!-- Set the global variable "_insert" to "false" if:
                  - the event notification is BioAPI_NOTIFY_REMOVE; and
                  - the event is related to the expected unit, as specified by the parameter
                        "unitIDOrNULL" of the "LoadAndAttach" activity invocation. -->
      <set name="_insert" value="false">
            <only_if>
                  <equal_to var1="EventType" var2="__BioAPI_NOTIFY_REMOVE"/>
                  <equal_to var1="_unitID" var2="UnitID"/>
            </only_if>
      </set>
```

```
        <!-- Set the global variable "_sourcePresent" to "true" if:
                    - the event notification is BioAPI_NOTIFY_SOURCE_PRESENT; and
                    - the event is related to the expected unit, as specified by the parameter
                          "unitIDOrNULL" of the "LoadAndAttach" activity invocation. -->
        <set name="_sourcePresent" value="true">
              <only_if>
                    <equal_to var1="EventType"
                                var2="__BioAPI_NOTIFY_SOURCE_PRESENT"/>
                    <equal_to var1="_unitID" var2="UnitID"/>
              </only_if>
        </set>

        <!-- Set the global variable "_sourcePresent" to "false" if:
                    - the event notification is BioAPI_NOTIFY_SOURCE_REMOVED; and
                    - the event is related to the expected unit, as specified by the parameter
                          "unitIDOrNULL" of the "LoadAndAttach" activity invocation. -->
        <set name="_sourcePresent" value="false">
              <only_if>
                    <equal_to var1="EventType"
                                var2="__BioAPI_NOTIFY_SOURCE_REMOVED"/>
                    <equal_to var1="_unitID" var2="UnitID"/>
              </only_if>
        </set>

        <!-- Set the global variable "_eventtype" -->
        <set name="_eventtype" var="EventType">
              <only_if>
                    <equal_to var1="_unitID" var2="UnitID"/>
              </only_if>
        </set>
  </activity>

  <!-- This activity invokes the functions BioSPI_BSPDetach and BioSPI_BSPUnload. -->
  <activity name="DetachAndUnload" >
        <input name="bspUuid"/>
        <input name="BSPHandle"/>

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

        <!-- Issue a conformity response.
                    If the condition specified in the <description> below is false, an UNDECIDED
                          conformity response is issued and the execution of the activity is
                          interrupted, otherwise a PASS conformity response is issued . -->
        <assert_condition response_if_false="undecided"
                    break_if_false="true">
              <description>
                    The function BioSPI_BSPDetach has returned BioAPI_OK.
              </description>
              <equal_to var1="return" var2="__BioAPI_OK"/>
        </assert_condition>

        <!-- Invoke the function BioSPI_BSPUnload.-->
        <invoke function="BioSPI_BSPUnload" >
              <input name="BSPUuid" var="bspUuid" />
              <return setvar="return"/>
        </invoke>

        <!-- Issue a conformity response.
                    If the condition specified in the <description> below is false, an UNDECIDED
                          conformity response is issued, otherwise a PASS conformity response is
                          issued.-->
        <assert_condition response_if_false="undecided"
                    break_if_false="true">
              <description>
                    The function BioSPI_BSPUnload has returned BioAPI_OK.
              </description>
              <equal_to var1="return" var2="__BioAPI_OK"/>
        </assert_condition>
  </activity>
```

```
<activity name="check_capturedBIR_datatype">
      <input name="BSPHandle" />
      <input name="BIRHandle" />

      <!-- Invoke the function BioSPI_GetHeaderFromHandle on the captured BIR. -->
      <invoke function="BioSPI_GetHeaderFromHandle">
            <input name="BSPHandle" var="BSPHandle"/>
            <input name="Handle" var="BIRHandle"/>
            <output name="ProcessedLevel" setvar="processedLevel"/>
            <return setvar="return"/>
      </invoke>

      <!-- Issue a conformity response.
                  If the condition specified in the <description> below is false, an UNDECIDED
                        conformity response is issued and the execution of the activity is
                        interrupted, otherwise a PASS conformity response is issued.-->
      <assert_condition response_if_false="undecided"
                  break_if_false="true">
            <description>
                        The function BioSPI_GetHeaderFromHandle has returned BioAPI_OK
            </description>
            <equal_to var1="return" var2="__BioAPI_OK"/>
      </assert_condition>

      <!-- Issue a conformity response.
                  If the condition specified in the <description> below is false, an UNDECIDED
                        conformity response is issued and the execution of the activity is
                        interrupted, otherwise a PASS conformity response is issued.-->
      <assert_condition response_if_false="undecided"
                  break_if_false="true">
      <description>
            The processed level of the captured BIR is different from PROCESSED.
      </description>
      <not_equal_to var1="processedLevel"
                  var2="__BioAPI_BIR_DATA_TYPE_PROCESSED" />
      </assert_condition>
</activity>

<activity name="process_bir">
      <input name="BSPHandle" />
      <input name="CapturedBIR_BIRHandle" />

      <!-- Invoke the function BioSPI_Process. -->
      <invoke function="BioSPI_Process">
            <input name="BSPHandle" var="BSPHandle"/>
            <input name="CapturedBIR_Form"
                        var="__BioAPI_BIR_HANDLE_INPUT" />
            <input name="CapturedBIR_BIRHandle"
                        var="CapturedBIR_BIRHandle"/>
            <output name="ProcessedBIR" setvar="_processedbir_handle"/>
            <return setvar="return"/>
      </invoke>

      <!-- Issue a conformity response.
                  If the condition specified in the <description> below is false, an UNDECIDED
                        conformity response is issued, otherwise a PASS conformity response is
                        issued.-->
      <assert_condition response_if_false="undecided"
                  break_if_false="true">
            <description>
                  The function BioSPI_Process has returned BioAPI_OK
            </description>
            <equal_to var1="return" var2="__BioAPI_OK"/>
      </assert_condition>
</activity>


<activity name="payloadSupport_checkPayload" >
      <input name="inputPayload"/>
      <input name="outputPayload"/>
      <input name="result"/>
      <input name="payloadPolicy"/>
      <input name="fmrAchieved"/>
```

```
        <invoke activity="resultFalse_checkPayload"
                break_on_break="true">
            <only_if>
                <same_as var1="result" value2="false"/>
            </only_if>
            <input name="outputPayload"  var="outputPayload" />
        </invoke>

        <invoke activity="resultTrue_checkPayload"
                break_on_break="true">
            <only_if>
                <same_as var1="result" value2="true" />
            </only_if>
            <input name="inputPayload" var="inputPayload" />
            <input name="outputPayload" var="outputPayload" />
            <input name="payloadPolicy" var="payloadPolicy" />
            <input name="fmrAchieved" var="fmrAchieved"/>
        </invoke>

    </activity>

    <activity name="payloadNotSupport_checkPayload" >
        <input name="outputPayload"/>

        <!-- Issue a conformity response.
                If the condition specified in the <description> below is false, a FAIL
                    conformity response is issued and the execution of the activity is
                    interrupted, otherwise a PASS conformity response is issued.-->
        <assert_condition break_if_false="true">
            <description>
                No payload has been returned.
            </description>
            <same_as var1="outputPayload" value2=""/>
        </assert_condition>

    </activity>

    <activity name="resultFalse_checkPayload" >
        <input name="outputPayload"/>

        <!-- Issue a conformity response.
                If the condition specified in the <description> below is false, a FAIL
                    conformity response is issued and the execution of the activity is
                    interrupted, otherwise a PASS conformity response is issued.-->
        <assert_condition break_if_false="true">
            <description>
                The BSP supports payload, the result is false, and no payload has been
                    returned.
            </description>
            <same_as var1="outputPayload" value2=""/>
        </assert_condition>
    </activity>

    <activity name="resultTrue_checkPayload">
        <input name="inputPayload"/>
        <input name="outputPayload"/>
        <input name="payloadPolicy"/>
        <input name="fmrAchieved"/>

        <!-- Issue a conformity response.
                If the condition specified in the <description> below is false, an UNDECIDED
                    conformity response is issued and the execution of the activity is
                    interrupted, otherwise a PASS conformity response is issued.-->
        <assert_condition response_if_false="undecided"
                break_if_false="true">
            <description>
                The returned payload is the same as the provided payload or no payload has
                    been returned because the achieved FMR is too high.
            </description>
            <or>
                <and>
                    <less_than var1="fmrAchieved" var2="payloadPolicy"/>
                    <same_as var1="outputPayload" var2="inputPayload"/>
                </and>
                <and>
                    <greater_than_or_equal_to var1="fmrAchieved" var2="payloadPolicy"/>
```

```
                        <same_as var1="outputPayload" value2=""/>
                  </and>
            </or>
      </assert_condition>
</activity>

<!-- This activity checks the value of the returned adapted BIR. It is used if the BSP claims
                        to support template adaptation. -->
<activity name="check_adaptation_supported">
      <input name="adaptedbir_handle" />

      <assert_condition response_if_false="undecided"
                  break_if_false="true" >
            <description>
                  The adapted BIR handle has a valid value.
            </description>
            <not_equal_to var1="adaptedbir_handle" value2="-1" />
      </assert_condition>

      <assert_condition break_if_false="true">
            <description>
                  The adapted BIR handle is non-negative.
            </description>
      <greater_than_or_equal_to var1="adaptedbir_handle"
                  value2="0"/>
      </assert_condition>
</activity>

<!-- This activity checks the value of the returned adapted BIR. It is used if the BSP does not
                        claim to support template adaptation. -->
<activity name="check_adaptation_not_supported" >
      <input name="adaptedbir_handle" />

      <assert_condition break_if_false="true">
            <description>
                  Adaptation is not supported by the BSP and the BSP has returned
                        BioAPI_UNSUPPORTED_BIR_HANDLE (-2).
            </description>
            <equal_to var1="adaptedbir_handle" value2="-2" />
      </assert_condition>
</activity>

<!-- This activity checks the value of the returned quality in case processed quality is
                        supported by the BSP. -->
<activity name="check_quality_supported" >
      <input name="quality"/>

      <!-- Issue a conformity response.
                  If the condition specified in the <description> below is false, an UNDECIDED
                        conformity response is issued and the execution of the activity is
                        interrupted, otherwise a PASS conformity response is issued.-->
      <assert_condition response_if_false="undecided"
                  break_if_false="true">
            <description>
                  Quality has a valid value.
            </description>
            <not_equal_to var1="quality" value2="-1"/>
      </assert_condition>

      <!-- Issue a conformity response.
                  If the condition specified in the <description> below is false, a FAIL
                        conformity response is issued and the execution of the activity is
                        interrupted, otherwise a PASS conformity response is issued.-->
      <assert_condition break_if_false="true">
            <description>
                  Quality is supported by the BSP and has a value between 0 and 100.
            </description>
            <greater_than_or_equal_to var1="quality" value2="0"/>
            <less_than_or_equal_to var1="quality" value2="100"/>
      </assert_condition>
</activity>

<!-- This activity checks the value of the returned quality in case processed quality is not
                        supported by the BSP. -->
<activity name="check_quality_not_supported" >
      <input name="quality"/>
```

```
        <assert_condition break_if_false="true">
              <description>
                    Quality is not  supported by the BSP and -2 is returned.
              </description>
              <equal_to var1="quality" value2="-2"/>
        </assert_condition>
  </activity>

  <!-- This activity create a database -->
  <activity name="PrepareDBTesting" >
        <input name="bspHandle" />
        <input name="dbUuid"/>
        <input name="nosourcepresentsupported" />
        <input name="sourcepresenttimeouttime"/>
        <output name="biruuid"/>

        <invoke function="BioSPI_DbDelete" >
              <input name="BSPHandle" var="bspHandle" />
              <input name="DbUuid" var="dbUuid" />
              <return setvar="return"/>
        </invoke>

        <invoke function="BioSPI_DbCreate">
              <input name="BSPHandle" var="bspHandle"/>
              <input name="DbUuid" var="dbUuid"/>
              <input name="NumberOfRecords" value="1"/>
              <input name="ReadAccessRequest" value="true"/>
              <input name="WriteAccessRequest" value="true"/>
              <output name="DbHandle" setvar="dbHandle"/>
              <return setvar="return"/>
        </invoke>

        <!-- Issue a conformity response.
                    If the condition specified in the <description> below is false, an UNDECIDED
                          conformity response is issued, otherwise a PASS conformity response is
                          issued.-->
        <assert_condition response_if_false="undecided"
                    break_if_false="true">
              <description>
                    The function BioSPI_DbCreate has returned BioAPI_OK
              </description>
              <equal_to var1="return" var2="__BioAPI_OK"/>
        </assert_condition>

        <set name="eventtimeoutflag" value="false"/>

        <!-- If the BSP under test claims support for the BioAPI_NOTIFY_SOURCE_PRESENT event
                          notification, wait until that notification has been received, but no
                          longer than the specified maximum duration.-->
        <wait_until timeout_var="sourcepresenttimeouttime"
                    setvar="eventtimeoutflag">
              <or var1="nosourcepresentsupported" var2="_sourcePresent" />
        </wait_until>

        <!-- Issue a conformity response.
                    If the condition specified in the <description> below is false, an UNDECIDED
                          conformity response is issued and the execution of the activity is
                          interrupted, otherwise a PASS conformity response is issued.-->
        <assert_condition response_if_false="undecided"
                    break_if_false="true">
              <description>
                    Either the BSP under test does not claim support for the
                          BioAPI_NOTIFY_SOURCE_PRESENT event notification, or the event
                          notification has been received within the specified maximum duration.
              </description>
              <not var="eventtimeoutflag"/>
        </assert_condition>

        <!-- The BSP is ready to capture. Invoke the function BioSPI_Enroll for the purpose of
                          enrollment. -->
        <invoke function="BioSPI_Enroll">
              <input name="BSPHandle" var="bspHandle"/>
              <input name="Purpose"
                          var="__BioAPI_PURPOSE_ENROLL_FOR_VERIFICATION_ONLY"/>
              <input name="Timeout" value="15000"/>
```

```
            <output name="NewTemplate" setvar="newtemplate_handle"/>
            <return setvar="return"/>
      </invoke>

      <!-- Issue a conformity response.
                  If the condition specified in the <description> below is false, an UNDECIDED
                        conformity response is issued and the execution of the activity is
                        interrupted, otherwise a PASS conformity response is issued.-->
      <assert_condition response_if_false="undecided"
                  break_if_false="true">
            <description>
                  The function BioSPI_Enroll has returned BioAPI_OK
            </description>
            <equal_to var1="return" var2="__BioAPI_OK"/>
      </assert_condition>

      <!-- Invoke the function BioSPI_DbStoreBIR to store the enrolled BIR into database -->
      <invoke function="BioSPI_DbStoreBIR">
            <input name="BSPHandle" var="bspHandle"/>
            <input name="BIRToStore_Form" var="__BioAPI_BIR_HANDLE_INPUT"/>
            <input name="BIRToStore_BIRHandle" var="newtemplate_handle"/>
            <input name="DbHandle" var="dbHandle"/>
            <input name="no_BirUuid" value="false"/>
            <output name="BirUuid" setvar="biruuid"/>
            <return setvar="return"/>
      </invoke>

      <!-- Issue a conformity response.
                  If the condition specified in the <description> below is false, a FAIL
                        conformity response is issued, otherwise a PASS conformity response is
                        issued.-->
      <assert_condition response_if_false="undecided"
                  break_if_false="true">
            <description>
                  The function BioSPI_DbStoreBIR has returned BioAPI_OK, and the output UUID is
                        not NULL.
            </description>
            <equal_to var1="return" var2="__BioAPI_OK"/>
            <not_same_as var1="biruuid" value2=""/>
      </assert_condition>

      <!-- Invoke the function BioSPI_DbClose with valid input parameters -->
      <invoke function="BioSPI_DbClose">
            <input name="BSPHandle" var="bspHandle"/>
            <input name="DbHandle" var="dbHandle"/>
            <return setvar="return"/>
      </invoke>

      <!-- Issue a conformity response.
                  If the condition specified in the <description> below is false, an UNDECIDED
                        conformity response is issued, otherwise a PASS conformity response is
                        issued.-->
      <assert_condition response_if_false="undecided"
                  break_if_false="true">
            <description>
                  The function BioSPI_DbClose has returned BioAPI_OK
            </description>
            <equal_to var1="return" var2="__BioAPI_OK"/>
      </assert_condition>
</activity>

<!-- This activity  delete a database -->
<activity name="CleanUpDBTesting" >
      <input name="BSPHandle" />
      <input name="dbUuid"/>

      <!-- Invoke the function BioSPI_DbDelete valid parameters -->
      <invoke function="BioSPI_DbDelete">
            <input name="BSPHandle" var="_bsphandle"/>
            <input name="DbUuid" var="dbUuid"/>
            <return setvar="return"/>
      </invoke>
```

```
            <!-- Issue a conformity response.
                        If the condition specified in the <description> below is false, an UNDECIDED
                              conformity response is issued, otherwise a PASS conformity response is
                              issued.-->
            <assert_condition response_if_false="undecided">
                  <description>
                        The function BioSPI_DbDelete has returned BioAPI_OK
                  </description>
                  <equal_to var1="return" var2="__BioAPI_OK"/>
            </assert_condition>

      </activity>
</package>
```

8.3 Assertion 1a - *BioSPI_BSPLoad_InvalidUUID*

Description: This assertion checks if calling the function BioSPI_BSPLoad with an invalid input parameter UUID returns BioAPIERR_H_FRAMEWORK_INVALID_UUID.

Excerpts:

Subclause 9.3.1.1

*BioAPI_RETURN BioAPI BioSPI_BSPLoad(const BioAPI_UUID *BSPUuid,*
BioSPI_EventHandler BioAPINotifyCallback,
BioSPI_BFP_ENUMERATION_HANDLER BFPEnumerationHandler,
BioSPI_MEMORY_FREE_HANDLER MemoryFreeHandler);

This function completes the module initialization process between BioAPI and the biometric service module.

Return Value

A BioAPI_RETURN value indicating success or specifying a particular error condition. The value BioAPI_OK indicates success. All other values represent an error condition.
The BSPUuid identifies the invoked module.

Subclause 11.2.3

Errors: BioAPIERR_INVALID_UUID.

References: 9.3.1.1 and 11.2.3

Scenario:

1) Call BioSPI_BSPLoad with an invalid input parameter UUID.

2) Check the return value. If it is BioAPIERR_INVALID_UUID, then issue a PASS conformity response, otherwise issue a FAIL conformity response.

If any of the intermediate operations fails, an UNDECIDED conformity response is issued.

Expected results: The call to BioSPI_BSPLoad returns BioAPIERR_INVALID_UUID.

Assertion language package

```
<package name="020e90c8-0c19-1085-ab54-0002a5d5fd2e">
      <author>
            ISO/IEC JTC1 SC37
      </author>

      <description>
            This package contains the assertion "BioSPI_BSPLoad_InvalidUUID" (see the "description"
                              element of the assertion below).
      </description>
      <assertion name="BioSPI_BSPLoad_InvalidUUID" model="BSPTesting">
            <description>
                  This assertion checks if calling the function BioSPI_BSPLoad with an invalid input
                              parameter UUID returns BioAPIERR_H_FRAMEWORK_INVALID_UUID.
                  The relevant text in BioAPI 2.0 is quoted below from subclauses 9.3.1.1 and 11.2.3.
                  ________________________________________________________
                  BioAPI_RETURN BioAPI BioSPI_BSPLoad(const BioAPI_UUID *BSPUuid,
                        BioSPI_EventHandler BioAPINotifyCallback,
                        BioSPI_BFP_ENUMERATION_HANDLER  BFPEnumerationHandler,
                        BioSPI_MEMORY_FREE_HANDLER  MemoryFreeHandler);

                  This function completes the module initialization process between BioAPI and the
                              biometric service module.

                  Return Value
                  A BioAPI_RETURN value indicating success or specifying a particular error
                              condition. The value BioAPI_OK indicates success. All other values
                              represent an error condition.
                  ________________________________________________________
                  Subclause 9.3.1.1:
                  The BSPUuid identifies the invoked module.
                  Subclause 11.2.3:
                  Errors: BioAPIERR_INVALID_UUID
                  ________________________________________________________

                  In order to determine conformance with respect to the text above, the following
                              steps are performed:

                        1)    Call BioSPI_BSPLoad with an invalid input parameter UUID.
                        2)    Check the return value.  If it is BioAPIERR_INVALID_UUID, then issue a
                              PASS conformity response, otherwise issue a FAIL conformity response.

                  If any of the intermediate operations fails, an UNDECIDED conformity response is
                              issued.
            </description>

            <!-- Invocation of the primary activity of this assertion with input parameter values
                              assigned from the assertion's parameters. -->
            <invoke activity="BioSPI_BSPLoad">
                  <input name="bspUuid" value="00000000-0000-0000-0000-000000000000"/>
            </invoke>
      </assertion>

      <activity name="BioSPI_BSPLoad">
            <input name="bspUuid"/>

            <!-- Invoke the function BioSPI_BSPLoad. -->
            <invoke function="BioSPI_BSPLoad">
                  <input name="BSPUuid" var="bspUuid"/>
                  <input name="BioAPINotifyCallback" value="*"/>
                  <input name="BFPEnumerationHandler" value="*"/>
                  <input name="MemoryFreeHandler" value="*" />
                  <return setvar="return"/>
            </invoke>
            <!-- Issue a conformity response.
                        If the condition specified in the <description> below is false, a FAIL
                              conformity response is issued, otherwise a PASS conformity response is
                              issued.-->
            <assert_condition>
                  <description>
                        The function BioSPI_BSPLoad has returned BioAPIERR_INVALID_UUID
                  </description>
                  <equal_to var1="return"
                              var2="__BioAPIERR_BSP_INVALID_UUID"/>
            </assert_condition>
      </activity>

</package>
```

8.4 Assertion 1b - *BioSPI_BSPLoad_ValidParam*

Description: This assertion checks if calling BioSPI_BSPLoad with valid input parameters returns BioAPI_OK.

Excerpts

Subclause 9.3.1.1

*BioAPI_RETURN BioAPI BioSPI_BSPLoad(const BioAPI_UUID *BSPUuid,*

BioSPI_EventHandler BioAPINotifyCallback,

BioSPI_BFP_ENUMERATION_HANDLER BFPEnumerationHandler,

BioSPI_MEMORY_FREE_HANDLER MemoryFreeHandler);

This function completes the module initialization process between BioAPI and the biometric service module.

Return Value

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

This function completes the component initialization process between 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.

Subclause A.4

This function must be supported by all types of BSP.

References: 9.3.1.1 and A.4

Scenario:

1) Call BioSPI_BSPLoad with valid input parameters.

2) Check the return value. If it is BioAPI_OK, then issue a PASS conformity response, otherwise issue a FAIL conformity response.

If any of the intermediate operations fails, an UNDECIDED conformity response is issued.

Expected results: The call to BioSPI_BSPLoad returns BioAPI_OK.

Assertion language package

```
<package name="01f6c6f0-0c19-1085-97fe-0002a5d5fd2e">
      <author>
            ISO/IEC JTC1 SC37
      </author>

      <description>
            This package contains the assertion "BioSPI_BSPLoad_ValidParam" (see the "description"
                              element of the assertion below).
      </description>

      <assertion name="BioSPI_BSPLoad_ValidParam" model="BSPTesting">
            <description>
                  This assertion checks if calling BioSPI_BSPLoad with valid input parameters returns
                              BioAPI_OK.
                  The relevant text in BioAPI 2.0 is quoted below from subclauses 9.3.1.1 and A.4.
                  ________________________________________________________
                  BioAPI_RETURN BioAPI BioSPI_BSPLoad(const BioAPI_UUID *BSPUuid,
                        BioSPI_EventHandler BioAPINotifyCallback,
                        BioSPI_BFP_ENUMERATION_HANDLER  BFPEnumerationHandler,
                        BioSPI_MEMORY_FREE_HANDLER  MemoryFreeHandler);

                  This function completes the module initialization process between BioAPI and the
                              biometric service module.

                  Return Value
                  A BioAPI_RETURN value indicating success or specifying a particular error
                              condition. The value BioAPI_OK indicates success. All other values
                              represent an error condition.
                  ________________________________________________________
                  Subclause 9.3.1.1:
                  This function completes the component initialization process between 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.
                  Subclause A.4:
                  This function must be supported by all types of BSP.
                  ________________________________________________________

                  In order to determine conformance with respect to the text above, the following
                              steps are performed:

                        1)    Call BioSPI_BSPLoad with valid input parameters.
                        2)    Check the return value.  If it is BioAPI_OK, then issue a PASS
                              conformity response, otherwise issue a FAIL conformity response.

                  If any of the intermediate operations fails, an UNDECIDED conformity response is
                              issued.
            </description>

            <!-- UUID of the BSP under test -->
            <input name="_BspUuid"/>

            <!-- Indicates whether a framework callback address for BSP event notifications will be
                              provided (value "*") or not (value "") -->
            <input name="_BioAPINotifyCallback" />

            <!-- address of the BFP enumeration handler -->
            <input name="_BFPEnumerationHandler" />

            <!-- address of the memory deallocation handler -->
            <input name="_MemoryFreeHandler" />
            <!-- Invocation of the primary activity of this assertion with input parameter values
                              assigned from the assertion's parameters. -->
            <invoke activity="BioSPI_BspLoad">
                  <input name="BspUuid" var="_BspUuid"/>
                  <input name="BioAPIdNotifyCallback"
                              var="_BioAPINotifyCallback" />
                  <input name="BFPEnumerationHandler"
                              var="_BFPEnumerationHandler" />
                  <input name="MemoryFreeHandler"
                              var="_MemoryFreeHandler" />
            </invoke>
      </assertion>

      <activity name="BioSPI_BspLoad">
            <input name="BspUuid"/>
            <input name="BioAPIdNotifyCallback"/>
            <input name="BFPEnumerationHandler" />
            <input name="MemoryFreeHandler" />

            <!-- Invoke the function BioSPI_BspLoad.  -->
            <invoke function="BioSPI_BSPLoad">
                  <input name="BSPUuid" var="BspUuid"/>
                  <input name="BioAPINotifyCallback"
                              var="BioAPIdNotifyCallback"/>
                  <input name="BFPEnumerationHandler"
                              var="BFPEnumerationHandler"/>
                  <input name="MemoryFreeHandler"
                          var="MemoryFreeHandler" />
                  <return setvar="return"/>
            </invoke>

            <!-- Issue a conformity response.
                        If the condition specified in the <description> below is false, a FAIL
                              conformity response is issued, otherwise a PASS conformity response is
                              issued.-->
            <assert_condition>
                  <description>
                        The function BioSPI_BSPLoad has returned BioAPI_OK.
                  </description>
                  <equal_to var1="return" var2="__BioAPI_OK"/>
            </assert_condition>

            <!-- Invoke the function BioSPI_BSPUnload. -->
            <invoke function="BioSPI_BSPUnload">
                  <input name="BSPUuid" var="BspUuid"/>
                  <return setvar="return"/>       </invoke>

            <!-- Issue a conformity response.
                        If the condition specified in the <description> below is false, an UNDECIDED
                              conformity response is issued, otherwise a PASS conformity response is
                              issued.-->
            <assert_condition response_if_false="undecided">
                  <description>
                        The function BioSPI_BSPUnload has returned BioAPI_OK.
                  </description>
                  <equal_to var1="return" var2="__BioAPI_OK"/>
            </assert_condition>
      </activity>

</package>
```

8.5 Assertion 2a - *BioSPI_BSPUnload_ValidParam*

Description: This assertion checks if calling BioSPI_BSPUnload with valid input parameters returns BioAPI_OK.

Excerpts

Subclause 9.3.1.2

BioAPI_RETURN BioAPI BioSPI_BSPUnload

*(const BioAPI_UUID *BSPUuid);*

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

Return Value

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

Subclause A.4

This function must be supported by all types of BSP.

References: 9.3.1.2 and A.4

Scenario:

1) Call BioSPI_BSPLoad with valid input parameters. The call is expected to succeed.

2) Call BioSPI_BSPUnload with valid input parameters.

3) Check the return value, which is expected to be BioAPI_OK.

If any of the intermediate operations fails, an UNDECIDED conformity response is issued.

Expected results: The call to BioSPI_BSPUnload returns BioAPI_OK.

Assertion language package

```
<package name="01661010-0c22-1085-8688-0002a5d5fd2e">
      <author>
            ISO/IEC JTC1 SC37
      </author>

      <description>
            This package contains the assertion "BioSPI_BSPUnload_ValidParam" (see the "description"
                        element of the assertion below).
      </description>

      <assertion name="BioSPI_BSPUnload_ValidParam" model="BSPTesting">
            <description>
                  This assertion checks if calling BioSPI_BSPUnload with valid input parameters
                        returns BioAPI_OK.
                  The relevant text in BioAPI 2.0 is quoted below from subclauses 9.3.1.2 and A.4.
                  _______________________________________________________
                  BioAPI_RETURN BioAPI BioSPI_BSPUnload
                        (const BioAPI_UUID *BSPUuid);

                  This function disables events and de-registers the BioAPI event-notification
                        function. The biometric service module may perform cleanup operations,
                        reversing the initialization performed in BioSPI_BSPLoad.
            Return Value
            A BioAPI_RETURN value indicating success or specifying a particular error
                        condition. The value BioAPI_OK indicates success. All other values
                        represent an error condition.
            _______________________________________________________
            Subclause 9.3.1.2:
            This function disables events and de-registers the BioAPI event-notification
                        function.
            Subclause A.4:
            This function must be supported by all types of BSP.
            _______________________________________________________
            In order to determine conformance with respect to the text above, the following
                        steps are performed:

                  1)    Call BioSPI_BSPLoad with valid input parameters.  The call is expected
                        to succeed.
                  2)    Call BioSPI_BSPUnload with valid input parameters.
                  3)    Check the return value, which is expected to be BioAPI_OK.

            If any of the intermediate operations fails, an UNDECIDED conformity response is
                        issued.
      </description>

      <!-- UUID of the BSP under test -->
      <input name="_bspUuid"/>

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

   <activity name="BioSPI_BSPUnload">
      <input name="bspUuid"/>

      <!-- Invoke the function BioSPI_BSPLoad.  -->
      <invoke function="BioSPI_BSPLoad">
            <input name="BSPUuid" var="bspUuid"/>
            <input name="BioAPINotifyCallback" value="*"/>
            <input name="BFPEnumerationHandler" value="*"/>
            <input name="MemoryFreeHandler" value="*" />
            <return setvar="return"/>
      </invoke>

      <!-- Issue a conformity response.
                  If the condition specified in the <description> below is false, an UNDECIDED
                        conformity response is issued and the execution of the activity is
                        interrupted, otherwise a PASS conformity response is issued.-->
      <assert_condition response_if_false="undecided"
                  break_if_false="true">
            <description>
                  The function BioSPI_BSPLoad has returned BioAPI_OK
            </description>
            <equal_to var1="return" var2="__BioAPI_OK"/>
      </assert_condition>

      <!-- Invoke the function BioSPI_BSPUnload. -->
      <invoke function="BioSPI_BSPUnload">
            <input name="BSPUuid" var="bspUuid"/>
            <return setvar="return"/>
      </invoke>

      <!-- Issue a conformity response.
                  If the condition specified in the <description> below is false, a FAIL
                        conformity response is issued, otherwise a PASS conformity response is
                        issued.-->
      <assert_condition>
            <description>
                  The function BioSPI_BSPUnload has returned BioAPI_OK
            </description>
            <equal_to var1="return" var2="__BioAPI_OK"/>
      </assert_condition>
   </activity>
</package>
```

8.6 Assertion 2b - BioSPI_BSPUnload_InvalidUUID

Description: This assertion checks if calling BioSPI_BSPUnload with an invalid input parameter UUID returns BioAPIERR_INVALID_UUID.

Excerpts

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 initialization performed in BioSPI_BSPLoad.

Return Value

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

Parameters: BSPUuid (input) - The UUID of the invoked biometric service provider.

Subclause 11.2.3

Errors: BioAPIERR_INVALID_UUID

References: 9.3.1.2 and 11.2.3.

Scenario:

1) Call BioSPI_BSPLoad with valid input parameters. This call is expected to succeed.

2) Call BioSPI_BSPUnload with an invalid UUID.

3) Check the return value. If it is not BioAPIERR_INVALID_UUID, issue a FAIL conformity response.

If any of the intermediate operations fails, an UNDECIDED conformity response is issued.

Expected results: The call to BioSPI_BSPUnload returns BioAPIERR_INVALID_UUID.

Assertion language package

```
<package name="01c2e5b0-0c3b-1085-b31d-0002a5d5fd2e">
      <author>
            ISO/IEC JTC1 SC37
      </author>

      <description>
            This package contains the assertion "BioSPI_BSPUnload_InvalidUUID" (see the "description"
                        element of the assertion below).
      </description>

      <assertion name="BioSPI_BSPUnload_InvalidUUID" model="BSPTesting">
            <description>
                  This assertion checks if calling BioSPI_BSPUnload with an invalid input parameter
                        UUID returns BioAPIERR_INVALID_UUID.
                  The relevant text in BioAPI 2.0 is quoted below from subclauses 9.3.1.2 and 11.2.3
```

```
          ________________________________________________________
          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
                      initialization performed in BioSPI_BSPLoad.

          Return Value
          A BioAPI_RETURN value indicating success or specifying a particular error
                      condition. The value BioAPI_OK indicates success. All other values
                      represent an error condition.
          ________________________________________________________
          Subclause 9.3.1.2:
          Parameters: BSPUuid (input) - The UUID of the invoked biometric service provider.
          Subclause 11.2.3:
          Errors: BioAPIERR_INVALID_UUID
          ________________________________________________________

          In order to determine conformance with respect to the text above, the following
                      steps are performed:

                1)    Call BioSPI_BSPLoad with valid input parameters.  This call is expected
                      to succeed.
                2)    Call BioSPI_BSPUnload with an invalid UUID.
                3)    Check the return value.  If it is not BioAPIERR_INVALID_UUID, issue a
                      FAIL conformity response.

          If any of the intermediate operations fails, an UNDECIDED conformity response is
                      issued.
    </description>

    <!-- UUID of the BSP under test -->
    <input name="_bspUuid"/>

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

<activity name="BioSPI_BSPUnload">
    <input name="bspUuid"/>

    <!-- Invoke the function BioSPI_BSPLoad. -->
    <invoke function="BioSPI_BSPLoad">
          <input name="BSPUuid" var="bspUuid"/>
          <input name="BioAPINotifyCallback" value="*"/>
          <input name="BFPEnumerationHandler" value="*"/>
          <input name="MemoryFreeHandler" value="*" />
          <return setvar="return"/>
    </invoke>

    <!-- Issue a conformity response.
                If the condition specified in the <description> below is false, an UNDECIDED
                      conformity response is issued and the execution of the activity is
                      interrupted, otherwise a PASS conformity response is issued.-->
    <assert_condition response_if_false="undecided"
                break_if_false="true">
          <description>
                The function BioSPI_BSPLoad has returned BioAPI_OK
          </description>
          <equal_to var1="return" var2="__BioAPI_OK"/>
    </assert_condition>

    <!-- Invoke the function BioSPI_BSPUnload.-->
    <invoke function="BioSPI_BSPUnload">
          <input name="BSPUuid"
                      value="00000000-0000-0000-0000-000000000000"/>
          <return setvar="return"/>
    </invoke>

    <!-- Issue a conformity response.
```

```
                    If the condition specified in the <description> below is false, a FAIL
                          conformity response is issued, otherwise a PASS conformity response is
                          issued.-->
        <assert_condition>
              <description>
                    The function BioSPI_BSPUnload has returned BioAPIERR_INVALID_UUID
              </description>
              <equal_to var1="return"
                          var2="__BioAPIERR_BSP_INVALID_UUID"/>
        </assert_condition>

        <!-- Invoke the function BioSPI_BSPUnload to unload the BSP-->
        <invoke function="BioSPI_BSPUnload">
              <input name="BSPUuid" var="bspUuid"/>
              <return setvar="return"/>
        </invoke>

        <!-- Issue a conformity response.
                    If the condition specified in the <description> below is false, a FAIL
                          conformity response is issued, otherwise a PASS conformity response is
                          issued.-->
        <assert_condition>
              <description>
                    The function BioSPI_BSPUnload has returned BioAPI_OK.
              </description>
              <equal_to var1="return"
                          var2="__BioAPI_OK"/>
        </assert_condition>

  </activity>

</package>
```

8.7 Assertion 2c - *BioSPI_BSPUnload_UnmatchedLoad*

Description: This assertion checks if calling BioSPI_BSPUnload without a matching call to BioSPI_BSPLoad returns BioAPIERR_BSP_NOT_LOADED.

Excerpts

Subclause 9.3.1.2

BioAPI_RETURN BioAPI BioSPI_BSPUnload

*(const BioAPI_UUID *BSPUuid);*

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

Subclause 8.1.6.1

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.

References: 9.3.1.2 and 8.1.6.1

Scenario:

1) Call BioSPI_BSPLoad with valid input parameters.

2) Call BioSPI_BSPUnload with valid input parameters.

3) Call BioSPI_BSPUnload with valid input parameters again.

4) Check the return value. If the value is BioAPIERR_BSP_NOT_LOADED, then the test passed, otherwise the test failed.

If any of the intermediate operations fails, an UNDECIDED conformity response is issued.

Expected results: The second call to BioSPI_BSPUnload returns BioAPIERR_BSP_NOT_LOADED.

Assertion language package

```
<package name="02f6c618-0c23-1085-ba89-0002a5d5fd2e">
      <author>
            ISO/IEC JTC1 SC37
      </author>

      <description>
            This package contains the assertion "BioSPI_BSPUnload_UnmatchedLoad" (see the
                              "description" element of the assertion below).
      </description>

      <assertion name="BioSPI_BSPUnload_UnmatchedLoad" model="BSPTesting">
            <description>
                  This assertion checks if calling BioSPI_BSPUnload without a matching call to
                              BioSPI_BSPLoad returns BioAPIERR_BSP_NOT_LOADED.
                  The relevant text in BioAPI 2.0 is quoted below from subclauses 9.3.1.2 and 8.1.6.1
                  _______________________________________________________
                  BioAPI_RETURN BioAPI BioSPI_BSPUnload
                        (const BioAPI_UUID *BSPUuid);

                  This function disables events and de-registers the BioAPI event-notification
                              function. The biometric service module may perform cleanup operations,
                              reversing the initialization performed in BioSPI_BSPLoad.
                  _______________________________________________________
                  Subclause 8.1.6.1
                  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.
                  _______________________________________________________
                  In order to determine conformance with respect to the text above, the following
                              steps are performed:

                        1)    Call BioSPI_BSPLoad with valid input parameters.
                        2)    Call BioSPI_BSPUnload with valid input parameters.
                        3)    Call BioSPI_BSPUnload with valid input parameters again.
                        4)    Check the return value.  If the value is BioAPIERR_BSP_NOT_LOADED, then
                              the test passed, otherwise the test failed.

                  If any of the intermediate operations fails, an UNDECIDED conformity response is
                              issued.
            </description>

            <!-- UUID of the BSP under test -->
            <input name="_bspUuid"/>

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

      <activity name="BioSPI_BSPUnload_Unmatch">
            <input name="bspUuid"/>

            <!-- Invoke the function BioSPI_BSPLoad. -->
            <invoke function="BioSPI_BSPLoad">
                  <input name="BSPUuid" var="bspUuid"/>
                  <input name="BioAPINotifyCallback" value="*"/>
                  <input name="BFPEnumerationHandler" value="*"/>
                  <input name="MemoryFreeHandler" value="*" />
                  <return setvar="return"/>
            </invoke>

            <!-- Issue a conformity response.
                        If the condition specified in the <description> below is false, an UNDECIDED
                              conformity response is issued and the execution of the activity is
                              interrupted, otherwise a PASS conformity response is issued.-->
            <assert_condition response_if_false="undecided"
```

```
                    break_if_false="true">
              <description>
                    The function BioSPI_BSPLoad has returned BioAPI_OK
              </description>
              <equal_to var1="return" var2="__BioAPI_OK"/>
        </assert_condition>

        <!-- Invoke the function BioSPI_BSPUnload. -->
        <invoke function="BioSPI_BSPUnload">
              <input name="BSPUuid" var="bspUuid"/>
              <return setvar="return"/>
        </invoke>

        <!-- Issue a conformity response.
                    If the condition specified in the <description> below is false, an UNDECIDED
                          conformity response is issued and the execution of the activity is
                          interrupted, otherwise a PASS conformity response is issued.-->
        <assert_condition response_if_false="undecided"
                    break_if_false="true">
              <description>
                    The function BioSPI_BSPUnload has returned BioAPI_OK
              </description>
              <equal_to var1="return" var2="__BioAPI_OK"/>
        </assert_condition>

        <!-- Invoke the function BioSPI_BSPUnload again. -->
        <invoke function="BioSPI_BSPUnload">
              <input name="BSPUuid" var="bspUuid"/>
              <return setvar="return"/>
        </invoke>

        <!-- Issue a conformity response.
                    If the condition specified in the <description> below is false, a FAIL
                          conformity response is issued, otherwise a PASS conformity response is
                          issued, otherwise a PASS conformity response is issued.-->
        <assert_condition>
              <description>
                    The function BioSPI_BSPUnload has returned BioAPIERR_BSP_NOT_LOADED.
              </description>
              <equal_to var1="return"
                          var2="__BioAPIERR_BSP_BSP_NOT_LOADED"/>
        </assert_condition>
  </activity>

</package>
```

8.8 Assertion 2d - *BioSPI_BSPUnload_Confirm*

Description: This assertion checks if calling BioSPI_BSPUnload truly unloads the BSP.

Excerpts

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 initialization performed in BioSPI_BSPLoad.

Return Value

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

References: 9.3.1.2

Scenario:

1) Call BioSPI_BSPLoad with valid input parameters. The call is expected to succeed.

2) Call BioSPI_BSPUnload with valid input parameters.

3) Call BioSPI_BSPAttach with valid input parameters. It is expected to return BioAPIERR_BSP_NOT_LOADED.

If any of the intermediate operations fails, an UNDECIDED conformity response is issued.

Expected results: The call to BioSPI_BSPAttach returns BioAPIERR_BSP_NOT_LOADED.

Assertion language package

```
<package name="03daf040-0c3b-1085-a9fd-0002a5d5fd2e">
      <author>
            ISO/IEC JTC1 SC37
      </author>

      <description>
            This package contains the assertion "BioSPI_BSPUnload_Confirm" (see the "description"
                              element of the assertion below).
      </description>

      <assertion name="BioSPI_BSPUnload_Confirm" model="BSPTesting">
            <description>
                  This assertion checks if calling BioSPI_BSPUnload truly unloads the BSP.
                  The relevant text in BioAPI 2.0 is quoted below from 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
                              initialization performed in BioSPI_BSPLoad.

                  Return Value
                  A BioAPI_RETURN value indicating success or specifying a particular error
                              condition. The value BioAPI_OK indicates success. All other values
                              represent an error condition.
                  _________________________________________________________

                  In order to determine conformance with respect to the text above, the following
                              steps are performed:

                        1)    Call BioSPI_BSPLoad with valid input parameters.  The call is expected
                              to succeed.
                        2)    Call BioSPI_BSPUnload with valid input parameters.
                        3)    Call BioSPI_BSPAttach with valid input parameters.  It is expected to
                              return BioAPIERR_BSP_NOT_LOADED.

                  If any of the intermediate operations fails, an UNDECIDED conformity response is
                              issued.
            </description>

            <!-- UUID of the BSP under test -->
            <input name="_bspUuid"/>

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

      <activity name="BioSPI_BSPUnload">
            <input name="bspUuid"/>

            <!-- Invoke the function BioSPI_BSPLoad. -->
            <invoke function="BioSPI_BSPLoad">
                  <input name="BSPUuid" var="bspUuid"/>
            <input name="BioAPINotifyCallback" value="*"/>
            <input name="BFPEnumerationHandler" value="*"/>
            <input name="MemoryFreeHandler" value="*" />
            <return setvar="return"/>
      </invoke>

      <!-- Issue a conformity response.
                  If the condition specified in the <description> below is false, an UNDECIDED
                        conformity response is issued and the execution of the activity is
                        interrupted, otherwise a PASS conformity response is issued.-->
      <assert_condition response_if_false="undecided"
                  break_if_false="true">
            <description>
                  The function BioSPI_BSPLoad has returned BioAPI_OK
            </description>
            <equal_to var1="return" var2="__BioAPI_OK"/>
      </assert_condition>

      <!-- Invoke the function BioSPI_BSPUnload.-->
            <invoke function="BioSPI_BSPUnload">
            <input name="BSPUuid" var="bspUuid"/>
            <return setvar="return"/>
      </invoke>

      <!-- Issue a conformity response.
                  If the condition specified in the <description> below is false, an UNDECIDED
                        conformity response is issued and the execution of the activity is
                        interrupted, otherwise a PASS conformity response is issued.-->
      <assert_condition response_if_false="undecided"
                  break_if_false="true">
            <description>
                  The function BioSPI_BSPUnload has returned BioAPI_OK
            </description>
            <equal_to var1="return" var2="__BioAPI_OK"/>
      </assert_condition>

      <!-- Invoke the function BioSPI_BSPAttach to check if the BSP is no longer loaded -->
      <invoke function="BioSPI_BSPAttach">
            <input name="BSPUuid" var="bspUuid"/>
            <input name="Version" value="32"/>
            <input name="NumUnits" value="0" />
            <input name="BSPHandle" value="1"/>
            <return setvar="return"/>
      </invoke>

      <!-- Issue a conformity response.
                  If the condition specified in the <description> below is false, a FAIL
                        conformity response is issued, otherwise a PASS conformity response is
                        issued.-->
      <assert_condition>
            <description>
                  The function BioSPI_BSPAttach has returned BioAPIERR_BSP_NOT_LOADED
            </description>
            <equal_to var1="return"
                        var2="__BioAPIERR_BSP_BSP_NOT_LOADED"/>
      </assert_condition>
   </activity>

</package>
```

8.9 Assertion 3a - *BioSPI_BSPAttach_ValidParam*

Description: This assertion checks if a call to the function BioSPI_BSPAttach with valid input parameters returns BioAPI_OK.

Excerpts

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 BSP should perform all initializations required to support the new BSP invocation.

Subclause A.4

This function should be supported all types of BSPs.

References: 9.3.1.3 and A.4

Scenario:

1) Load the BSP under test

2) Attach the BSP under test

3) Check the return value of the function call.

4) Detach and unload the BSP.

If any of the intermediate operations fails, an UNDECIDED conformity response is issued.

Expected results: The call to BioSPI_BSPAttach returns BioAPI_OK

Assertion language package

```
<package name="00ae6488-0c3d-1085-9912-0002a5d5fd2e">
      <author>
            ISO/IEC JTC1 SC37
      </author>

      <description>
            This package contains the assertion "BioSPI_BSPAttach_ValidParam" (see the "description"
                              element of the assertion below).
      </description>

      <assertion name="BioSPI_BSPAttach_ValidParam" model="BSPTesting">
            <description>
                  This assertion checks if a call to the function BioSPI_BSPAttach with valid input
                              parameters returns BioAPI_OK.
                  The relevant text in BioAPI 2.0 is quoted below from subclauses 9.3.1.3 and A.4.
```

```
          ___________________________________________________
          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 BSP should perform all initializations required to support the new BSP
                      invocation.
          ____________________________________________________
          Subclause A.4:
          This function should be supported all types of BSPs.
          ____________________________________________________

          In order to determine conformance with respect to the text above, the following
                      steps are performed:

                1)    Load the BSP under test
                2)    Attach the BSP under test
                3)    Check the return value of the function call.
                4)    Detach and unload the BSP.

          If any of the intermediate operations fails, an UNDECIDED conformity response is
                      issued.
      </description>

      <!-- UUID of the BSP under test -->
      <input name="_bspUuid"/>

      <!-- Major version number of the BSP under test -->
      <input name="_bspVersion"/>


      <!-- Timeout for the BioAPI_NOTIFY_INSERT event -->
      <input name="_inserttimeout"/>

      <!-- Module handle for attached BSP -->
      <input  name="_bsphandle"/>

      <!-- BioAPI Unit category -->

      <!-- Invocation of the primary activity of this assertion with input parameter values
                      assigned from the assertion's parameters. -->
      <invoke activity="LoadAndAttach" >
            <input name="bspUuid" var="_bspUuid"/>
            <input name="bspVersion" var="_bspVersion"/>
            <input name="unitIDOrNull" value="0"/>
            <input name="bspHandle" var="_bsphandle"/>
            <input name="eventtimeouttime" var="_inserttimeout"/>
      </invoke>

      <!-- Activity bound to a function of the framework callback interface exposed by the
                      testing component.  This activity will be automatically invoked on each
                      incoming call to the function to which it is bound. -->
      <bind activity="EventHandler"
                package="02c59458-0c46-1085-95d7-0002a5d5fd2e"
                function="BioSPI_EventHandler"/>
</assertion>

<activity name="LoadAndAttach">
      <input name="bspUuid"/>
      <input name="bspVersion"/>
      <input name="unitIDOrNull"/>
      <input name="bspHandle"/>
      <input name="eventtimeouttime"/>

      <!-- Initialize the global variable "_unitIDOrNull" to make it available to the activity
                      "EventHandler" -->
      <set name="_unitIDOrNull" var="unitIDOrNull"/>

      <!-- Initialize the global variable "_insert" to "false".  The activity "EventHandler"
                      will set this variable to "true" when a BioAPI_NOTIFY_INSERT event
```

```
                        notification is received and will set it to "false" when a
                        BioAPI_NOTIFY_REMOVE event notification is received -->
            <set name="_insert" value="false"/>

            <set name="eventtimeoutflag" value="false"/>

            <!-- Invoke the function BioSPI_BSPLoad.-->
            <invoke function="BioSPI_BSPLoad">
                  <input name="BSPUuid" var="bspUuid"/>
                  <input name="BioAPINotifyCallback" value="*"/>
                  <input name="BFPEnumerationHandler" value="*"/>
                  <input name="MemoryFreeHandler" value="*" />
                  <return setvar="return"/>
            </invoke>

            <!-- Issue a conformity response.
                  If the condition specified in the <description> below is false, an UNDECIDED
                        conformity response is issued and the execution of the activity is
                        interrupted, otherwise a PASS conformity response is issued.-->
            <assert_condition response_if_false="undecided"
                  break_if_false="true">
                  <description>
                        The function BioSPI_BSPLoad has returned BioAPI_OK.
                  </description>
                  <equal_to var1="return" var2="__BioAPI_OK"/>
            </assert_condition>

            <!-- Wait until the BioAPI_NOTIFY_INSERT event notification has been received, but no
                        longer than the specified maximum duration.-->
            <wait_until timeout_var="eventtimeouttime"
                  setvar="eventtimeoutflag" var="_insert"/>

            <!-- Issue a conformity response.
                  If the condition specified in the <description> below is false, an UNDECIDED
                        conformity response is issued and the execution of the activity is
                        interrupted, otherwise a PASS conformity response is issued.-->
            <assert_condition response_if_false="undecided"
                  break_if_false="true">
                  <description>
                        The BioAPI_NOTIFY_INSERT event notification has been received within the
                        specified maximum duration
                  </description>
                  <not var="eventtimeoutflag"/>
            </assert_condition>

            <!-- Invoke the function BioSPI_BSPAttach. -->
            <invoke function="BioSPI_BSPAttach">
                  <input name="BSPUuid" var="bspUuid"/>
                  <input name="Version" var="bspVersion"/>
                  <input name="Unit_1_UnitCategory" var="_unitCategory" />
                  <input name="Unit_1_UnitID" var="_unitID"/>
                  <input name="NumUnits" value="1" />
                  <input name="BSPHandle" var="bspHandle"/>
                  <return setvar="return"/>
            </invoke>

            <!-- Issue a conformity response.
                  If the condition specified in the <description> below is false, a FAIL
                        conformity response is issued and the execution of the activity is
                        interrupted, otherwise a PASS conformity response is issued.-->
            <assert_condition break_if_false="true">
                  <description>
                        The function BioSPI_BSPAttach has returned BioAPI_OK
                  </description>
                  <equal_to var1="return" var2="__BioAPI_OK"/>
            </assert_condition>

            <!-- Invoke the functions BioSPI_BSPDetach and BioSPI_BSPUnload -->
            <invoke activity="DetachAndUnload"
                  package="02c59458-0c46-1085-95d7-0002a5d5fd2e" >
                  <input name="bspUuid" var="bspUuid"/>
                  <input name="BSPHandle" var="bspHandle"/>
            </invoke>
      </activity>

</package>
```

8.10 Assertion 3b - *BioSPI_BSPAttach_InvalidUUID*

Description: This assertion checks if a call to the function BioSPI_BSPAttach returns BioAPIERR_INVALID_UUID when the input UUID is invalid.

Excerpts

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.

Parameters: BSPUuid (input) - The BioAPI_UUID of the invoked service provider module.

Subclause 11.2.3

Errors: BioAPIERR_INVALID_UUID

References: 9.3.1.3 and 11.2.3

Scenario:

1) Load the BSP under test.

2) Attach the BSP under test passing an invalid UUID.

3) Check the return value of the function call.

If any of the intermediate operations fails, an UNDECIDED conformity response is issued.

Expected results: The call to BioSPI_BSPAttach returns BioAPIERR_INVALID_UUID.

Assertion language package

```
<package name="049cc170-0c5f-1085-981f-0002a5d5fd2e">
      <author>
            ISO/IEC JTC1 SC37
      </author>

      <description>
            This package contains the assertion "BioSPI_BSPAttach_InvalidUUID" (see the "description"
                              element of the assertion below).
      </description>

      <assertion name="BioSPI_BSPAttach_InvalidUUID" model="BSPTesting">
            <description>
                  This assertion checks if a call to the function BioSPI_BSPAttach returns
                              BioAPIERR_INVALID_UUID when the input UUID is invalid.
                  The relevant text in BioAPI 2.0 is quoted below from subclauses 9.3.1.3 and 11.2.3.
            ______________________________________________________
            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.
            ________________________________________________________
            Subclause 9.3.1.3:
            Parameters: BSPUuid (input) - The BioAPI_UUID of the invoked service provider
                        module.
            Subclause 11.2.3:
            Errors: BioAPIERR_INVALID_UUID
            ________________________________________________________

            In order to determine conformance with respect to the text above, the following
                        steps are performed:

                  1)    Load the BSP under test.
                  2)    Attach the BSP under test passing an invalid UUID.
                  3)    Check the return value of the function call.

            If any of the intermediate operations fails, an UNDECIDED conformity response is
                        issued.
      </description>

      <!-- UUID of the BSP under test -->
      <input name="_bspUuid"/>

      <!-- version number of the BSP under test -->
      <input name="_bspVersion"/>

      <!-- Timeout for the BioAPI_NOTIFY_INSERT event -->
      <input name="_inserttimeout"/>

      <!-- Module handle for attached BSP -->
      <input  name="_bsphandle"/>

      <!-- Invocation of the primary activity of this assertion with input parameter values
                        assigned from the assertion's parameters. -->
      <invoke activity="LoadAndAttach">
            <input name="bspUuid" var="_bspUuid"/>
            <input name="bspVersion" var="_bspVersion"/>
            <input name="unitIDOrNull" value="0"/>
            <input name="bspHandle" var="_bsphandle"/>
            <input name="eventtimeouttime" var="_inserttimeout"/>
      </invoke>

      <!-- Activity bound to a function of the framework callback interface exposed by the
                        testing component.  This activity will be automatically invoked on each
                        incoming call to the function to which it is bound. -->
      <bind activity="EventHandler"
                  package="02c59458-0c46-1085-95d7-0002a5d5fd2e"
                  function="BioSPI_EventHandler"/>
</assertion>

<activity name="LoadAndAttach">
      <input name="bspUuid"/>
      <input name="bspVersion"/>
      <input name="unitIDOrNull"/>
      <input name="bspHandle"/>
      <input name="eventtimeouttime"/>

      <!-- Initialize the global variable "_unitIDOrNull" to make it available to the activity
                        "EventHandler" -->
      <set name="_unitIDOrNull" var="unitIDOrNull"/>

      <!-- Initialize the global variable "_insert" to "false".  The activity "EventHandler"
                        will set this variable to "true" when a BioAPI_NOTIFY_INSERT event
                        notification is received, and will set it to "false" when a
                        BioAPI_NOTIFY_REMOVE event notification is received. -->
      <set name="_insert" value="false"/>
```

```
<set name="eventtimeoutflag" value="false"/>

<!-- Invoke the function BioSPI_BSPLoad. -->
<invoke function="BioSPI_BSPLoad">
      <input name="BSPUuid" var="bspUuid"/>
      <input name="BioAPINotifyCallback" value="*"/>
      <input name="BFPEnumerationHandler" value="*"/>
      <input name="MemoryFreeHandler" value="*" />
      <return setvar="return"/>
</invoke>

<!-- Issue a conformity response.
            If the condition specified in the <description> below is false, an UNDECIDED
                  conformity response is issued and the execution of the activity is
                  interrupted, otherwise a PASS conformity response is issued.-->
<assert_condition response_if_false="undecided"
            break_if_false="true">
      <description>
            The function BioSPI_BSPLoad has returned BioAPI_OK
      </description>
      <equal_to var1="return" var2="__BioAPI_OK"/>
</assert_condition>

<!-- Wait until the BioAPI_NOTIFY_INSERT event notification has been received, but no
                  longer than the specified maximum duration.-->
<wait_until timeout_var="eventtimeouttime"
            setvar="eventtimeoutflag" var="_insert"/>

<!-- Issue a conformity response.
            If the condition specified in the <description> below is false, an UNDECIDED
                  conformity response is issued and the execution of the activity is
                  interrupted, otherwise a PASS conformity response is issued.-->
<assert_condition response_if_false="undecided"
            break_if_false="true">
      <description>
            The BioAPI_NOTIFY_INSERT event notification has been received within the
                  specified maximum duration
      </description>
      <not var="eventtimeoutflag"/>
</assert_condition>

<!-- Invoke the function BioSPI_BSPAttach passing an invalid UUID. -->
<invoke function="BioSPI_BSPAttach">
      <input name="BSPUuid"
                  value="00000000-0000-0000-0000-000000000000"/>
      <input name="Version" var="bspVersion"/>
      <input name="Unit_1_UnitCategory" var="_unitCategory" />
      <input name="Unit_1_UnitID" var="_unitID"/>
      <input name="NumUnits" value="1" />
      <input name="BSPHandle" var="bspHandle"/>
      <return setvar="return"/>
</invoke>

<!-- Issue a conformity response.
            If the condition specified in the <description> below is false, a FAIL
                  conformity response is issued, otherwise a PASS conformity response is
                  issued.-->
<assert_condition>
      <description>
            The function BioSPI_BSPAttach has returned BioAPIERR_INVALID_UUID.
      </description>
      <equal_to var1="return"
                  var2="__BioAPIERR_BSP_INVALID_UUID"/>
</assert_condition>

<!-- Invoke the function BioSPI_BSPUnload -->
<invoke function="BioSPI_BSPUnload" >
      <input name="BSPUuid" var="bspUuid" />
      <return setvar="return"/>
</invoke>

<!-- Issue a conformity response.
            If the condition specified in the <description> below is false, an UNDECIDED
                  conformity response is issued, otherwise a PASS conformity response is
                  issued.-->
<assert_condition response_if_false="undecided"
```

```
                    break_if_false="true">
                <description>
                    The function BioSPI_BSPUnload has returned BioAPI_OK
                </description>
                <equal_to var1="return" var2="__BioAPI_OK"/>
            </assert_condition>
        </activity>

</package>
```

8.11 Assertion 3c - *BioSPI_BSPAttach_InvalidVersion*

Description: This assertion checks if a call to the function BioSPI_BSPAttach returns BioAPIERR_INCOMPATIBLE_VERSION when the caller specifies an incompatible version.

Excerpts

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);

The biometric service provider shall verify compatibility with the version level specified by Version. If the version is not compatible, then this function fails.

Parameters: Version (input) - the major and minor version number of the BioAPI specification that the application is expecting the BSP to support. The BSP shall determine whether it is compatible with the required version.

Subclause 11.2.3

Errors: BioAPIERR_INCOMPATIBLE_VERSION

References: 9.3.1.3 and 11.2.3

Scenario:

1) Load the BSP under test.

2) Attach the BSP under test using an invalid version number.

3) Check the return value of the function call.

If any of the intermediate operations fails, an UNDECIDED conformity response is issued.

Expected results: The call to BioSPI_BSPAttach returns BioAPIERR_INCOMPATIBLE_VERSION.

Assertion language package

```
<package name="0052ac10-0c60-1085-9883-0002a5d5fd2e">
      <author>
            ISO/IEC JTC1 SC37
      </author>

      <description>
            This package contains the assertion "BioSPI_BSPAttach_InvalidVersion" (see the
                              "description" element of the assertion below).
      </description>

      <assertion name="BioSPI_BSPAttach_InvalidVersion" model="BSPTesting">
            <description>
                  This assertion checks if a call to the function BioSPI_BSPAttach returns
                              BioAPIERR_INCOMPATIBLE_VERSION when the caller specifies an incompatible
                              version.
                  The relevant text in BioAPI 2.0 is quoted below from subclauses 9.3.1.3 and 11.2.3.
                  _______________________________________________________
                  BioAPI_RETURN BioAPI BioSPI_BSPAttach
                        (const BioAPI_UUID *BSPUuid,
                        BioAPI_VERSION Version,
                        const BioAPI_UNIT_LIST_ELEMENT *UnitList,
                        uint32_t NumUnits,
                        BioAPI_HANDLE BSPHandle);

                  The biometric service provider shall verify compatibility with the version level
                              specified by Version. If the version is not compatible, then this
                              function fails.
                  _________________________________________________________
                  Subclause 9.3.1.3:
                  Parameters: Version (input) The major and minor version number of the BioAPI
                              specification that the application
                  is expecting the BSP to support. The BSP shall determine whether it is compatible
                              with the required version.
                  Subclause 11.2.3:
                  Errors: BioAPIERR_INCOMPATIBLE_VERSION
                  _________________________________________________________


                  In order to determine conformance with respect to the text above, the following
                              steps are performed:

                        1)    Load the BSP under test.
                        2)    Attach the BSP under test using an invalid version    number.
                        3)    Check the return value of the function call.

                  If any of the intermediate operations fails, an UNDECIDED conformity response is
                              issued.
            </description>

            <!-- UUID of the BSP under test -->
            <input name="_bspUuid"/>

            <!-- Timeout for the BioAPI_NOTIFY_INSERT event -->
            <input name="_inserttimeout"/>

            <!-- Module handle for attached BSP -->
            <input  name="_bsphandle"/>

            <!-- Invocation of the primary activity of this assertion with input parameter values
                              assigned from the assertion's parameters. -->
            <invoke activity="LoadAndAttach" >
                  <input name="bspUuid" var="_bspUuid"/>
                  <input name="unitIDOrNull" value="0"/>
                  <input name="bspHandle" var="_bsphandle"/>
                  <input name="eventtimeouttime" var="_inserttimeout"/>
            </invoke>

            <!-- Activity bound to a function of the framework callback interface exposed by the
                              testing component.  This activity will be automatically invoked on each
                              incoming call to the function to which it is bound. -->
            <bind activity="EventHandler"
                        package="02c59458-0c46-1085-95d7-0002a5d5fd2e"
                        function="BioSPI_EventHandler"/>
      </assertion>
```

```
<activity name="LoadAndAttach">
      <input name="bspUuid"/>
      <input name="unitIDOrNull"/>
      <input name="bspHandle"/>
      <input name="eventtimeouttime"/>

      <!-- Initialize the global variable "_unitIDOrNull" to make it available to the activity
                  "EventHandler" -->
      <set name="_unitIDOrNull" var="unitIDOrNull"/>

      <!-- Initialize the global variable "_insert" to "false". The activity "EventHandler"
                  will set this variable to "true" when a BioAPI_NOTIFY_INSERT event
                  notification is received and will set it to "false" when a
                  BioAPI_NOTIFY_REMOVE event notification is received. -->
      <set name="_insert" value="false"/>

      <set name="eventtimeoutflag" value="false"/>

      <!-- Invoke the function BioSPI_BSPLoad. -->
      <invoke function="BioSPI_BSPLoad">
            <input name="BSPUuid" var="bspUuid"/>
            <input name="BioAPINotifyCallback" value="*"/>
            <input name="BFPEnumerationHandler" value="*"/>
            <input name="MemoryFreeHandler" value="*" />
            <return setvar="return"/>
      </invoke>

      <!-- Issue a conformity response.
                  If the condition specified in the <description> below is false, an UNDECIDED
                        conformity response is issued and the execution of the activity is
                        interrupted, otherwise a PASS conformity response is issued.-->
      <assert_condition response_if_false="undecided"
                  break_if_false="true">
            <description>
                  The function BioSPI_BSPLoad has returned BioAPI_OK
            </description>
            <equal_to var1="return" var2="__BioAPI_OK"/>
      </assert_condition>

      <!-- Wait until the BioAPI_NOTIFY_INSERT event notification has been received, but no
                  longer than the specified maximum duration.-->
      <wait_until timeout_var="eventtimeouttime"
                  setvar="eventtimeoutflag" var="_insert"/>

      <!-- Issue a conformity response.
                  If the condition specified in the <description> below is false, an UNDECIDED
                        conformity response is issued and the execution of the activity is
                        interrupted, otherwise a PASS conformity response is issued.-->
      <assert_condition response_if_false="undecided"
                  break_if_false="true">
            <description>
                  The BioAPI_NOTIFY_INSERT event notification has been received within the
                        specified maximum duration
            </description>
            <not var="eventtimeoutflag"/>
      </assert_condition>

      <!-- Invoke the function BioSPI_BSPAttach passing an invalid version number. -->
            <invoke function="BioSPI_BSPAttach">
            <input name="BSPUuid" var="bspUuid"/>
            <input name="Version" value="1"/>
            <input name="Unit_1_UnitCategory" var="_unitCategory" />
            <input name="Unit_1_UnitID" var="_unitID"/>
            <input name="NumUnits" value="1" />
            <input name="BSPHandle" var="bspHandle"/>
            <return setvar="return"/>
      </invoke>

      <!-- Issue a conformity response.
                  If the condition specified in the <description> below is false, a FAIL
                        conformity response is issued, otherwise a PASS conformity response is
                        issued.-->
      <assert_condition>
            <description>
                  The function BioSPI_BSPAttach has returned BioAPIERR_INCOMPATIBLE_VERSION
            </description>
```

```
                        <equal_to var1="return"
                                    var2="__BioAPIERR_BSP_INCOMPATIBLE_VERSION"/>
                  </assert_condition>

                  <!-- Invoke the function BioSPI_BSPUnload -->
                  <invoke function="BioSPI_BSPUnload" >
                        <input name="BSPUuid" var="bspUuid" />
                        <return setvar="return"/>
                  </invoke>

                  <!-- Issue a conformity response.
                              If the condition specified in the <description> below is false, an UNDECIDED
                                    conformity response is issued, otherwise a PASS conformity response is
                                    issued.-->
                  <assert_condition response_if_false="undecided"
                              break_if_false="true">
                        <description>
                              The function BioSPI_BSPUnload has returned BioAPI_OK
                        </description>
                        <equal_to var1="return" var2="__BioAPI_OK"/>
                  </assert_condition>
            </activity>

</package>
```

8.12 Assertion 3d - *BioSPI_BSPAttach_InvalidBSPHandle*

Description: This assertion checks if the function BioSPI_BSPAttach returns an error when called with an invalid BSP handle.

Excerpts

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.

Parameter BSPHandle (input) - The BioAPI_HANDLE value assigned by the Framework and associated with the attach session being created by this function.

References: 9.3.1.3

Scenario:

1) Load the BSP under test

2) Attach the BSP under test with an invalid BSP handle

3) Check the return value of the function call.

4) Unload the BSP under test.

If any of the intermediate operations fails, an UNDECIDED conformity response is issued.

Expected results: The call to BioSPI_BSPAttach returns BioAPIERR_INVALID_BSP_HANDLE.

Assertion language package

```
<package name="03826830-0c57-1085-bfb0-0002a5d5fd2e">
      <author>
            ISO/IEC JTC1 SC37
      </author>

      <description>
            This package contains the assertion "BioSPI_BSPAttach_InvalidModuleHandle" (see the
                        "description" element of the assertion below).
      </description>

      <assertion name="BioSPI_BSPAttach_InvalidBSPHandle" model="BSPTesting">
            <description>
                  This assertion checks if the function BioSPI_BSPAttach returns an error when called
                        with an invalid BSP handle.
                  The relevant text in BioAPI 2.0 is quoted below from 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.
                  __________________________________________________________
                  Subclause 9.3.1.3.2:
                  Parameter BSPHandle (input)  - The BioAPI_HANDLE value assigned by the Framework
                        and associated with the attach session being created by this function.
                  __________________________________________________________
                  NOTE: Currently, the only indication of the need for special value that represents
                        invalid BSP handle is in the description of BioAPI_BSPAttach (see
                        8.1.7.2):

                  NewBSPHandle (output) - a new handle that can be used to interact with the
                        requested biometric service provider.
                  The value will be set to BioAPIERR_FRAMEWORK_INVALID_BSP_HANDLE if the function
                        fails.
                  __________________________________________________________
                  The name BioAPIERR_FRAMEWORK_INVALID_BSP_HANDLE suggests a BioAPI error, which
                        cannot be used as a handle.
                  Besides, BioAPIERR_FRAMEWORK_INVALID_BSP_HANDLE is not defined.
                  This assertion uses 0 for invalid BSP handle because it is expected that a
                        corrigendum will be released to address this issue.
                  __________________________________________________________

                  In order to determine conformance with respect to the text above, the following
                        steps are performed:

                        1)    Load the BSP under test
                        2)    Attach the BSP under test with an invalid BSP handle
                        3)    Check the return value of the function call.
                        4)    Unload the BSP under test.

                  If any of the intermediate operations fails, an UNDECIDED conformity response is
                        issued.
            </description>

            <!-- UUID of the BSP under test -->
            <input name="_bspUuid"/>

            <!-- version number of the BSP under test -->
            <input name="_bspVersion"/>

            <!-- Timeout for the BioAPI_NOTIFY_INSERT event -->
            <input name="_inserttimeout"/>

            <!-- Invocation of the primary activity of this assertion with input parameter values
                        assigned from the assertion's parameters. -->
            <invoke activity="LoadAndAttach" >
                  <input name="bspUuid" var="_bspUuid"/>
            <input name="bspVersion" var="_bspVersion"/>
            <input name="unitIDOrNull" value="0"/>
            <input name="eventtimeouttime" var="_inserttimeout"/>
      </invoke>

      <!-- Activity bound to a function of the framework callback interface exposed by the
                  testing component.  This activity will be automatically invoked on each
                  incoming call to the function to which it is bound. -->
      <bind activity="EventHandler"
                  package="02c59458-0c46-1085-95d7-0002a5d5fd2e"
                  function="BioSPI_EventHandler"/>
</assertion>

<activity name="LoadAndAttach">
      <input name="bspUuid"/>
      <input name="bspVersion"/>
      <input name="unitIDOrNull"/>
      <input name="eventtimeouttime"/>

      <!-- Initialize the global variable "_unitIDOrNull" to make it available to the activity
                  "EventHandler" -->
      <set name="_unitIDOrNull" var="unitIDOrNull"/>

      <!-- Initialize the global variable "_insert" to "false".  The activity "EventHandler"
                  will set this variable to "true" when a BioAPI_NOTIFY_INSERT event
                  notification is received, and will set it to "false" when a
                  BioAPI_NOTIFY_REMOVE event notification is received. -->
      <set name="_insert" value="false"/>

      <set name="eventtimeoutflag" value="false"/>

      <!-- Invoke the function BioSPI_BSPLoad. -->
      <invoke function="BioSPI_BSPLoad">
            <input name="BSPUuid" var="bspUuid"/>
            <input name="BioAPINotifyCallback" value="*"/>
            <input name="BFPEnumerationHandler" value="*"/>
            <input name="MemoryFreeHandler" value="*" />
            <return setvar="return"/>
      </invoke>

      <!-- Issue a conformity response.
                  If the condition specified in the <description> below is false, an UNDECIDED
                        conformity response is issued and the execution of the activity is
                        interrupted, otherwise a PASS conformity response is issued.-->
      <assert_condition response_if_false="undecided"
                  break_if_false="true">
            <description>
                  The function BioSPI_BSPLoad has returned BioAPI_OK
            </description>
            <equal_to var1="return" var2="__BioAPI_OK"/>
      </assert_condition>

      <!-- Wait until the BioAPI_NOTIFY_INSERT event notification has been received, but no
                  longer than the specified maximum duration.-->
      <wait_until timeout_var="eventtimeouttime"
                  setvar="eventtimeoutflag" var="_insert"/>

      <!-- Issue a conformity response.
                  If the condition specified in the <description> below is false, an UNDECIDED
                        conformity response is issued and the execution of the activity is
                        interrupted, otherwise a PASS conformity response is issued.-->
      <assert_condition response_if_false="undecided"
                  break_if_false="true">
            <description>
                  The BioAPI_NOTIFY_INSERT event notification has been received within the
                        specified maximum duration
            </description>
            <not var="eventtimeoutflag"/>
      </assert_condition>

      <!-- Invoke the function BioSPI_BSPAttach, passing an invalid value for the input
                  parameter "BSPHandle". -->
      <invoke function="BioSPI_BSPAttach">
            <input name="BSPUuid" var="bspUuid"/>
            <input name="Version" var="bspVersion"/>
            <input name="Unit_1_UnitCategory" var="_unitCategory" />
            <input name="Unit_1_UnitID" var="_unitID"/>
            <input name="NumUnits" value="1" />
            <input name="BSPHandle" value="0"/>
            <return setvar="return"/>
      </invoke>

      <!-- Issue a conformity response.
                  If the condition specified in the <description> below is false, a FAIL
                        conformity response is issued, otherwise a PASS conformity response is
                        issued.-->
      <assert_condition>
            <description>
                  The function BioSPI_BSPAttach has returned BioAPIERR_INVALID_BSP_HANDLE
            </description>
            <equal_to var1="return"
                        var2="__BioAPIERR_BSP_INVALID_BSP_HANDLE"/>
      </assert_condition>

      <!-- Invoke the function BioSPI_BSPUnload -->
      <invoke function="BioSPI_BSPUnload" >
            <input name="BSPUuid" var="bspUuid" />
            <return setvar="return"/>
      </invoke>

      <!-- Issue a conformity response.
                  If the condition specified in the <description> below is false, an UNDECIDED
                        conformity response is issued, otherwise a PASS conformity response is
                        issued.-->
      <assert_condition response_if_false="undecided"
                  break_if_false="true">
            <description>
                  The function BioSPI_BSPUnload has returned BioAPI_OK
            </description>
            <equal_to var1="return" var2="__BioAPI_OK"/>
      </assert_condition>
</activity>

</package>
```

8.13 Assertion 4a - *BioSPI_BSPDetach_ValidParam*

Description: This assertion checks if a call to the function BioSPI_BSPDetach with a valid module handle returns BioAPI_OK.

Excerpts

Subclause 9.3.1.4

BioAPI_RETURN BioAPI BioSPI_BSPDetach

(BioAPI_HANDLE BSPHandle);

This function is invoked by BioAPI once for each invocation of BioAPI_BSPDetach specifying the attach session identified by BSPHandle. The biometric service provider shall perform all cleanup operations associated with the specified attach handle.

Subclause A.4

This function should be supported by all types of BSPs.

References: 9.3.1.4 and A.4

Scenario:

1) Load the BSP under test.

2) Attach the BSP under test.

3) Detach the BSP under test using the module handle.

4) Check the return value.

If any of the intermediate operations fails, an UNDECIDED conformity response is issued.

Expected results: The call to BioSPI_BSPDetach returns BioAPI_OK

Assertion language package

```
<package name="00e0d2b0-0c7a-1085-b8ac-0002a5d5fd2e">
      <author>
            ISO/IEC JTC1 SC37
      </author>

      <description>
            This package contains the assertion "BioSPI_BSPDetach_ValidParam" (see the "description"
                              element of the assertion below).
      </description>

      <assertion name="BioSPI_BSPDetach_ValidParam" model="BSPTesting">
            <description>
                  This assertion checks if a call to the function BioSPI_BSPDetach with a valid
                              module handle returns BioAPI_OK.
                  The relevant text in BioAPI 2.0 is quoted below from subclauses 9.3.1.4.
                  _______________________________________________________
                  BioAPI_RETURN BioAPI BioSPI_BSPDetach
                        (BioAPI_HANDLE BSPHandle);

                  This function is invoked by BioAPI once for each invocation of BioAPI_BSPDetach
                              specifying the attach session identified by BSPHandle. The service
                              provider shall perform all cleanup operations associated with the
                              specified attach handle.
                  ________________________________________________________
                  Subclause A.4:
                  This function should be supported by all types of BSPs.
                  ________________________________________________________

                  In order to determine conformance with respect to the text above, the following
                              steps are performed:

                        1)    Load the BSP under test.
                        2)    Attach the BSP under test.
                        3)    Detach the BSP under test using the module handle.
                        4)    Check the return value.

                  If any of the intermediate operations fails, an UNDECIDED conformity response is
                              issued.
            </description>

            <!-- UUID of the BSP under test -->
            <input name="_bspUuid"/>

            <!-- Timeout for the BioAPI_NOTIFY_INSERT event -->
            <input name="_inserttimeout"/>

            <!-- Module handle for the BSP -->
            <input  name="_bsphandle"/>

            <!-- Invocation of the primary activity of this assertion with input parameter values
                              assigned from the assertion's parameters. -->
            <invoke activity="LoadAndAttachAndDetach" >
                  <input name="bspUuid" var="_bspUuid"/>
                  <input name="unitIDOrNull" value="0"/>
                  <input name="bspHandle" var="_bsphandle"/>
                  <input name="eventtimeouttime" var="_inserttimeout"/>
            </invoke>

            <!-- Activity bound to a function of the framework callback interface exposed by the
                              testing component.  This activity will be automatically invoked on each
                              incoming call to the function to which it is bound. -->
            <bind activity="EventHandler"
                        package="02c59458-0c46-1085-95d7-0002a5d5fd2e"
                  function="BioSPI_EventHandler"/>
      </assertion>

      <activity name="LoadAndAttachAndDetach">
            <input name="bspUuid"/>
            <input name="unitIDOrNull"/>
            <input name="bspHandle"/>
            <input name="eventtimeouttime"/>

            <!-- Invoke the functions BioSPI_BSPLoad and BioSPI_BSPAttach exposed by the BSP under
                              test. -->
            <invoke activity="LoadAndAttach"
                        package="02c59458-0c46-1085-95d7-0002a5d5fd2e"
                        break_on_break="true">
                  <input name="bspUuid" var="bspUuid"/>
                  <input name="bspVersion" value="32"/>
                  <input name="unitIDOrNull" var="unitIDOrNull"/>
                  <input name="bspHandle" var="bspHandle"/>
                  <input name="eventtimeouttime" var="eventtimeouttime"/>
            </invoke>

            <!-- Invoke the function BioSPI_BSPDetach -->
            <invoke function="BioSPI_BSPDetach">
                  <input name="BSPHandle" var="bspHandle"/>
                  <return setvar="return"/>
            </invoke>

            <!-- Issue a conformity response.
                        If the condition specified in the <description> below is false, a FAIL
                              conformity response is issued, otherwise a PASS conformity response is
                              issued.-->
            <assert_condition>
                  <description>
                        The function BioSPI_BSPDetach has returned BioAPI_OK
                  </description>
                  <equal_to var1="return" var2="__BioAPI_OK"/>
            </assert_condition>

            <!-- Invoke the function BioSPI_BSPUnload -->
            <invoke function="BioSPI_BSPUnload" >
                  <input name="BSPUuid" var="bspUuid"/>
                  <return setvar="return"/>
            </invoke>

            <!-- Issue a conformity response.
                        If the condition specified in the <description> below is false, an UNDECIDED
                              conformity response is issued and the execution of the activity is
                              interrupted, otherwise a PASS conformity response is issued.-->
            <assert_condition response_if_false="undecided"
                        break_if_false="true">
                  <description>
                        The function BioSPI_BSPUnload has returned BioAPI_OK.
                  </description>
                  <equal_to var1="return" var2="__BioAPI_OK"/>
            </assert_condition>
      </activity>

</package>
```

8.14 Assertion 4b - *BioSPI_BSPDetach_InvalidBSPHandle*

Description: This assertion checks if a call to BioSPI_BSPDetach with an invalid module handle returns an error.

Excerpts

Subclause 9.3.1.4

BioAPI_RETURN BioAPI BioSPI_BSPDetach

(BioAPI_HANDLE BSPHandle);

This function is invoked by BioAPI once for each invocation of BioAPI_BSPDetach specifying the attach session identified by BSPHandle.

BSPHandle (input) - The handle associated with the attach session being terminated by this function.

Subclause 8.1.8.1

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.

References: 9.3.1.4 and 8.1.8.1

Scenario:

1) Load the BSP under test.

2) Attach the BSP under test.

3) Detach the BSP under test using an invalid module handle.

4) Check the return value of the function call.

If any of the intermediate operations fails, an UNDECIDED conformity response is issued.

Expected results: The call to BioSPI_BSPDetach returns BioAPIERR_INVALID_BSP_HANDLE.

Assertion language package

```
<package name="0434c458-0c79-1085-9f2c-0002a5d5fd2e">
      <author>
            ISO/IEC JTC1 SC37
      </author>

      <description>
            This package contains the assertion "BioSPI_BSPDetach_InvalidModuleHandle" (see the
                        "description" element of the assertion below).
      </description>

      <assertion name="BioSPI_BSPDetach_InvalidBSPHandle" model="BSPTesting">
            <description>
                  This assertion checks if a call to BioSPI_BSPDetach with an invalid module handle
                        returns an error.
                  The relevant text in BioAPI 2.0 is quoted below from subclauses 9.3.1.4 and
                        8.1.8.1.
                  _______________________________________________________
                  BioAPI_RETURN BioAPI BioSPI_BSPDetach
                        (BioAPI_HANDLE BSPHandle);

                  This function is invoked by BioAPI once for each invocation of BioAPI_BSPDetach
                        specifying the attach session identified by BSPHandle.
            _______________________________________________________
            Subclause 8.1.8.1:
            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.
            _______________________________________________________
            Subclause 9.3.1.4:
            BSPHandle (input) - The handle associated with the attach session being terminated
                        by this function.
            _______________________________________________________

            In order to determine conformance with respect to the text above, the following
                        steps are performed:

                  1)    Load the BSP under test.
                  2)    Attach the BSP under test.
                  3)    Detach the BSP under test using an invalid module handle.
                  4)    Check the return value of the function call.

            If any of the intermediate operations fails, an UNDECIDED conformity response is
                        issued.
      </description>

      <!-- UUID of the BSP under test -->
      <input name="_bspUuid"/>

      <!-- Timeout for the BioAPI_NOTIFY_INSERT event -->
      <input name="_inserttimeout"/>

      <!-- Module handle for attached BSP -->
      <input  name="_bsphandle"/>

      <!-- Invocation of the primary activity of this assertion with input parameter values
                        assigned from the assertion's parameters. -->
      <invoke activity="LoadAndAttachAndDetach" >
            <input name="bspUuid" var="_bspUuid"/>
            <input name="unitIDOrNull" value="0"/>
            <input name="bsphandle" var="_bsphandle"/>
            <input name="eventtimeouttime" var="_inserttimeout"/>
      </invoke>

      <!-- Activity bound to a function of the framework callback interface exposed by the
                        testing component.  This activity will be automatically invoked on each
                        incoming call to the function to which it is bound. -->
      <bind activity="EventHandler"
                  package="02c59458-0c46-1085-95d7-0002a5d5fd2e"
                  function="BioSPI_EventHandler"/>
</assertion>

<activity name="LoadAndAttachAndDetach">
      <input name="bspUuid"/>
      <input name="unitIDOrNull"/>
      <input name="bsphandle"/>
      <input name="eventtimeouttime"/>

      <!-- Invoke the functions BioSPI_BSPLoad and BioSPI_BSPAttach exposed by the BSP under
                        test. -->
      <invoke activity="LoadAndAttach"
                  package="02c59458-0c46-1085-95d7-0002a5d5fd2e"
                  break_on_break="true">
            <input name="bspUuid" var="bspUuid"/>
            <input name="bspVersion" value="32"/>
            <input name="unitIDOrNull" var="unitIDOrNull"/>
            <input name="bspHandle" var="bsphandle"/>
            <input name="eventtimeouttime" var="eventtimeouttime"/>
      </invoke>

      <!-- Invoke the function BioSPI_BSPDetach passing an invalid module handle. -->
      <invoke function="BioSPI_BSPDetach">
            <input name="BSPHandle" value="0"/>
            <return setvar="return"/>
      </invoke>

      <!-- Issue a conformity response.
```

```
                    If the condition specified in the <description> below is false, a FAIL
                          conformity response is issued, otherwise a PASS conformity response is
                          issued.-->
            <assert_condition>
                  <description>
                        The function BioSPI_BSPDetach has returned BioAPIERR_INVALID_BSP_HANDLE
                  </description>
                  <equal_to var1="return"
                          var2="__BioAPIERR_BSP_INVALID_BSP_HANDLE"/>
            </assert_condition>

            <!-- Invoke the functions BioSPI_BSPDetach and BioSPI_BSPUnload -->
            <invoke activity="DetachAndUnload"
                        package="02c59458-0c46-1085-95d7-0002a5d5fd2e" >
                  <input name="bspUuid" var="bspUuid" />
                  <input name="BSPHandle" var="bsphandle" />
            </invoke>
      </activity>

</package>
```

8.15 Assertion 4c - *BioSPI_BSPDetach_Confirm*

Description: This assertion checks if a call to the function BioSPI_BSPDetach truly terminates the attach session.

Excerpts

Subclause 9.3.1.4

BioAPI_RETURN BioAPI BioSPI_BSPDetach

(BioAPI_HANDLE BSPHandle);

This function is invoked by BioAPI once for each invocation of BioAPI_BSPDetach specifying the attach session identified by BSPHandle. The service provider shall perform all cleanup operations associated with the specified attach handle.

Subclause 8.1.8.4

Errors: BioAPIERR_INVALID_BSP_HANDLE

References: 9.3.1.4 and 8.1.8.4

Scenario:

1) Load the BSP under test

2) Attach the BSP under test

3) Detach the BSP under test using the valid module handle

4) Check the return value of the function call.

5) Call BioSPI_Enroll. This function is expected to fail with the error BioAPIERR_INVALID_BSP_HANDLE.

If any of the intermediate operations fails, an UNDECIDED conformity response is issued.

Expected results: The call to BioSPI_Enroll returns BioAPIERR_INVALID_BSP_HANDLE.

Assertion language package

```
<package name="002e7e58-0c78-1085-9e1d-0002a5d5fd2e">
      <author>
            ISO/IEC JTC1 SC37
      </author>

      <description>
            This package contains the assertion "BioSPI_BSPDetach_Confirm" (see the "description"
                              element of the assertion below).
      </description>

      <assertion name="BioSPI_BSPDetach_Confirm" model="BSPTesting">
            <description>
                  This assertion checks if a call to the function BioSPI_BSPDetach truly terminates
                              the attach session.
                  The relevant text in BioAPI 2.0 is quoted below from subclause 9.3.1.4.
                  _______________________________________________________
                  BioAPI_RETURN BioAPI BioSPI_BSPDetach
                        (BioAPI_HANDLE BSPHandle);

                  This function is invoked by BioAPI once for each invocation of BioAPI_BSPDetach
                              specifying the attach session identified by BSPHandle. The service
                              provider shall perform all cleanup operations associated with the
                              specified attach handle.
                  _______________________________________________________
                  8.1.8.4 Errors
                        BioAPIERR_INVALID_BSP_HANDLE
                  ________________________________________________________

                  In order to determine conformance with respect to the text above, the following
                              steps are performed:

                        1)    Load the BSP under test
                        2)    Attach the BSP under test
                        3)    Detach the BSP under test using the valid module handle
                        4)    Check the return value of the function call.
                        5)    Call BioSPI_Enroll. This function is expected to fail with the error
                              BioSPI_BSPDetach.

                  If any of the intermediate operations fails, an UNDECIDED conformity response is
                              issued.
            </description>

            <!-- UUID of the BSP under test -->
            <input name="_bspUuid"/>

            <!-- Timeout for the BioAPI_NOTIFY_INSERT event -->
            <input name="_inserttimeout"/>

            <!-- Module handle for attached BSP -->
            <input name="_bsphandle"/>

            <!-- Invocation of the primary activity of this assertion with input parameter values
                              assigned from the assertion's parameters. -->
            <invoke activity="LoadAndAttachAndDetach" >
                  <input name="bspUuid" var="_bspUuid"/>
                  <input name="unitIDOrNull" value="0"/>
                  <input name="bspHandle" var="_bsphandle"/>
                  <input name="eventtimeouttime" var="_inserttimeout"/>
            </invoke>

            <!-- Activity bound to a function of the framework callback interface exposed by the
                              testing component.  This activity will be automatically invoked on each
                              incoming call to the function to which it is bound. -->
            <bind activity="EventHandler"
                        package="02c59458-0c46-1085-95d7-0002a5d5fd2e"
                        function="BioSPI_EventHandler"/>
      </assertion>

      <activity name="LoadAndAttachAndDetach">
            <input name="bspUuid"/>
            <input name="unitIDOrNull"/>
            <input name="bspHandle"/>
            <input name="eventtimeouttime"/>
            <!-- Invoke the functions BioSPI_BSPLoad and BioSPI_BSPAttach exposed by the BSP under
                              test. -->
            <invoke activity="LoadAndAttach"
                        package="02c59458-0c46-1085-95d7-0002a5d5fd2e"
                        break_on_break="true">
                  <input name="bspUuid" var="bspUuid"/>
                  <input name="bspVersion" value="32"/>
                  <input name="unitIDOrNull" var="unitIDOrNull"/>
                  <input name="bspHandle" var="bspHandle"/>
                  <input name="eventtimeouttime" var="eventtimeouttime"/>
            </invoke>

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

            <!-- Issue a conformity response.
                        If the condition specified in the <description> below is false, a FAIL
                              conformity response is issued and the execution of the activity is
                              interrupted, otherwise a PASS conformity response is issued.-->
            <assert_condition break_if_false="true">
                  <description>
                        The function BioSPI_BSPDetach has returned BioAPI_OK.
                  </description>
                  <equal_to var1="return" var2="__BioAPI_OK"/>
            </assert_condition>

            <!-- Invoke the function BioSPI_Enroll.  This function is expected to return an error.
                              This confirms that the BSP is no longer attached. -->
            <invoke function="BioSPI_Enroll">
                  <input name="BSPHandle" var="bspHandle"/>
                  <input name="Purpose"
                              var="__BioAPI_PURPOSE_ENROLL_FOR_VERIFICATION_ONLY"/>
                  <input name="Subtype" value="0"/>
                  <input name="Timeout" value="15000"/>
                  <output name="NewTemplate" setvar="newtemplate_handle"/>
                  <return setvar="return"/>
            </invoke>

            <!-- Issue a conformity response.
                        If the condition specified in the <description> below is false, a FAIL
                              conformity response is issued, otherwise a PASS conformity response is
                              issued.-->
            <assert_condition>
                  <description>
                        The function BioSPI_Enroll has returned BioAPIERR_INVALID_BSP_HANDLE.
                  </description>
                  <equal_to var1="return" var2="__BioAPIERR_BSP_INVALID_BSP_HANDLE"/>
            </assert_condition>

            <!-- Invoke the function BioSPI_BSPUnload. -->
            <invoke function="BioSPI_BSPUnload" >
                  <input name="BSPUuid" var="bspUuid"/>
                  <return setvar="return"/>
            </invoke>

            <!-- Issue a conformity response.
                        If the condition specified in the <description> below is false, an UNDECIDED
                              conformity response is issued, otherwise a PASS conformity response is
                              issued.-->
            <assert_condition response_if_false="undecided">
                  <description>
                        The function BioSPI_BSPUnload has returned BioAPI_OK.
                  </description>
                  <equal_to var1="return" var2="__BioAPI_OK"/>
            </assert_condition>
      </activity>

</package>
```

8.16 Assertion 5a - *BioSPI_FreeBIRHandle_ValidParam*

Description: This assertion checks if calling the function BioSPI_FreeBIRHandle with a valid BIR handle frees the BIR handle.

Excerpts

Subclause 9.3.2.1

BioAPI_RETURN BioAPI BioSPI_FreeBIRHandle

(BioAPI_HANDLE BSPHandle,

BioAPI_BIR_HANDLE Handle);

Subclause 8.2.1

This function frees the memory and resources associated with the specified BIR Handle. The associated BIR is no longer referenceable through that handle.

Subclause A.4

This function should be supported by all types of BSPs.

References: 9.3.2.1, 8.2.1, and A.4.

Scenario:

1) Load the BSP under test.

2) Attach the BSP under test.

3) Enroll to obtain a BIR handle.

4) Invoke the function BioSPI_FreeBIRHandle to free the BIR handle.

5) Call BioSPI_GetBIRFromHandle, which is expected to return an BioAPIERR_INVALID_BIR_HANDLE error code.

If any of the intermediate operations fails, an UNDECIDED conformity response is issued.

Expected results: The call to BioSPI_FreeBIRHandle returns BioAPI_OK and the subsequent call to BioSPI_GetBIRFromHandle returns BioAPIERR_INVALID_BIR_HANDLE.

Assertion language package

```
<package name="0280a7d0-0c80-1085-a9a0-0002a5d5fd2e">
      <author>
            ISO/IEC JTC1 SC37
      </author>

      <description>
            This package contains the assertion "BioSPI_FreeBIRHandle_ValidParam" (see the
                              "description" element of the assertion below).
      </description>

      <assertion name="BioSPI_FreeBIRHandle_ValidParam" model="BSPTesting">
            <description>
                  This assertion checks if calling the function BioSPI_FreeBIRHandle with a valid BIR
                              handle frees the BIR handle.
                  The relevant text in BioAPI 2.0 is quoted below from subclauses A.4 and 9.3.2.1,
                              8.2.1.
```

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

          NOTE: Details of the function definition are located in clause 8.2.1,
                      BioAPI_FreeBIRHandle.

          ________________________________________________________
          Subclause 8.2.1:
          This function frees the memory and resources associated with the specified BIR
                      Handle. The associated BIR is no longer referenceable through that
                      handle.
          ________________________________________________________
          Subclause A.4:
          This function should be supported by all types of BSPs.
          ________________________________________________________

          In order to determine conformance with respect to the text above, the following
                      steps are performed:

                1)    Load the BSP under test.
                2)    Attach the BSP under test.
                3)    Enroll to obtain a BIR handle.
                4)    Invoke the function BioSPI_FreeBIRHandle to free the BIR handle.
                5)    Call BioSPI_GetBIRFromHandle, which is expected to return an
                      BioAPIERR_INVALID_BIR_HANDLE error code.

          If any of the intermediate operations fails, an UNDECIDED conformity response is
                      issued.
    </description>

    <!-- UUID of the BSP under test -->
    <input name="_bspUuid"/>

    <!-- Timeout for the BioAPI_NOTIFY_INSERT event -->
    <input name="_inserttimeout"/>

    <!-- Indicates whether the BSP under test does not claim support for the
                      BioAPI_NOTIFY_SOURCE_PRESENT event notification -->
    <input name="_noSourcePresentSupported"/>

    <!-- Timeout for the BioAPI_NOTIFY_SOURCE_PRESENT event -->
    <input name="_sourcepresenttimeout"/>

    <!-- Timeout for BioSPI_Enroll -->
    <input name="_capturetimeout"/>

    <!-- Invocation of the primary activity of this assertion with input parameter values
                      assigned from the assertion's parameters. -->
    <invoke activity="BioSPI_FreeBIRHandle">
          <input name="bspUuid" var="_bspUuid"/>
          <input name="inserttimeouttime" var="_inserttimeout"/>
          <input name="nosourcepresentsupported"
                      var="_noSourcePresentSupported"/>
          <input name="sourcepresenttimeouttime"
                      var="_sourcepresenttimeout"/>
          <input name="capturetimeouttime" var="_capturetimeout"/>
    </invoke>

    <!-- Activity bound to a function of the framework callback interface exposed by the
                      testing component.  This activity will be automatically invoked on each
                      incoming call to the function to which it is bound. -->
    <bind activity="EventHandler"
                package="02c59458-0c46-1085-95d7-0002a5d5fd2e"
                function="BioSPI_EventHandler"/>
</assertion>

<activity name="BioSPI_FreeBIRHandle">
    <input name="bspUuid"/>
    <input name="inserttimeouttime"/>
    <input name="nosourcepresentsupported"/>
    <input name="sourcepresenttimeouttime"/>
    <input name="capturetimeouttime"/>
```

```
<!-- This assertion will use BSPHandle "1" for all BioSPI calls that require it -->
<set name="_bsphandle" value="1"/>

<!-- Invoke the functions BioSPI_BSPLoad and BioSPI_BSPAttach exposed by the BSP under
                  test.-->
<invoke activity="LoadAndAttach"
            package="02c59458-0c46-1085-95d7-0002a5d5fd2e"
            break_on_break="true">
      <input name="bspUuid" var="bspUuid"/>
      <input name="bspVersion" value="32"/>
      <input name="unitIDOrNull" value="0"/>
      <input name="bspHandle" var="_bsphandle"/>
      <input name="eventtimeouttime" var="inserttimeouttime"/>
</invoke>

<set name="eventtimeoutflag" value="false"/>

<!-- If the BSP under test claims support for the BioAPI_NOTIFY_SOURCE_PRESENT event
                  notification, wait until that notification has been received, but no
                  longer than the specified maximum duration.-->
<wait_until timeout_var="sourcepresenttimeouttime"
            setvar="eventtimeoutflag">
      <or var1="nosourcepresentsupported" var2="_sourcePresent"/>
</wait_until>

<!-- Issue a conformity response.
            If the condition specified in the <description> below is false, an UNDECIDED
                  conformity response is issued and the execution of the activity is
                  interrupted, otherwise a PASS conformity response is issued.-->
<assert_condition response_if_false="undecided"
            break_if_false="true">
      <description>
            Either the BSP under test does not claim support for the
                  BioAPI_NOTIFY_SOURCE_PRESENT event notification, or the event
                  notification has been received within the specified maximum duration.
      </description>
      <not var="eventtimeoutflag"/>
</assert_condition>

<!-- The BSP is ready to capture. Invoke the function BioSPI_Enroll for the purpose of
                  enrollment. -->
<invoke function="BioSPI_Enroll">
      <input name="BSPHandle" var="_bsphandle"/>
      <input name="Purpose"
                  var="__BioAPI_PURPOSE_ENROLL_FOR_VERIFICATION_ONLY"/>
      <input name="Subtype" value="0"/>
      <input name="Timeout" var="capturetimeouttime"/>
      <output name="NewTemplate" setvar="newtemplate_handle"/>
      <return setvar="return"/>
</invoke>

<!-- Issue a conformity response.
            If the condition specified in the <description> below is false, an UNDECIDED
                  conformity response is issued and the execution of the activity is
                  interrupted, otherwise a PASS conformity response is issued.-->
<assert_condition response_if_false="undecided"
            break_if_false="true">
      <description>
            The function BioSPI_Enroll has returned BioAPI_OK
      </description>
      <equal_to var1="return" var2="__BioAPI_OK"/>
</assert_condition>

<!-- Invoke the function BioSPI_FreeBIRHandle passing the enrolled BIR handle. -->
<invoke function="BioSPI_FreeBIRHandle">
      <input name="BSPHandle" var="_bsphandle"/>
      <input name="Handle" var=" newtemplate_handle"/>
      <return setvar="return"/>
</invoke>

<!-- Issue a conformity response.
            If the condition specified in the <description> below is false, a FAIL
                  conformity response is issued and the execution of the activity is
                  interrupted, otherwise a PASS conformity response is issued.-->
<assert_condition break_if_false="true">
      <description>
```

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

        <!-- Invoke the function BioSPI_GetBIRFromHandle passing the enrolled BIR handle to see
                    if the handle has been freed. -->
        <invoke function="BioSPI_GetBIRFromHandle">
            <input name="BSPHandle" var="_bsphandle"/>
            <input name="Handle" var="newtemplate_handle"/>
            <output name="HeaderVersion" setvar="headerversion"/>
            <output name="ProcessedLevel" setvar="processedlevel"/>
            <output name="FormatOwner" setvar="formatowner"/>
            <output name="FormatType" setvar="formattype"/>
            <output name="Quality" setvar="quality"/>
            <output name="Purpose" setvar="purpose"/>
            <output name="ProductOwner" setvar="productowner" />
            <output name="ProductType" setvar="producttype" />
            <output name="Creation_Year" setvar="creationyear" />
            <output name="Creation_Month" setvar="creationmonth" />
            <output name="Creation_Day" setvar="creationday" />
            <output name="Creation_Hour" setvar="creationhour" />
            <output name="Creation_Minute" setvar="creationminute" />
            <output name="Creation_Second" setvar="creationsecond" />
            <output name="Expiration_Year" setvar="expirationyear" />
            <output name="Expiration_Month" setvar="expirationmonth" />
            <output name="Expiration_Day" setvar="expirationday" />
            <output name="SBFormatOwner" setvar="securityformatowner" />
            <output name="SBFormatType" setvar="securityformattype" />
            <output name="Index" setvar="index" />
            <output name="BiometricData" setvar="biometircdata" />
            <output name="SecurityBlock" setvar="securityblock"/>
            <return setvar="return"/>
        </invoke>

        <!-- Issue a conformity response.
                If the condition specified in the <description> below is false, an UNDECIDED
                    conformity response is issued, otherwise a PASS conformity response is
                    issued.-->
        <assert_condition response_if_false="undecided">
            <description>
                The function BioSPI_GetHeaderFromHandle has returned
                    BioAPIERR_INVALID_BIR_HANDLE.
            </description>
            <equal_to var1="return" var2="__BioAPIERR_BSP_INVALID_BIR_HANDLE"/>
        </assert_condition>

        <!-- Invoke the functions BioSPI_BSPDetach and BioSPI_BSPUnload -->
        <invoke activity="DetachAndUnload"
                package="02c59458-0c46-1085-95d7-0002a5d5fd2e" >
            <input name="bspUuid" var="bspUuid"/>
            <input name="BSPHandle" var="_bsphandle"/>
        </invoke>
    </activity>

</package>
```

8.17 Assertion 5b - *BioSPI_FreeBIRHandle_InvalidBSPHandle*

Description: This assertion checks if calling BioSPI_FreeBIRHandle with an invalid BSP handle returns an error.

Excerpts

Subclause 9.3.2.1

BioAPI_RETURN BioAPI BioSPI_FreeBIRHandle

(BioAPI_HANDLE BSPHandle,

BioAPI_BIR_HANDLE Handle);

Parameters: BSPHandle (input) - The handle of an attached BSP.

Subclause 8.2.1

This function frees the memory and resources associated with the specified BIR Handle.

References: 9.3.2.1 and 8.2.1.

Scenario:

1) Load the BSP under test
2) Attach the BSP under test
3) Enroll to obtain a BIR handle
4) Call BioSPI_FreeBIRHandle with an invalid BSP handle
5) Check the return code.

If any of the intermediate operations fails, an UNDECIDED conformity response is issued.

Expected results: The call to BioSPI_FreeBIRHandle returns BioAPIERR_INVALID_BSP_HANDLE.

Assertion language package

```
<package name="047aed48-0c80-1085-898b-0002a5d5fd2e">
      <author>
            ISO/IEC JTC1 SC37
      </author>

      <description>
            This package contains the assertion "BioSPI_FreeBIRHandle_InvalidBSPHandle" (see the
                              "description" element of the assertion below).
      </description>

      <assertion name="BioSPI_FreeBIRHandle_InvalidBSPHandle" model="BSPTesting">
            <description>
                  This assertion checks if calling BioSPI_FreeBIRHandle with an invalid BSP handle
                              returns an error.
                  The relevant text in BioAPI 2.0 is quoted below from subclauses 9.3.2.1, 8.2.1.
                  _________________________________________________________
                  BioAPI_RETURN BioAPI BioSPI_FreeBIRHandle
                        (BioAPI_HANDLE BSPHandle,
                        BioAPI_BIR_HANDLE Handle);

                  NOTE: Details of the function definition are located in clause 8.2.1,
                              BioAPI_FreeBIRHandle.
```

```
                    ________________________________________________________
                    Subclause 8.2.1:
                    This function frees the memory and resources associated with the specified BIR
                                Handle.
                    ________________________________________________________
                    Subclause 9.3.2.1:
                    Parameters: BSPHandle (input) - The handle of an attached BSP.
                    ________________________________________________________

                    In order to determine conformance with respect to the text above, the following
                                steps are performed:

                          1)    Load the BSP under test
                          2)    Attach the BSP under test
                          3)    Enroll to obtain a BIR handle
                          4)    Call BioSPI_FreeBIRHandle with an invalid BSP handle
                          5)    Check the return code.

                    If any of the intermediate operations fails, an UNDECIDED conformity response is
                                issued.
              </description>

              <!-- UUID of the BSP under test -->
              <input name="_bspUuid"/>

              <!-- Timeout for the BioAPI_NOTIFY_INSERT event -->
              <input name="_inserttimeout"/>

              <!-- Indicates whether the BSP under test does not claim support for the
                                BioAPI_NOTIFY_SOURCE_PRESENT event notification -->
              <input name="_noSourcePresentSupported" />

              <!-- Timeout for the BioAPI_NOTIFY_SOURCE_PRESENT event -->
              <input name="_sourcepresenttimeout"/>

              <!-- Timeout for BioSPI_Enroll -->
              <input name="_capturetimeout"/>

              <!-- Invocation of the primary activity of this assertion with input parameter values
                                assigned from the assertion's parameters. -->
              <invoke activity="BioSPI_FreeBIRHandle">
                    <input name="bspUuid" var="_bspUuid"/>
                    <input name="inserttimeouttime" var="_inserttimeout"/>
                    <input name="nosourcepresentsupported"
                                var="_noSourcePresentSupported" />
                    <input name="sourcepresenttimeouttime"
                                var="_sourcepresenttimeout"/>
                    <input name="capturetimeouttime" var="_capturetimeout"/>
              </invoke>

              <!-- Activity bound to a function of the framework callback interface exposed by the
                                testing component.  This activity will be automatically invoked on each
                                incoming call to the function to which it is bound. -->
              <bind activity="EventHandler"
                          package="02c59458-0c46-1085-95d7-0002a5d5fd2e"
                          function="BioSPI_EventHandler"/>
        </assertion>

        <activity name="BioSPI_FreeBIRHandle">
              <input name="bspUuid"/>
              <input name="inserttimeouttime"/>
              <input name="nosourcepresentsupported" />
              <input name="sourcepresenttimeouttime"/>
              <input name="capturetimeouttime"/>

              <!-- This assertion will use BSPHandle "1" for all BioSPI calls that require it -->
              <set name="_bsphandle" value="1"/>

              <!-- Invoke the functions BioSPI_BSPLoad and BioSPI_BSPAttach exposed by the BSP under
                                test. -->
              <invoke activity="LoadAndAttach"
                          package="02c59458-0c46-1085-95d7-0002a5d5fd2e"
                          break_on_break="true">
                    <input name="bspUuid" var="bspUuid"/>
                    <input name="bspVersion" value="32"/>
                    <input name="unitIDOrNull" value="0"/>
```

```
      <input name="bspHandle" var="_bsphandle"/>
      <input name="eventtimeouttime" var="inserttimeouttime"/>
</invoke>

<set name="eventtimeoutflag" value="false"/>
<!-- If the BSP under test claims support for the BioAPI_NOTIFY_SOURCE_PRESENT event
                  notification, wait until that notification has been received, but no
                  longer than the specified maximum duration.-->
<wait_until timeout_var="sourcepresenttimeouttime"
            setvar="eventtimeoutflag">
      <or var1="nosourcepresentsupported" var2="_sourcePresent" />
</wait_until>

<!-- Issue a conformity response.
            If the condition specified in the <description> below is false, an UNDECIDED
                  conformity response is issued and the execution of the activity is
                  interrupted, otherwise a PASS conformity response is issued.-->
<assert_condition response_if_false="undecided"
            break_if_false="true">
      <description>
            Either the BSP under test does not claim support for the
                  BioAPI_NOTIFY_SOURCE_PRESENT event notification, or the event
                  notification has been received within the specified maximum duration.
      </description>
      <not var="eventtimeoutflag"/>
</assert_condition>

<!-- The BSP is ready to capture. Invoke the function BioSPI_Enroll for the purpose of
                  enrollment. -->
<invoke function="BioSPI_Enroll">
      <input name="BSPHandle" var="_bsphandle"/>
      <input name="Purpose"
                  var="__BioAPI_PURPOSE_ENROLL_FOR_VERIFICATION_ONLY"/>
      <input name="Subtype" value="0"/>
      <input name="Timeout" var="capturetimeouttime"/>
      <output name="NewTemplate" setvar="newtemplate_handle"/>
      <return setvar="return"/>
</invoke>

<!-- Issue a conformity response.
            If the condition specified in the <description> below is false, an UNDECIDED
                  conformity response is issued and the execution of the activity is
                  interrupted, otherwise a PASS conformity response is issued.-->
<assert_condition response_if_false="undecided"
            break_if_false="true">
      <description>
            The function BioSPI_Enroll has returned BioAPI_OK
      </description>
      <equal_to var1="return" var2="__BioAPI_OK"/>
</assert_condition>

<!-- Invoke the function BioSPI_FreeBIRHandle passing an invalid module handle. -->
<invoke function="BioSPI_FreeBIRHandle">
      <input name="BSPHandle" value="0"/>
      <input name="Handle" var="newtemplate_handle"/>
      <return setvar="return"/>
</invoke>

<!-- Issue a conformity response.
            If the condition specified in the <description> below is false, a FAIL
                  conformity response is issued, otherwise a PASS conformity response is
                  issued.-->
<assert_condition>
      <description>
            The function BioSPI_FreeBIRHandle has returned BioAPIERR_INVALID_BSP_HANDLE
      </description>
      <equal_to var1="return"
                  var2="__BioAPIERR_BSP_INVALID_BSP_HANDLE"/>
</assert_condition>
```

```
            <!-- Invoke the functions BioSPI_BSPDetach and BioSPI_BSPUnload -->
            <invoke activity="DetachAndUnload"
                        package="02c59458-0c46-1085-95d7-0002a5d5fd2e" >
                  <input name="bspUuid" var="bspUuid" />
                  <input name="BSPHandle" var="_bsphandle" />
            </invoke>
      </activity>

</package>
```

8.18 Assertion 5c - *BioSPI_FreeBIRHandle_InvalidBIRHandle*

Description: This assertion checks if calling BioSPI_FreeBIRHandle with an invalid BIR handle returns an error.

Excerpts

Subclause 9.3.2.1

BioAPI_RETURN BioAPI BioSPI_FreeBIRHandle

(BioAPI_HANDLE BSPHandle,

BioAPI_BIR_HANDLE Handle);

Parameters: Handle (input) - the BIR Handle to be freed.

Subclause 8.2.1

This function frees the memory and resources associated with the specified BIR Handle.

Errors: BioAPIERR_INVALID_BIR_HANDLE

References: 9.3.2.1 and 8.2.1.

Scenario:

1) Load the BSP under test

2) Attach the BSP under test

3) Enroll to obtain a BIR handle

4) Call BioSPI_FreeBIRHandle to free an invalid BIR handle.

5) Check the return code.

If any of the intermediate operations fails, an UNDECIDED conformity response is issued.

Expected results: The call to BioSPI_FreeBIRHandle returns BioAPIERR_INVALID_BIR_HANDLE.

Assertion language package

```
<package name="018e6c18-0c9c-1085-afdf-0002a5d5fd2e">
      <author>
            ISO/IEC JTC1 SC37
      </author>

      <description>
            This package contains the assertion "BioSPI_FreeBIRHandle_InvalidBIRHandle" (see the
                              "description" element of the assertion below).
      </description>
```

```
<assertion name="BioSPI_FreeBIRHandle_InvalidBIRHandle" model="BSPTesting">
      <description>
            This assertion checks if calling BioSPI_FreeBIRHandle with an invalid BIR handle
                        returns an error.
            The relevant text in BioAPI 2.0 is quoted below from subclauses 9.3.2.1, 8.2.1
            ________________________________________________________
            BioAPI_RETURN BioAPI BioSPI_FreeBIRHandle
                  (BioAPI_HANDLE BSPHandle,
                  BioAPI_BIR_HANDLE Handle);

            NOTE: Details of the function definition are located in clause 8.2.1,
                        BioAPI_FreeBIRHandle.

            ________________________________________________________
            Subclause 8.2.1:
            This function frees the memory and resources associated with the specified BIR
                        Handle.

            BioAPIERR_INVALID_BIR_HANDLE
            ________________________________________________________
            Subclause 9.3.2.1:
            Parameters: Handle (input) - the BIR Handle to be freed.
            ________________________________________________________

            In order to determine conformance with respect to the text above, the following
                        steps are performed:

                  1)    Load the BSP under test
                  2)    Attach the BSP under test
                  3)    Enroll to obtain a BIR handle
                  4)    Call BioSPI_FreeBIRHandle to free an invalid BIR handle.
                  5)    Check the return code.

            If any of the intermediate operations fails, an UNDECIDED conformity response is
                        issued.
      </description>

      <!-- UUID of the BSP under test -->
      <input name="_bspUuid"/>

      <!-- Timeout for the BioAPI_NOTIFY_INSERT event -->
      <input name="_inserttimeout"/>

      <!-- Indicates whether the BSP under test does not claim support for the
                        BioAPI_NOTIFY_SOURCE_PRESENT event notification -->
      <input name="_noSourcePresentSupported" />

      <!-- Timeout for the BioAPI_NOTIFY_SOURCE_PRESENT event -->
      <input name="_sourcepresenttimeout"/>

      <!-- Timeout for BioSPI_Enroll -->
      <input name="_capturetimeout"/>

      <!-- Invocation of the primary activity of this assertion with input parameter values
                        assigned from the assertion's parameters. -->
      <invoke activity="BioSPI_FreeBIRHandle">
            <input name="bspUuid" var="_bspUuid"/>
            <input name="inserttimeouttime" var="_inserttimeout"/>
            <input name="nosourcepresentsupported"
                        var="_noSourcePresentSupported" />
            <input name="sourcepresenttimeouttime"
                        var="_sourcepresenttimeout"/>
            <input name="capturetimeouttime" var="_capturetimeout"/>
      </invoke>

      <!-- Activity bound to a function of the framework callback interface exposed by the
                        testing component.  This activity will be automatically invoked on each
                        incoming call to the function to which it is bound. -->
      <bind activity="EventHandler"
                  package="02c59458-0c46-1085-95d7-0002a5d5fd2e"
                  function="BioSPI_EventHandler"/>
</assertion>

<activity name="BioSPI_FreeBIRHandle">
      <input name="bspUuid"/>
      <input name="inserttimeouttime"/>
```

```
<input name="nosourcepresentsupported" />
<input name="sourcepresenttimeouttime"/>
<input name="capturetimeouttime"/>

<!-- This assertion will use BSPHandle "1" for all BioSPI calls that require it -->
<set name="_bsphandle" value="1"/>

<!-- Invoke the functions BioSPI_BSPLoad and BioSPI_BSPAttach exposed by the BSP under
                  test.  -->
<invoke activity="LoadAndAttach"
            package="02c59458-0c46-1085-95d7-0002a5d5fd2e"
            break_on_break="true">
      <input name="bspUuid" var="bspUuid"/>
      <input name="bspVersion" value="32"/>
      <input name="unitIDOrNull" value="0"/>
      <input name="bspHandle" var="_bsphandle"/>
      <input name="eventtimeouttime" var="inserttimeouttime"/>
</invoke>

<set name="eventtimeoutflag" value="false"/>

<!-- If the BSP under test claims support for the BioAPI_NOTIFY_SOURCE_PRESENT event
                  notification, wait until that notification has been received, but no
                  longer than the specified maximum duration.-->
<wait_until timeout_var="sourcepresenttimeouttime"
            setvar="eventtimeoutflag">
      <or var1="nosourcepresentsupported" var2="_sourcePresent" />
</wait_until>

<!-- Issue a conformity response.
            If the condition specified in the <description> below is false, an UNDECIDED
                  conformity response is issued and the execution of the activity is
                  interrupted, otherwise a PASS conformity response is issued.-->
<assert_condition response_if_false="undecided"
            break_if_false="true">
      <description>
            Either the BSP under test does not claim support for the
                  BioAPI_NOTIFY_SOURCE_PRESENT event notification, or the event
                  notification has been received within the specified maximum duration.
      </description>
      <not var="eventtimeoutflag"/>
</assert_condition>

<!-- The BSP is ready to capture. Invoke the function BioSPI_Enroll for the purpose of
                  enrollment. -->
<invoke function="BioSPI_Enroll">
      <input name="BSPHandle" var="_bsphandle"/>
      <input name="Purpose"
                  var="__BioAPI_PURPOSE_ENROLL_FOR_VERIFICATION_ONLY"/>
      <input name="Subtype" value="0"/>
      <input name="Timeout" var="capturetimeouttime"/>
      <output name="NewTemplate" setvar="newtemplate_handle"/>
      <return setvar="return"/>
</invoke>

<!-- Issue a conformity response.
            If the condition specified in the <description> below is false, an UNDECIDED
                  conformity response is issued and the execution of the activity is
                  interrupted, otherwise a PASS conformity response is issued.-->
<assert_condition response_if_false="undecided"
            break_if_false="true">
      <description>
            The function BioSPI_Enroll has returned BioAPI_OK
      </description>
      <equal_to var1="return" var2="__BioAPI_OK"/>
</assert_condition>

<!-- Invoke the function BioSPI_FreeBIRHandle passing an invalid BIR handle. -->
<invoke function="BioSPI_FreeBIRHandle">
      <input name="BSPHandle" var="_bsphandle"/>
      <input name="Handle" value="-1"/>
      <return setvar="return"/>
</invoke>
<!-- Issue a conformity response.
```

```
                        If the condition specified in the <description> below is false, a FAIL
                              conformity response is issued, otherwise a PASS conformity response is
                              issued.-->
            <assert_condition>
                  <description>
                        The function BioSPI_FreeBIRHandle has returned BioAPIERR_INVALID_BIR_HANDLE
                  </description>
                  <equal_to var1="return" var2="__BioAPIERR_BSP_INVALID_BIR_HANDLE"/>
            </assert_condition>

            <!-- Invoke the functions BioSPI_BSPDetach and BioSPI_BSPUnload -->
            <invoke activity="DetachAndUnload"
                        package="02c59458-0c46-1085-95d7-0002a5d5fd2e" >
                  <input name="bspUuid" var="bspUuid" />
                  <input name="BSPHandle" var="_bsphandle" />
            </invoke>
      </activity>

</package>
```

8.19 Assertion 6a - *BioSPI_GetBIRFromHandle_ValidParam*

Description: This assertion checks if calling BioSPI_GetBIRFromHandle with valid parameters returns BioAPI_OK.

Excerpts

Subclause 9.3.2.2

BioAPI_RETURN BioAPI BioSPI_GetBIRFromHandle

(BioAPI_HANDLE BSPHandle,

BioAPI_BIR_HANDLE Handle,

*BioAPI_BIR *BIR);*

Subclause 8.2.2

This function returns the BIR associated with a BIR handle returned by a BSP.

A BioAPI_RETURN value indicating success or specifying a particular error condition.

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

The BIR handle is freed by the BSP before the function returns.

Subclause A.4

This function should be supported by all types of BSPs.

References: 9.3.2.2, 8.2.2, and A.4

Scenario:

1) Load the BSP under test

2) Attach the BSP under test

3) Enroll to obtain a BIR handle

4) Call BioSPI_GetBIRFromHandle. The function is expected to return BioAPI_OK.

If any of the intermediate operations fails, an UNDECIDED conformity response is issued.

Expected results: The call to BioSPI_GetBIRFromHandle returns BioAPI_OK

Assertion language package

```
<package name="0460b658-0cb4-1085-a304-0002a5d5fd2e">
      <author>
            ISO/IEC JTC1 SC37
      </author>

      <description>
            This package contains the assertion "BioSPI_GetBIRFromHandle_ValidParam" (see the
                              "description" element of the assertion below).
      </description>

      <assertion name="BioSPI_GetBIRFromHandle_ValidParam" model="BSPTesting">
            <description>
                  This assertion checks if calling BioSPI_GetBIRFromHandle with valid parameters
                              returns BioAPI_OK.
                  The relevant text in BioAPI 2.0 is quoted below from subclauses 9.3.2.2, 8.2.2 and
                              A.4.
                  ___________________________________________________________
                  BioAPI_RETURN BioAPI BioSPI_GetBIRFromHandle
                        (BioAPI_HANDLE BSPHandle,
                        BioAPI_BIR_HANDLE Handle,
                        BioAPI_BIR *BIR);

                  NOTE: Details of the function definition are located in clause 8.2.2,
                              BioAPI_GetBIRFromHandle.

                  ____________________________________________________________
                  Subclause 8.2.2:
                  This function returns the BIR associated with a BIR handle returned by a BSP.
                  A BioAPI_RETURN value indicating success or specifying a particular error
                              condition.
                  The value BioAPI_OK indicates success. All other values represent an error
                              condition.

                  The BIR handle is freed by the BSP before the function returns.
                  ____________________________________________________________
                  Subclause A.4:
                  This function should be supported by all types of BSPs.
                  ____________________________________________________________

                  In order to determine conformance with respect to the text above, the following
                              steps are performed:

                        1)    Load the BSP under test
                        2)    Attach the BSP under test
                        3)    Enroll to obtain a BIR handle
                        4)    Call BioSPI_GetBIRFromHandle.  The function is expected to return
                              BioAPI_OK.

                  If any of the intermediate operations fails, an UNDECIDED conformity response is
                              issued.
            </description>

            <!-- UUID of the BSP under test -->
            <input name="_bspUuid"/>

            <!-- Timeout for the BioAPI_NOTIFY_INSERT event -->
            <input name="_inserttimeout"/>

            <!-- Indicates whether the BSP under test does not claim support for the
                              BioAPI_NOTIFY_SOURCE_PRESENT event notification -->
            <input name="_noSourcePresentSupported" />

            <!-- Timeout for the BioAPI_NOTIFY_SOURCE_PRESENT event -->
            <input name="_sourcepresenttimeout"/>

            <!-- Timeout for BioSPI_Enroll -->
            <input name="_capturetimeout"/>
```

```
        <!-- Invocation of the primary activity of this assertion with input parameter values
                         assigned from the assertion's parameters. -->
        <invoke activity="BioSPI_GetBIRFromHandle">
              <input name="bspUuid" var="_bspUuid"/>
              <input name="inserttimeouttime" var="_inserttimeout"/>
              <input name="nosourcepresentsupported"
                         var="_noSourcePresentSupported" />
              <input name="sourcepresenttimeouttime"
                         var="_sourcepresenttimeout"/>
              <input name="capturetimeouttime" var="_capturetimeout"/>
        </invoke>

        <!-- Activity bound to a function of the framework callback interface exposed by the
                         testing component.  This activity will be automatically invoked on each
                         incoming call to the function to which it is bound. -->
        <bind activity="EventHandler"
                    package="02c59458-0c46-1085-95d7-0002a5d5fd2e"
                    function="BioSPI_EventHandler"/>
  </assertion>

  <activity name="BioSPI_GetBIRFromHandle">
        <input name="bspUuid"/>
        <input name="inserttimeouttime"/>
        <input name="nosourcepresentsupported" />
        <input name="sourcepresenttimeouttime"/>
        <input name="capturetimeouttime"/>

        <!-- This assertion will use BSPHandle "1" for all BioSPI calls that require it -->
        <set name="_bsphandle" value="1"/>

        <!-- Invoke the functions BioSPI_BSPLoad and BioSPI_BSPAttach exposed by the BSP under
                         test.-->
        <invoke activity="LoadAndAttach"
                    package="02c59458-0c46-1085-95d7-0002a5d5fd2e"
                    break_on_break="true">
              <input name="bspUuid" var="bspUuid"/>
              <input name="bspVersion" value="32"/>
              <input name="unitIDOrNull" value="0"/>
              <input name="bspHandle" var="_bsphandle"/>
              <input name="eventtimeouttime" var="inserttimeouttime"/>
        </invoke>

        <set name="eventtimeoutflag" value="false"/>

        <!-- If the BSP under test claims support for the BioAPI_NOTIFY_SOURCE_PRESENT event
                         notification, wait until that notification has been received, but no
                         longer than the specified maximum duration.-->
        <wait_until timeout_var="sourcepresenttimeouttime"
                    setvar="eventtimeoutflag">
              <or var1="nosourcepresentsupported" var2="_sourcePresent" />
        </wait_until>

        <!-- Issue a conformity response.
                    If the condition specified in the <description> below is false, an UNDECIDED
                         conformity response is issued and the execution of the activity is
                         interrupted, otherwise a PASS conformity response is issued.-->
        <assert_condition response_if_false="undecided"
                    break_if_false="true">
              <description>
                    Either the BSP under test does not claim support for the
                         BioAPI_NOTIFY_SOURCE_PRESENT event notification, or the event
                         notification has been received within the specified maximum duration.
              </description>
              <not var="eventtimeoutflag"/>
        </assert_condition>
```

```
            <!-- The BSP is ready to capture. Invoke the function BioSPI_Enroll for the purpose of
                        enrollment. -->
            <invoke function="BioSPI_Enroll">
                  <input name="BSPHandle" var="_bsphandle"/>
                  <input name="Purpose"
                        var="__BioAPI_PURPOSE_ENROLL_FOR_VERIFICATION_ONLY"/>
                  <input name="Subtype" value="0"/>
                  <input name="Timeout" var="capturetimeouttime"/>
                  <output name="NewTemplate" setvar="newtemplate_handle"/>
                  <return setvar="return"/>
            </invoke>

            <!-- Issue a conformity response.
                  If the condition specified in the <description> below is false, an UNDECIDED
                        conformity response is issued and the execution of the activity is
                        interrupted, otherwise a PASS conformity response is issued.-->
            <assert_condition response_if_false="undecided"
                  break_if_false="true">
                  <description>
                        The function BioSPI_Enroll has returned BioAPI_OK
                  </description>
                  <equal_to var1="return" var2="__BioAPI_OK"/>
            </assert_condition>

            <!-- Invoke the function BioSPI_GetBIRFromHandle passing the enrolled BIR handle. -->
            <invoke function="BioSPI_GetBIRFromHandle">
                  <input name="BSPHandle" var="_bsphandle"/>
                  <input name="Handle" var="newtemplate_handle"/>
                  <output name="HeaderVersion" setvar="headerversion"/>
                  <output name="ProcessedLevel" setvar="processedlevel"/>
                  <output name="FormatOwner" setvar="formatowner"/>
                  <output name="FormatType" setvar="formattype"/>
                  <output name="Quality" setvar="quality"/>
                  <output name="Purpose" setvar="purpose"/>
                  <output name="ProductOwner" setvar="productowner" />
                  <output name="ProductType" setvar="producttype" />
                  <output name="Creation_Year" setvar="creationyear" />
                  <output name="Creation_Month" setvar="creationmonth" />
                  <output name="Creation_Day" setvar="creationday" />
                  <output name="Creation_Hour" setvar="creationhour" />
                  <output name="Creation_Minute" setvar="creationminute" />
                  <output name="Creation_Second" setvar="creationsecond" />
                  <output name="Expiration_Year" setvar="expirationyear" />
                  <output name="Expiration_Month" setvar="expirationmonth" />
                  <output name="Expiration_Day" setvar="expirationday" />
                  <output name="SBFormatOwner" setvar="securityformatowner" />
                  <output name="SBFormatType" setvar="securityformattype" />
                  <output name="Index" setvar="index" />
                  <output name="BiometricData" setvar="biometircdata" />
                  <output name="SecurityBlock" setvar="securityblock"/>
                  <return setvar="return"/>
            </invoke>

            <!-- Issue a conformity response.
                  If the condition specified in the <description> below is false, a FAIL
                        conformity response is issued, otherwise a PASS conformity response is
                        issued.-->
            <assert_condition>
                  <description>
                        The function BioSPI_GetBIRFromHandle has returned BioAPI_OK
                  </description>
                  <equal_to var1="return" var2="__BioAPI_OK"/>
            </assert_condition>

            <!-- Invoke the functions BioSPI_BSPDetach and BioSPI_BSPUnload -->
            <invoke activity="DetachAndUnload"
                  package="02c59458-0c46-1085-95d7-0002a5d5fd2e" >
                  <input name="bspUuid" var="bspUuid" />
                  <input name="BSPHandle" var="_bsphandle" />
            </invoke>
      </activity>

</package>
```

8.20 Assertion 6b - *BioSPI_GetBIRFromHandle_InvalidBSPHandle*

Description: This assertion checks if calling BioSPI_GetBIRFromHandle with an invalid BSP handle returns an error.

Excerpts

Subclause 9.3.2.2

BioAPI_RETURN BioAPI BioSPI_GetBIRFromHandle

(BioAPI_HANDLE BSPHandle,

BioAPI_BIR_HANDLE Handle,

*BioAPI_BIR *BIR);*

Subclause 8.2.2

This function returns the BIR associated with a BIR handle returned by a BSP.

Return value: A BioAPI_RETURN value indicating success or specifying a particular error condition.

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

Parameters: BSPHandle (input) - the handle of the attached biometric service provider.

References: 9.3.2.2 and 8.2.2.

Scenario:

1) Load the BSP under test

2) Attach the BSP under test

3) Enroll to obtain a BIR handle

4) Call BioSPI_GetBIRFromHandle with an invalid BSP handle. The function is expected to return an error.

If any of the intermediate operations fails, an UNDECIDED conformity response is issued.

Expected results: The call to BioSPI_GetBIRFromHandle returns BioAPIERR_INVALID_BSP_HANDLE.

Assertion language package

```
<package name="02445668-0cc5-1085-a3ac-0002a5d5fd2e">
      <author>
            ISO/IEC JTC1 SC37
      </author>

      <description>
            This package contains the assertion "BioSPI_GetBIRFromHandle_InvalidBSPHandle" (see the
                              "description" element of the assertion below).
      </description>

      <assertion name="BioSPI_GetBIRFromHandle_InvalidBSPHandle" model="BSPTesting">
            <description>
                  This assertion checks if calling BioSPI_GetBIRFromHandle with an invalid BSP handle
                              returns an error.
                  The relevant text in BioAPI 2.0 is quoted below from subclauses 9.3.2.2, 8.2.2.
```

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

          NOTE: Details of the function definition are located in clause 8.2.2,
                      BioAPI_GetBIRFromHandle.

          ________________________________________________________
          Subclause 8.2.2:
          This function returns the BIR associated with a BIR handle returned by a BSP.

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

          Parameters: BSPHandle (input) - the handle of the attached biometric service
                      provider.
          ________________________________________________________

          In order to determine conformance with respect to the text above, the following
                      steps are performed:

                1)    Load the BSP under test
                2)    Attach the BSP under test
                3)    Enroll to obtain a BIR handle
                4)    Call BioSPI_GetBIRFromHandle with an invalid BSP handle.  The function
                      is expected to return an error.

          If any of the intermediate operations fails, an UNDECIDED conformity response is
                      issued.
    </description>

    <!-- UUID of the BSP under test -->
    <input name="_bspUuid"/>

    <!-- Timeout for the BioAPI_NOTIFY_INSERT event -->
    <input name="_inserttimeout"/>

    <!-- Indicates whether the BSP under test does not claim support for the
                      BioAPI_NOTIFY_SOURCE_PRESENT event notification -->
    <input name="_noSourcePresentSupported" />

    <!-- Timeout for the BioAPI_NOTIFY_SOURCE_PRESENT event -->
    <input name="_sourcepresenttimeout"/>

    <!-- Timeout for BioSPI_Enroll -->
    <input name="_capturetimeout"/>

    <!-- Invocation of the primary activity of this assertion with input parameter values
                      assigned from the assertion's parameters. -->
    <invoke activity="BioSPI_GetBIRFromHandle">
          <input name="bspUuid" var="_bspUuid"/>
          <input name="inserttimeouttime" var="_inserttimeout"/>
          <input name="nosourcepresentsupported"
                      var="_noSourcePresentSupported" />
          <input name="sourcepresenttimeouttime"
                      var="_sourcepresenttimeout"/>
          <input name="capturetimeouttime" var="_capturetimeout"/>
    </invoke>

    <!-- Activity bound to a function of the framework callback interface exposed by the
                      testing component.  This activity will be automatically invoked on each
                      incoming call to the function to which it is bound. -->
    <bind activity="EventHandler"
                package="02c59458-0c46-1085-95d7-0002a5d5fd2e"
                function="BioSPI_EventHandler"/>
</assertion>
```

```
<activity name="BioSPI_GetBIRFromHandle">
      <input name="bspUuid"/>
      <input name="inserttimeouttime"/>
      <input name="nosourcepresentsupported" />
      <input name="sourcepresenttimeouttime"/>
      <input name="capturetimeouttime"/>

      <!-- This assertion will use BSPHandle "1" for all BioSPI calls that require it -->
      <set name="_bsphandle" value="1"/>

      <!-- Invoke the functions BioSPI_BSPLoad and BioSPI_BSPAttach exposed by the BSP under
                  test. -->
      <invoke activity="LoadAndAttach"
            package="02c59458-0c46-1085-95d7-0002a5d5fd2e"
            break_on_break="true">
            <input name="bspUuid" var="bspUuid"/>
            <input name="bspVersion" value="32"/>
            <input name="unitIDOrNull" value="0"/>
            <input name="bspHandle" var="_bsphandle"/>
            <input name="eventtimeouttime" var="inserttimeouttime"/>
      </invoke>

      <set name="eventtimeoutflag" value="false"/>

      <!-- If the BSP under test claims support for the BioAPI_NOTIFY_SOURCE_PRESENT event
                  notification, wait until that notification has been received, but no
                  longer than the specified maximum duration.-->
      <wait_until timeout_var="sourcepresenttimeouttime"
            setvar="eventtimeoutflag">
            <or var1="nosourcepresentsupported" var2="_sourcePresent" />
      </wait_until>

      <!-- Issue a conformity response.
            If the condition specified in the <description> below is false, an UNDECIDED
                  conformity response is issued and the execution of the activity is
                  interrupted, otherwise a PASS conformity response is issued.-->
      <assert_condition response_if_false="undecided"
            break_if_false="true">
            <description>
            Either the BSP under test does not claim support for the
                  BioAPI_NOTIFY_SOURCE_PRESENT event notification, or the event
                  notification has been received within the specified maximum duration.
            </description>
            <not var="eventtimeoutflag"/>
      </assert_condition>

      <!-- The BSP is ready to capture. Invoke the function BioSPI_Enroll for the purpose of
                  enrollment. -->
      <invoke function="BioSPI_Enroll">
            <input name="BSPHandle" var="_bsphandle"/>
            <input name="Purpose"
                  var="__BioAPI_PURPOSE_ENROLL_FOR_VERIFICATION_ONLY"/>
            <input name="Subtype" value="0"/>
            <input name="Timeout" var="capturetimeouttime"/>
            <output name="NewTemplate" setvar="newtemplate_handle"/>
            <return setvar="return"/>
      </invoke>

      <!-- Issue a conformity response.
            If the condition specified in the <description> below is false, an UNDECIDED
                  conformity response is issued and the execution of the activity is
                  interrupted, otherwise a PASS conformity response is issued.-->
      <assert_condition response_if_false="undecided"
            break_if_false="true">
            <description>
            The function BioSPI_Enroll has returned BioAPI_OK
            </description>
            <equal_to var1="return" var2="__BioAPI_OK"/>
      </assert_condition>

      <!-- Invoke the function BioSPI_GetBIRFromHandle passing an invalid module handle. -->
      <invoke function="BioSPI_GetBIRFromHandle">
            <input name="BSPHandle" value="0"/>
            <input name="Handle" var="newtemplate_handle"/>
            <output name="HeaderVersion" setvar="headerversion"/>
            <output name="ProcessedLevel" setvar="processedlevel"/>
```

```
                <output name="FormatOwner" setvar="formatowner"/>
                <output name="FormatType" setvar="formattype"/>
                <output name="Quality" setvar="quality"/>
                <output name="Purpose" setvar="purpose"/>
                <output name="BiometricData" setvar="biometricdata"/>
                <return setvar="return"/>
          </invoke>

          <!-- Issue a conformity response.
                      If the condition specified in the <description> below is false, a FAIL
                            conformity response is issued, otherwise a PASS conformity response is
                            issued.-->
          <assert_condition>
                <description>
                      The function BioSPI_GetBIRFromHandle has returned
                            BioAPIERR_INVALID_BSP_HANDLE
                </description>
                <equal_to var1="return"
                            var2="__BioAPIERR_BSP_INVALID_BSP_HANDLE"/>
          </assert_condition>

          <!-- Invoke the functions BioSPI_BSPDetach and BioSPI_BSPUnload -->
          <invoke activity="DetachAndUnload"
                      package="02c59458-0c46-1085-95d7-0002a5d5fd2e" >
                <input name="bspUuid" var="bspUuid" />
                <input name="BSPHandle" var="_bsphandle" />
          </invoke>
    </activity>

</package>
```

8.21 Assertion 6c - *BioSPI_GetBIRFromHandle_InvalidBIRHandle*

Description: This assertion checks if calling the function BioSPI_GetBIRFromHandle with an invalid BIR handle returns an error.

Excerpts

Subclause 9.3.2.2

BioAPI_RETURN BioAPI BioAPI_GetBIRFromHandle

(BioAPI_HANDLE BSPHandle,

BioAPI_BIR_HANDLE Handle,

*BioAPI_BIR *BIR);*

Subclause 8.2.2

This function returns the BIR associated with a BIR handle returned by a BSP.

Parameters: Handle (input) - the handle of the BIR whose header is to be retrieved.

Return value: A BioAPI_RETURN value indicating success or specifying a particular error condition.

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

References: 9.3.2.2 and 8.2.2.

Scenario:

1) Load the BSP under test.

2) Attach the BSP under test.

3) Enroll to obtain a BIR handle.

4) Invoke the function BioSPI_GetBIRFromHandle with an invalid BIR handle, expecting an error.

If any of the intermediate operations fails, an UNDECIDED conformity response is issued.

Expected results: The call to BioSPI_GetBIRFromHandle returns BioAPIERR_INVALID_BIR_HANDLE.

Assertion language package

```
<package name="0194a9c0-0cc7-1085-8780-0002a5d5fd2e">
      <author>
            ISO/IEC JTC1 SC37
      </author>

      <description>
            This package contains the assertion "BioSPI_GetBIRFromHandle_InvalidBIRHandle" (see the
                              "description" element of the assertion below).
      </description>

      <assertion name="BioSPI_GetBIRFromHandle_InvalidBIRHandle" model="BSPTesting">
            <description>
                  This assertion checks if calling the function BioSPI_GetBIRFromHandle with an
                              invalid BIR handle returns an error.
                  The relevant text in BioAPI 2.0 is quoted below from subclauses 9.3.2.2 and 8.2.2.
                  ________________________________________________________
                  BioAPI_RETURN BioAPI BioSPI_GetBIRFromHandle
                        (BioAPI_HANDLE BSPHandle,
                        BioAPI_BIR_HANDLE Handle,
                        BioAPI_BIR *BIR);

                  NOTE: Details of the function definition are located in clause 8.2.2,
                              BioAPI_GetBIRFromHandle.

                  ________________________________________________________
                  Subclause 8.2.2:
                  This function returns the BIR associated with a BIR handle returned by a BSP.

                  Parameters: Handle (input) - the handle of the BIR whose header is to be retrieved.
                  Return value: A BioAPI_RETURN value indicating success or specifying a particular
                              error condition.
                  The value BioAPI_OK indicates success. All other values represent an error
                              condition.
                  ________________________________________________________

                  In order to determine conformance with respect to the text above, the following
                              steps are performed:

                        1)    Load the BSP under test.
                        2)    Attach the BSP under test.
                        3)    Enroll to obtain a BIR handle.
                        4)    Invoke the function BioSPI_GetBIRFromHandle with an invalid BIR handle,
                              expecting an error.

                  If any of the intermediate operations fails, an UNDECIDED conformity response is
                              issued.
            </description>

            <!-- UUID of the BSP under test -->
            <input name="_bspUuid"/>

            <!-- Timeout for the BioAPI_NOTIFY_INSERT event -->
            <input name="_inserttimeout"/>

            <!-- Indicates whether the BSP under test does not claim support for the
                              BioAPI_NOTIFY_SOURCE_PRESENT event notification -->
            <input name="_noSourcePresentSupported"/>

            <!-- Timeout for the BioAPI_NOTIFY_SOURCE_PRESENT event -->
            <input name="_sourcepresenttimeout"/>

            <!-- Timeout for BioSPI_Enroll -->
            <input name="_capturetimeout"/>
```

```
		<!-- Invocation of the primary activity of this assertion with input parameter values
				assigned from the assertion's parameters. -->
		<invoke activity="BioSPI_GetBIRFromHandle">
			<input name="bspUuid" var="_bspUuid"/>
			<input name="inserttimeouttime" var="_inserttimeout"/>
			<input name="nosourcepresentsupported"
				var="_noSourcePresentSupported"/>
			<input name="sourcepresenttimeouttime"
				var="_sourcepresenttimeout"/>
			<input name="capturetimeouttime" var="_capturetimeout"/>
		</invoke>

		<!-- Activity bound to a function of the framework callback interface exposed by the
				testing component.  This activity will be automatically invoked on each
				incoming call to the function to which it is bound. -->
		<bind activity="EventHandler"
			package="02c59458-0c46-1085-95d7-0002a5d5fd2e"
			function="BioSPI_EventHandler"/>
	</assertion>

	<activity name="BioSPI_GetBIRFromHandle">
		<input name="bspUuid"/>
		<input name="inserttimeouttime"/>
		<input name="nosourcepresentsupported"/>
		<input name="sourcepresenttimeouttime"/>
		<input name="capturetimeouttime"/>

		<!-- This assertion will use BSPHandle "1" for all BioSPI calls that require it -->
		<set name="_bsphandle" value="1"/>

		<!-- Invoke the functions BioSPI_BSPLoad and BioSPI_BSPAttach exposed by the BSP under
				test. -->
		<invoke activity="LoadAndAttach"
			package="02c59458-0c46-1085-95d7-0002a5d5fd2e"
			break_on_break="true">
		<input name="bspUuid" var="bspUuid"/>
		<input name="bspVersion" value="32"/>
		<input name="unitIDOrNull" value="0"/>
		<input name="bspHandle" var="_bsphandle"/>
		<input name="eventtimeouttime" var="inserttimeouttime"/>
	</invoke>

	<set name="eventtimeoutflag" value="false"/>
		<!-- If the BSP under test claims support for the BioAPI_NOTIFY_SOURCE_PRESENT event
				notification, wait until that notification has been received, but no
				longer than the specified maximum duration.-->
		<wait_until timeout_var="sourcepresenttimeouttime"
			setvar="eventtimeoutflag">
			<or var1="nosourcepresentsupported" var2="_sourcePresent"/>
		</wait_until>

		<!-- Issue a conformity response.
			If the condition specified in the <description> below is false, an UNDECIDED
				conformity response is issued and the execution of the activity is
				interrupted, otherwise a PASS conformity response is issued.-->
		<assert_condition response_if_false="undecided"
			break_if_false="true">
			<description>
				Either the BSP under test does not claim support for the
					BioAPI_NOTIFY_SOURCE_PRESENT event notification, or the event
					notification has been received within the specified maximum duration.
			</description>
			<not var="eventtimeoutflag"/>
		</assert_condition>

		<!-- The BSP is ready to capture. Invoke the function BioSPI_Enroll for the purpose of
				enrollment. -->
		<invoke function="BioSPI_Enroll">
			<input name="BSPHandle" var="_bsphandle"/>
			<input name="Purpose"
				var="__BioAPI_PURPOSE_ENROLL_FOR_VERIFICATION_ONLY"/>
			<input name="Subtype" value="0"/>
			<input name="Timeout" var="capturetimeouttime"/>
			<output name="NewTemplate" setvar="newtemplate_handle"/>
			<return setvar="return"/>
		</invoke>
```

```
            <!-- Issue a conformity response.
                        If the condition specified in the <description> below is false, an UNDECIDED
                              conformity response is issued and the execution of the activity is
                              interrupted, otherwise a PASS conformity response is issued.-->
            <assert_condition response_if_false="undecided"
                        break_if_false="true">
                  <description>
                        The function BioSPI_Enroll has returned BioAPI_OK
                  </description>
                  <equal_to var1="return" var2="__BioAPI_OK"/>
            </assert_condition>

            <!-- Invoke the function BioSPI_GetBIRFromHandle passing an invalid BIR handle. -->
            <invoke function="BioSPI_GetBIRFromHandle">
                  <input name="BSPHandle" var="_bsphandle"/>
                  <input name="Handle" value="-1"/>
                  <output name="HeaderVersion" setvar="headerversion"/>
                  <output name="ProcessedLevel" setvar="processedLevel"/>
                  <output name="FormatOwner" setvar="formatowner"/>
                  <output name="FormatType" setvar="formattype"/>
                  <output name="Quality" setvar="quality"/>
                  <output name="Purpose" setvar="purpose"/>
                  <output name="BiometricData" setvar="biometricdata"/>
                  <return setvar="return"/>
            </invoke>

            <!-- Issue a conformity response.
                        If the condition specified in the <description> below is false, a FAIL
                              conformity response is issued, otherwise a PASS conformity response is
                              issued.-->
            <assert_condition>
                  <description>
                        The function BioSPI_GetBIRFromHandle has returned
                              BioAPIERR_INVALID_BIR_HANDLE.
                  </description>
                  <equal_to var1="return" var2="__BioAPIERR_BSP_INVALID_BIR_HANDLE"/>
            </assert_condition>

            <!-- Invoke the functions BioSPI_BSPDetach and BioSPI_BSPUnload -->
            <invoke activity="DetachAndUnload"
                        package="02c59458-0c46-1085-95d7-0002a5d5fd2e" >
                  <input name="bspUuid" var="bspUuid"/>
                  <input name="BSPHandle" var="_bsphandle"/>
            </invoke>
      </activity>

</package>
```

8.22 Assertion 7a - *BioSPI_GetHeaderFromHandle_ValidParam*

Description: This assertion checks if a call to the function BioSPI_GetHeaderFromHandle with valid input parameters returns BioAPI_OK.

Excerpts

Subclause 9.3.2.3

BioAPI_RETURN BioAPI BioSPI_GetHeaderFromHandle

(BioAPI_HANDLE BSPHandle,

BioAPI_BIR_HANDLE Handle,

*BioAPI_BIR_HEADER *Header);*

Subclause 8.2.3

Retrieves the BIR header identified by Handle.

Subclause A.4

This function should be supported by all type of BSPs.

References: 9.3.2.3, 8.2.3, and A.4.

Scenario:

1) Load the BSP under test.

2) Attach the BSP under test.

3) Call BioSPI_Enroll.

4) Call BioSPI_GetHeaderFromHandle.

5) Check the return code of the call.

6) Detach and unload the BSP.

If any of the intermediate operations fails, an UNDECIDED conformity response is issued.

Expected results: The call to BioSPI_GetHeaderFromHandle returns BioAPI_OK

Assertion language package

```
<package name="027a7db0-0cc7-1085-9391-0002a5d5fd2e">
      <author>
            ISO/IEC JTC1 SC37
      </author>

      <description>
            This package contains the assertion "BioSPI_GetHeaderFromHandle_ValidParam" (see the
                              "description" element of the assertion below).
      </description>

      <assertion name="BioSPI_GetHeaderFromHandle_ValidParam" model="BSPTesting">
            <description>
                  This assertion checks if a call to the function BioSPI_GetHeaderFromHandle with
                              valid input parameters returns BioAPI_OK.
                  The relevant text in BioAPI 2.0 is quoted below from subclauses 9.3.2.3, 8.2.3 and
                              A.4.
                  ___________________________________________________________
                  BioAPI_RETURN BioAPI BioSPI_GetHeaderFromHandle
                        (BioAPI_HANDLE BSPHandle,
                        BioAPI_BIR_HANDLE Handle,
                        BioAPI_BIR_HEADER *Header);

                  NOTE: Details of the function definition are located in clause 8.2.3,
                              BioAPI_GetHeaderFromHandle.

                  ___________________________________________________________
                  Subclause 8.2.3:
                  Retrieves the BIR header identified by Handle.

                  Subclause A.4:
                  This function should be supported by all type of BSPs.
                  ___________________________________________________________

                  In order to determine conformance with respect to the text above, the following
                              steps are performed:

                        1)    Load the BSP under test.
                        2)    Attach the BSP under test.
                        3)    Call BioSPI_Enroll.
                        4)    Call BioSPI_GetHeaderFromHandle.
                        5)    Check the return code of the call.
                        6)    Detach and unload the BSP.
            If any of the intermediate operations fails, an UNDECIDED conformity response is
                        issued.
      </description>

      <!-- UUID of the BSP under test -->
      <input name="_bspUuid"/>

      <!-- Timeout for the BioAPI_NOTIFY_INSERT event -->
      <input name="_inserttimeout"/>

      <!-- Indicates whether the BSP under test does not claim support for the
                        BioAPI_NOTIFY_SOURCE_PRESENT event notification -->
      <input name="_noSourcePresentSupported" />

      <!-- Timeout for the BioAPI_NOTIFY_SOURCE_PRESENT event -->
      <input name="_sourcepresenttimeout"/>

      <!-- Timeout for BioSPI_Enroll -->
      <input name="_capturetimeout"/>

      <!-- Invocation of the primary activity of this assertion with input parameter values
                        assigned from the assertion's parameters. -->
      <invoke activity="BioSPI_GetHeaderFromHandle">
            <input name="bspUuid" var="_bspUuid"/>
            <input name="inserttimeouttime" var="_inserttimeout"/>
            <input name="nosourcepresentsupported"
                        var="_noSourcePresentSupported" />
            <input name="sourcepresenttimeouttime"
                        var="_sourcepresenttimeout"/>
            <input name="capturetimeouttime" var="_capturetimeout"/>
      </invoke>

      <!-- Activity bound to a function of the framework callback interface exposed by the
                        testing component.  This activity will be automatically invoked on each
                        incoming call to the function to which it is bound. -->
      <bind activity="EventHandler"
                  package="02c59458-0c46-1085-95d7-0002a5d5fd2e"
                  function="BioSPI_EventHandler"/>
</assertion>

<activity name="BioSPI_GetHeaderFromHandle">
      <input name="bspUuid"/>
      <input name="inserttimeouttime"/>
      <input name="nosourcepresentsupported" />
      <input name="sourcepresenttimeouttime"/>
      <input name="capturetimeouttime"/>

      <!-- This assertion will use BSPHandle "1" for all BioSPI calls that require it -->
      <set name="_bsphandle" value="1"/>

      <!-- Invoke the functions BioSPI_BSPLoad and BioSPI_BSPAttach exposed by the BSP under
                        test.
                  The input value for the parameter "unitIDOrNull" is "0", therefore the
                        assertion will test a sensor unit chosen by the BSP. -->
      <invoke activity="LoadAndAttach"
                  package="02c59458-0c46-1085-95d7-0002a5d5fd2e"
                  break_on_break="true">
            <input name="bspUuid" var="bspUuid"/>
            <input name="bspVersion" value="32"/>
            <input name="unitIDOrNull" value="0"/>
            <input name="bspHandle" var="_bsphandle"/>
            <input name="eventtimeouttime" var="inserttimeouttime"/>
      </invoke>

      <set name="eventtimeoutflag" value="false"/>

      <!-- If the BSP under test claims support for the BioAPI_NOTIFY_SOURCE_PRESENT event
                        notification, wait until that notification has been received, but no
                        longer than the specified maximum duration.-->
      <wait_until timeout_var="sourcepresenttimeouttime"
                  setvar="eventtimeoutflag">
            <or var1="nosourcepresentsupported" var2="_sourcePresent" />
      </wait_until>
```

```
<!-- Issue a conformity response.
            If the condition specified in the <description> below is false, an UNDECIDED
                  conformity response is issued and the execution of the activity is
                  interrupted, otherwise a PASS conformity response is issued.-->
<assert_condition response_if_false="undecided"
            break_if_false="true">
      <description>
            Either the BSP under test does not claim support for the
                  BioAPI_NOTIFY_SOURCE_PRESENT event notification, or the event
                  notification has been received within the specified maximum duration.
      </description>
      <not var="eventtimeoutflag"/>
</assert_condition>

<!-- The BSP is ready to capture. Invoke the function BioSPI_Enroll for the purpose of
                  enrollment. -->
<invoke function="BioSPI_Enroll">
      <input name="BSPHandle" var="_bsphandle"/>
      <input name="Purpose"
                  var="__BioAPI_PURPOSE_ENROLL_FOR_VERIFICATION_ONLY"/>
      <input name="Subtype" value="0"/>
      <input name="Timeout" var="capturetimeouttime"/>
      <output name="NewTemplate" setvar="newtemplate_handle"/>
      <return setvar="return"/>
</invoke>

<!-- Issue a conformity response.
            If the condition specified in the <description> below is false, an UNDECIDED
                  conformity response is issued and the execution of the activity is
                  interrupted, otherwise a PASS conformity response is issued.-->
<assert_condition response_if_false="undecided"
            break_if_false="true">
      <description>
            The function BioSPI_Enroll has returned BioAPI_OK
      </description>
      <equal_to var1="return" var2="__BioAPI_OK"/>
</assert_condition>

<!-- Invoke the function BioSPI_GetHeaderFromHandle. -->
<invoke function="BioSPI_GetHeaderFromHandle">
      <input name="BSPHandle" var="_bsphandle"/>
      <input name="Handle" var="newtemplate_handle"/>
      <output name="HeaderVersion" setvar="headerversion"/>
      <output name="ProcessedLevel" setvar="processedLevel"/>
      <output name="FormatOwner" setvar="formatowner"/>
      <output name="FormatType" setvar="formattype"/>
      <output name="Quality" setvar="quality"/>
      <output name="Purpose" setvar="purpose"/>
      <output name="ProductOwner" setvar="productowner" />
      <output name="ProductType" setvar="producttype" />
      <output name="Creation_Year" setvar="creationyear" />
      <output name="Creation_Month" setvar="creationmonth" />
      <output name="Creation_Day" setvar="creationday" />
      <output name="Creation_Hour" setvar="creationhour" />
      <output name="Creation_Minute" setvar="creationminute" />
      <output name="Creation_Second" setvar="creationsecond" />
      <output name="Expiration_Year" setvar="expirationyear" />
      <output name="Expiration_Month" setvar="expirationmonth" />
      <output name="Expiration_Day" setvar="expirationday" />
      <output name="SBFormatOwner" setvar="securityformatowner" />
      <output name="SBFormatType" setvar="securityformattype" />
      <output name="Index" setvar="index" />
      <return setvar="return"/>
</invoke>
```

```
            <!-- Issue a conformity response.
                        If the condition specified in the <description> below is false, a FAIL
                              conformity response is issued, otherwise a PASS conformity response is
                              issued.-->
            <assert_condition>
                  <description>
                        The function BioSPI_GetHeaderFromHandle has returned BioAPI_OK
                  </description>
                  <equal_to var1="return" var2="__BioAPI_OK"/>
            </assert_condition>

            <!-- Invoke the functions BioSPI_BSPDetach and BioSPI_BSPUnload -->
            <invoke activity="DetachAndUnload"
                        package="02c59458-0c46-1085-95d7-0002a5d5fd2e" >
                  <input name="bspUuid" var="bspUuid" />
                  <input name="BSPHandle" var="_bsphandle" />
            </invoke>
      </activity>

</package>
```

8.23 Assertion 7b - *BioSPI_GetHeaderFromHandle_InvalidBSPHandle*

Description: This assertion checks if invoking the function BioSPI_GetHeaderFromHandle with an invalid module handle returns an error.

Excerpts

Subclause 9.3.2.3

BioAPI_RETURN BioAPI BioSPI_GetHeaderFromHandle

(BioAPI_HANDLE BSPHandle,

BioAPI_BIR_HANDLE Handle,

*BioAPI_BIR_HEADER *Header);*

Subclause 8.2.3

Retrieves the BIR header identified by Handle.

Parameters: BSPHandle (input) - the handle of the attached biometric service provider.

References: 9.3.2.3 and 8.2.3.

Scenario:

1) Load the BSP under test.

2) Attach the BSP under test.

3) Call the function BioSPI_Enroll.

4) Call BioSPI_GetHeaderFromHandle using an invalid BSP handle.

5) Check the return code of the call.

6) Detach and unload the BSP under test.

If any of the intermediate operations fails, an UNDECIDED conformity response is issued.

Expected results: The call to BioSPI_GetHeaderFromHandle returns BioAPIERR_INVALID_BSP_HANDLE.

Assertion language package

```
<package name="057e0d38-0ccd-1085-83b8-0002a5d5fd2e">
      <author>
            ISO/IEC JTC1 SC37
      </author>

      <description>
            This package contains the assertion "BioSPI_GetHeaderFromHandle_InvalidBSPHandle" (see
                              the "description" element of the assertion below).
      </description>

      <assertion name="BioSPI_GetHeaderFromHandle_InvalidBSPHandle" model="BSPTesting">
            <description>
                  This assertion checks if invoking the function BioSPI_GetHeaderFromHandle with an
                              invalid module handle returns an error.
                  The relevant text in BioAPI 2.0 is quoted below from subclauses 9.3.2.3 and 8.2.3
                  ___________________________________________________________
                  BioAPI_RETURN BioAPI BioSPI_GetHeaderFromHandle
                        (BioAPI_HANDLE BSPHandle,
                        BioAPI_BIR_HANDLE Handle,
                        BioAPI_BIR_HEADER *Header);

                  NOTE: Details of the function definition are located in clause 8.2.3,
                              BioAPI_GetHeaderFromHandle.

                  ___________________________________________________________
                  Subclause 8.2.3:
                  Retrieves the BIR header identified by Handle.

                  Parameters: BSPHandle (input) - the handle of the attached biometric service
                              provider.
                  ___________________________________________________________

                  In order to determine conformance with respect to the text above, the following
                              steps are performed:

                        1)    Load the BSP under test.
                        2)    Attach the BSP under test.
                        3)    Call the function BioSPI_Enroll.
                        4)    Call BioSPI_GetHeaderFromHandle using an invalid BSP handle.
                        5)    Check the return code of the call.
                        6)    Detach and unload the BSP under test.

                  If any of the intermediate operations fails, an UNDECIDED conformity response is
                              issued.
            </description>

            <!-- UUID of the BSP under test -->
            <input name="_bspUuid"/>

            <!-- Timeout for the BioAPI_NOTIFY_INSERT event -->
            <input name="_inserttimeout"/>

            <!-- Indicates whether the BSP under test does not claim support for the
                              BioAPI_NOTIFY_SOURCE_PRESENT event notification -->
            <input name="_noSourcePresentSupported"/>

            <!-- Timeout for the BioAPI_NOTIFY_SOURCE_PRESENT event -->
            <input name="_sourcepresenttimeout"/>

            <!-- Timeout for BioSPI_Enroll -->
            <input name="_capturetimeout"/>

            <!-- Invocation of the primary activity of this assertion with input parameter values
                              assigned from the assertion's parameters. -->
            <invoke activity="BioSPI_GetHeaderFromHandle">
                  <input name="bspUuid" var="_bspUuid"/>
                  <input name="inserttimeouttime" var="_inserttimeout"/>
                  <input name="nosourcepresentsupported"
                              var="_noSourcePresentSupported"/>
                  <input name="sourcepresenttimeouttime"
                              var="_sourcepresenttimeout"/>
            <input name="capturetimeouttime" var="_capturetimeout"/>
        </invoke>

        <!-- Activity bound to a function of the framework callback interface exposed by the
                        testing component.  This activity will be automatically invoked on each
                        incoming call to the function to which it is bound. -->
        <bind activity="EventHandler"
                    package="02c59458-0c46-1085-95d7-0002a5d5fd2e"
                    function="BioSPI_EventHandler"/>
    </assertion>

    <activity name="BioSPI_GetHeaderFromHandle">
        <input name="bspUuid"/>
        <input name="inserttimeouttime"/>
        <input name="nosourcepresentsupported"/>
        <input name="sourcepresenttimeouttime"/>
        <input name="capturetimeouttime"/>

        <!-- This assertion will use BSPHandle "1" for all BioSPI calls that require it -->
        <set name="_bsphandle" value="1"/>

        <!-- Invoke the functions BioSPI_BSPLoad and BioSPI_BSPAttach exposed by the BSP under
                        test.
                    The input value for the parameter "unitIDOrNull" is "0", therefore the
                        assertion will test a sensor unit chosen by the BSP. -->
        <invoke activity="LoadAndAttach"
                    package="02c59458-0c46-1085-95d7-0002a5d5fd2e"
                    break_on_break="true">
            <input name="bspUuid" var="bspUuid"/>
            <input name="bspVersion" value="32"/>
            <input name="unitIDOrNull" value="0"/>
            <input name="bspHandle" var="_bsphandle"/>
            <input name="eventtimeouttime" var="inserttimeouttime"/>
        </invoke>

        <set name="eventtimeoutflag" value="false"/>

        <!-- If the BSP under test claims support for the BioAPI_NOTIFY_SOURCE_PRESENT event
                        notification, wait until that notification has been received, but no
                        longer than the specified maximum duration.-->
        <wait_until timeout_var="sourcepresenttimeouttime"
                    setvar="eventtimeoutflag">
            <or var1="nosourcepresentsupported" var2="_sourcePresent"/>
        </wait_until>

        <!-- Issue a conformity response.
                    If the condition specified in the <description> below is false, an UNDECIDED
                        conformity response is issued and the execution of the activity is
                        interrupted, otherwise a PASS conformity response is issued.-->
        <assert_condition response_if_false="undecided"
                    break_if_false="true">
            <description>
                    Either the BSP under test does not claim support for the
                        BioAPI_NOTIFY_SOURCE_PRESENT event notification, or the event
                        notification has been received within the specified maximum duration.
            </description>
            <not var="eventtimeoutflag"/>
        </assert_condition>

        <!-- The BSP is ready to capture. Invoke the function BioSPI_Enroll for the purpose of
                        enrollment. -->
        <invoke function="BioSPI_Enroll">
            <input name="BSPHandle" var="_bsphandle"/>
            <input name="Purpose"
                        var="__BioAPI_PURPOSE_ENROLL_FOR_VERIFICATION_ONLY"/>
            <input name="Subtype" value="0"/>
            <input name="Timeout" var="capturetimeouttime"/>
            <output name="NewTemplate" setvar="newtemplate_handle"/>
            <return setvar="return"/>
        </invoke>

        <!-- Issue a conformity response.
                    If the condition specified in the <description> below is false, an UNDECIDED
                        conformity response is issued and the execution of the activity is
                        interrupted, otherwise a PASS conformity response is issued.-->
        <assert_condition response_if_false="undecided"
```

```
                    break_if_false="true">
              <description>
                    The function BioSPI_Enroll has returned BioAPI_OK
              </description>
              <equal_to var1="return" var2="__BioAPI_OK"/>
        </assert_condition>

        <!-- Invoke the function BioSPI_GetHeaderFromHandle passing an invalid module handle. -->
        <invoke function="BioSPI_GetHeaderFromHandle">
              <input name="BSPHandle" value="0"/>
              <input name="Handle" var="newtemplate_handle"/>
              <output name="HeaderVersion" setvar="headerversion"/>
              <output name="ProcessedLevel" setvar="processedLevel"/>
              <output name="FormatOwner" setvar="formatowner"/>
              <output name="FormatType" setvar="formattype"/>
              <output name="Quality" setvar="quality"/>
              <output name="Purpose" setvar="purpose"/>
              <return setvar="return"/>
        </invoke>

        <!-- Issue a conformity response.
                    If the condition specified in the <description> below is false, a FAIL
                          conformity response is issued, otherwise a PASS conformity response is
                          issued.-->
        <assert_condition>
              <description>
                    The function BioSPI_GetHeaderFromHandle has returned
                          BioAPIERR_INVALID_BSP_HANDLE
              </description>
              <equal_to var1="return"
                          var2="__BioAPIERR_BSP_INVALID_BSP_HANDLE"/>
        </assert_condition>

        <!-- Invoke the functions BioSPI_BSPDetach and BioSPI_BSPUnload -->
        <invoke activity="DetachAndUnload"
                    package="02c59458-0c46-1085-95d7-0002a5d5fd2e" >
              <input name="bspUuid" var="bspUuid"/>
              <input name="BSPHandle" var="_bsphandle"/>
        </invoke>
  </activity>

</package>
```

8.24 Assertion 7c - *BioSPI_GetHeaderFromHandle_InvalidBIRHandle*

Description: This assertion checks if a call to the function BioSPI_GetHeaderFromHandle with an invalid BIR handle returns an error.

Excerpts

Subclause 9.3.2.3

BioAPI_RETURN BioAPI BioSPI_GetHeaderFromHandle

(BioAPI_HANDLE BSPHandle,

BioAPI_BIR_HANDLE Handle,

*BioAPI_BIR_HEADER *Header);*

Subclause 8.2.3

Retrieves the BIR header identified by Handle.

Parameters: Handle (input) - the handle of the BIR whose header is to be retrieved.

Errors: BioAPIERR_INVALID_BIR_HANDLE

References: 9.3.2.3 and 8.2.3.

Scenario:

1) Load the BSP under test.

2) Attach the BSP under test.

3) Call BioSPI_Enroll.

4) Call BioSPI_GetHeaderFromHandle with an invalid BIR handle.

5) Check the return code of the call.

6) Detach and unload the BSP.

If any of the intermediate operations fails, an UNDECIDED conformity response is issued.

Expected results: The call to BioSPI_GetHeaderFromHandle returns BioAPIERR_INVALID_BIR_HANDLE.

Assertion language package

```
<package name="02195e68-0cce-1085-a46f-0002a5d5fd2e">
      <author>
            ISO/IEC JTC1 SC37
      </author>

      <description>
            This package contains the assertion "BioSPI_GetHeaderFromHandle_InvalidBIRHandle" (see
                        the "description" element of the assertion below).
      </description>

      <assertion name="BioSPI_GetHeaderFromHandle_InvalidBIRHandle" model="BSPTesting">
            <description>
                  This assertion checks if a call to the function BioSPI_GetHeaderFromHandle with an
                        invalid BIR handle returns an error.
                  The relevant text in BioAPI 2.0 is quoted below from subclauses 9.3.2.3 and 8.2.3.
                  _______________________________________________________
                  BioAPI_RETURN BioAPI BioSPI_GetHeaderFromHandle
                        (BioAPI_HANDLE BSPHandle,
                        BioAPI_BIR_HANDLE Handle,
                        BioAPI_BIR_HEADER *Header);

                  NOTE: Details of the function definition are located in clause 8.2.3,
                        BioAPI_GetHeaderFromHandle.

                  ________________________________________________________
                  Subclause 8.2.3:
                  Retrieves the BIR header identified by Handle.

                  Parameters: Handle (input) - the handle of the BIR whose header is to be retrieved.

                  BioAPIERR_INVALID_BIR_HANDLE
                  ________________________________________________________

                  In order to determine conformance with respect to the text above, the following
                        steps are performed:

                        1)    Load the BSP under test.
                        2)    Attach the BSP under test.
                        3)    Call BioSPI_Enroll.
                        4)    Call BioSPI_GetHeaderFromHandle with an invalid BIR handle.
                        5)    Check the return code of the call.
                        6)    Detach and unload the BSP.

                  If any of the intermediate operations fails, an UNDECIDED conformity response is
                        issued.
            </description>

            <!-- UUID of the BSP under test -->
```

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

        <!-- Timeout for the BioAPI_NOTIFY_INSERT event -->
        <input name="_inserttimeout"/>

        <!-- Indicates whether the BSP under test does not claim support for the
                         BioAPI_NOTIFY_SOURCE_PRESENT event notification -->
        <input name="_noSourcePresentSupported" />

        <!-- Timeout for the BioAPI_NOTIFY_SOURCE_PRESENT event -->
        <input name="_sourcepresenttimeout"/>

        <!-- Timeout for BioSPI_Enroll -->
        <input name="_capturetimeout"/>

        <!-- Invocation of the primary activity of this assertion with input parameter values
                         assigned from the assertion's parameters. -->
        <invoke activity="BioSPI_GetHeaderFromHandle">
              <input name="bspUuid" var="_bspUuid"/>
              <input name="inserttimeouttime" var="_inserttimeout"/>
              <input name="nosourcepresentsupported"
                         var="_noSourcePresentSupported" />
              <input name="sourcepresenttimeouttime"
                         var="_sourcepresenttimeout"/>
              <input name="capturetimeouttime" var="_capturetimeout"/>
        </invoke>

        <!-- Activity bound to a function of the framework callback interface exposed by the
                         testing component.  This activity will be automatically invoked on each
                         incoming call to the function to which it is bound. -->
        <bind activity="EventHandler"
                   package="02c59458-0c46-1085-95d7-0002a5d5fd2e"
                   function="BioSPI_EventHandler"/>
  </assertion>

  <activity name="BioSPI_GetHeaderFromHandle">
        <input name="bspUuid"/>
        <input name="inserttimeouttime"/>
        <input name="nosourcepresentsupported" />
        <input name="sourcepresenttimeouttime"/>
        <input name="capturetimeouttime"/>

        <!-- This assertion will use BSPHandle "1" for all BioSPI calls that require it -->
        <set name="_bsphandle" value="1"/>

        <!-- Invoke the functions BioSPI_BSPLoad and BioSPI_BSPAttach exposed by the BSP under
                         test.
                   The input value for the parameter "unitIDOrNull" is "0", therefore the
                         assertion will test a sensor unit chosen by the BSP. -->
        <invoke activity="LoadAndAttach"
                   package="02c59458-0c46-1085-95d7-0002a5d5fd2e"
                   break_on_break="true">
              <input name="bspUuid" var="bspUuid"/>
              <input name="bspVersion" value="32"/>
              <input name="unitIDOrNull" value="0"/>
              <input name="bspHandle" var="_bsphandle"/>
              <input name="eventtimeouttime" var="inserttimeouttime"/>
        </invoke>

        <set name="eventtimeoutflag" value="false"/>

        <!-- If the BSP under test claims support for the BioAPI_NOTIFY_SOURCE_PRESENT event
                         notification, wait until that notification has been received, but no
                         longer than the specified maximum duration.-->
        <wait_until timeout_var="sourcepresenttimeouttime"
                   setvar="eventtimeoutflag">
              <or var1="nosourcepresentsupported" var2="_sourcePresent" />
        </wait_until>

        <!-- Issue a conformity response.
                   If the condition specified in the <description> below is false, an UNDECIDED
                         conformity response is issued and the execution of the activity is
                         interrupted, otherwise a PASS conformity response is issued.-->
        <assert_condition response_if_false="undecided"
                   break_if_false="true">
              <description>
```

```
                Either the BSP under test does not claim support for the
                      BioAPI_NOTIFY_SOURCE_PRESENT event notification, or the event
                      notification has been received within the specified maximum duration.
          </description>
          <not var="eventtimeoutflag"/>
    </assert_condition>

    <!-- The BSP is ready to capture. Invoke the function BioSPI_Enroll for the purpose of
                      enrollment. -->
    <invoke function="BioSPI_Enroll">
          <input name="BSPHandle" var="_bsphandle"/>
          <input name="Purpose"
                      var="__BioAPI_PURPOSE_ENROLL_FOR_VERIFICATION_ONLY"/>
          <input name="Subtype" value="0"/>
          <input name="Timeout" var="capturetimeouttime"/>
          <output name="NewTemplate" setvar="newtemplate_handle"/>
          <return setvar="return"/>
    </invoke>

    <!-- Issue a conformity response.
                If the condition specified in the <description> below is false, an UNDECIDED
                      conformity response is issued and the execution of the activity is
                      interrupted, otherwise a PASS conformity response is issued.-->
    <assert_condition response_if_false="undecided"
                break_if_false="true">
          <description>
                The function BioSPI_Enroll has returned BioAPI_OK
          </description>
          <equal_to var1="return" var2="__BioAPI_OK"/>
    </assert_condition>

    <!-- Invoke the function BioSPI_GetHeaderFromHandle. -->
    <invoke function="BioSPI_GetHeaderFromHandle">
          <input name="BSPHandle" var="_bsphandle"/>
          <input name="Handle" value="-1" />
          <output name="HeaderVersion" setvar="headerversion"/>
          <output name="ProcessedLevel" setvar="processedLevel"/>
          <output name="FormatOwner" setvar="formatowner"/>
          <output name="FormatType" setvar="formattype"/>
          <output name="Quality" setvar="quality"/>
          <output name="Purpose" setvar="purpose"/>
          <return setvar="return"/>
    </invoke>

    <!-- Issue a conformity response.
                If the condition specified in the <description> below is false, a FAIL
                      conformity response is issued, otherwise a PASS conformity response is
                      issued.-->
    <assert_condition>
          <description>
                The function BioSPI_GetHeaderFromHandle has returned
                      BioAPIERR_INVALID_BIR_HANDLE
          </description>
          <equal_to var1="return" var2="__BioAPIERR_BSP_INVALID_BIR_HANDLE"/>
    </assert_condition>

    <!-- Invoke the functions BioSPI_BSPDetach and BioSPI_BSPUnload -->
    <invoke activity="DetachAndUnload"
                package="02c59458-0c46-1085-95d7-0002a5d5fd2e" >
          <input name="bspUuid" var="bspUuid" />
          <input name="BSPHandle" var="_bsphandle" />
    </invoke>
</activity>

</package>
```

8.25 Assertion 7d - *BioSPI_GetHeaderFromHandle_BIRHandleNotFreed*

Description: This assertion checks that after a call to the function BioSPI_GetHeaderFromHandle, the BIR handle has not been freed.

Excerpts

Subclause 9.3.2.3

BioAPI_RETURN BioAPI BioSPI_GetHeaderFromHandle

(BioAPI_HANDLE BSPHandle,

BioAPI_BIR_HANDLE Handle,

*BioAPI_BIR_HEADER *Header);*

Subclause 8.2.3

Retrieves the BIR header identified by Handle. The BIR Handle is not freed by the BSP.

References: 9.3.2.3 and 8.2.3.

Scenario:

1) Load the BSP under test.

2) Attach the BSP under test.

3) Call BioSPI_Enroll.

4) Call BioSPI_GetHeaderFromHandle.

5) Call BioSPI_GetBIRFromHandle, which is expected to return BioAPI_OK.

6) Unload and detach the BSP under test.

If any of the intermediate operations fails, an UNDECIDED conformity response is issued.

Expected results: The call to BioSPI_GetBIRFromHandle returns BioAPI_OK

Assertion language package

```
<package name="01cc0988-0ccf-1085-a367-0002a5d5fd2e" >
      <author>
            ISO/IEC JTC1 SC37
      </author>

      <description>
            This package contains the assertion "BioSPI_GetHeaderFromHandle_BIRHandleNotFreed" (see
                              the "description" element of the assertion below).
      </description>

      <assertion name="BioSPI_GetHeaderFromHandle_BIRHandleNotFreed" model="BSPTesting">
            <description>
                  This assertion checks that after a call to the function BioSPI_GetHeaderFromHandle,
                              the BIR handle has not been freed.
                  The relevant text in BioAPI 2.0 is quoted below from subclauses 9.3.2.3 and 8.2.3
```

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

          NOTE: Details of the function definition are located in clause 8.2.3,
                      BioAPI_GetHeaderFromHandle.

          ________________________________________________________
          Subclause 8.2.3:
          Retrieves the BIR header identified by Handle. The BIR Handle is not freed by the
                      BSP.

          In order to determine conformance with respect to the text above, the following
                      steps are performed:

                1)    Load the BSP under test.
                2)    Attach the BSP under test.
                3)    Call BioSPI_Enroll.
                4)    Call BioSPI_GetHeaderFromHandle.
                5)    Call BioSPI_GetBIRFromHandle, which is expected to return BioAPI_OK.
                6)    Unload and detach the BSP under test.

          If any of the intermediate operations fails, an UNDECIDED conformity response is
                      issued.
    </description>

    <!-- UUID of the BSP under test -->
    <input name="_bspUuid"/>

    <!-- Timeout for the BioAPI_NOTIFY_INSERT event -->
    <input name="_inserttimeout"/>

    <!-- Indicates whether the BSP under test does not claim support for the
                      BioAPI_NOTIFY_SOURCE_PRESENT event notification -->
    <input name="_noSourcePresentSupported"/>

    <!-- Timeout for the BioAPI_NOTIFY_SOURCE_PRESENT event -->
    <input name="_sourcepresenttimeout"/>

    <!-- Timeout for BioSPI_Enroll -->
    <input name="_capturetimeout"/>

    <!-- Invocation of the primary activity of this assertion with input parameter values
                      assigned from the assertion's parameters. -->
    <invoke activity="BioSPI_GetHeaderFromHandle">
          <input name="bspUuid" var="_bspUuid"/>
          <input name="inserttimeouttime" var="_inserttimeout"/>
          <input name="nosourcepresentsupported"
                      var="_noSourcePresentSupported"/>
          <input name="sourcepresenttimeouttime"
                      var="_sourcepresenttimeout"/>
          <input name="capturetimeouttime" var="_capturetimeout"/>
    </invoke>

    <!-- Activity bound to a function of the framework callback interface exposed by the
                      testing component.  This activity will be automatically invoked on each
                      incoming call to the function to which it is bound. -->
    <bind activity="EventHandler"
                package="02c59458-0c46-1085-95d7-0002a5d5fd2e"
                function="BioSPI_EventHandler"/>
</assertion>

<activity name="BioSPI_GetHeaderFromHandle">
    <input name="bspUuid"/>
    <input name="inserttimeouttime"/>
    <input name="nosourcepresentsupported"/>
    <input name="sourcepresenttimeouttime"/>
    <input name="capturetimeouttime"/>

    <!-- This assertion will use BSPHandle "1" for all BioSPI calls that require it -->
    <set name="_bsphandle" value="1"/>
```

```
<!-- Invoke the functions BioSPI_BSPLoad and BioSPI_BSPAttach exposed by the BSP under
                  test.
            The input value for the parameter "unitIDOrNull" is "0", therefore the
                  assertion will test a sensor unit chosen by the BSP. -->
<invoke activity="LoadAndAttach"
            package="02c59458-0c46-1085-95d7-0002a5d5fd2e"
            break_on_break="true">
      <input name="bspUuid" var="bspUuid"/>
      <input name="bspVersion" value="32"/>
      <input name="unitIDOrNull" value="0"/>
      <input name="bspHandle" var="_bsphandle"/>
      <input name="eventtimeouttime" var="inserttimeouttime"/>
</invoke>

<set name="eventtimeoutflag" value="false"/>

<!-- If the BSP under test claims support for the BioAPI_NOTIFY_SOURCE_PRESENT event
                  notification, wait until that notification has been received, but no
                  longer than the specified maximum duration.-->
<wait_until timeout_var="sourcepresenttimeouttime"
            setvar="eventtimeoutflag">
      <or var1="nosourcepresentsupported" var2="_sourcePresent"/>
</wait_until>

<!-- Issue a conformity response.
            If the condition specified in the <description> below is false, an UNDECIDED
                  conformity response is issued and the execution of the activity is
                  interrupted, otherwise a PASS conformity response is issued.-->
<assert_condition response_if_false="undecided"
            break_if_false="true">
      <description>
            Either the BSP under test does not claim support for the
                  BioAPI_NOTIFY_SOURCE_PRESENT event notification, or the event
                  notification has been received within the specified maximum duration.
      </description>
      <not var="eventtimeoutflag"/>
</assert_condition>

<!-- The BSP is ready to capture. Invoke the function BioSPI_Enroll for the purpose of
                  enrollment. -->
<invoke function="BioSPI_Enroll">
      <input name="BSPHandle" var="_bsphandle"/>
      <input name="Purpose"
                  var="__BioAPI_PURPOSE_ENROLL_FOR_VERIFICATION_ONLY"/>
      <input name="Subtype" value="0"/>
      <input name="Timeout" var="capturetimeouttime"/>
      <output name="NewTemplate" setvar="newtemplate_handle"/>
      <return setvar="return"/>
</invoke>

<!-- Issue a conformity response.
            If the condition specified in the <description> below is false, an UNDECIDED
                  conformity response is issued and the execution of the activity is
                  interrupted, otherwise a PASS conformity response is issued.-->
<assert_condition response_if_false="undecided"
            break_if_false="true">
      <description>
            The function BioSPI_Enroll has returned BioAPI_OK
      </description>
      <equal_to var1="return" var2="__BioAPI_OK"/>
</assert_condition>

<!-- Invoke the function BioSPI_GetHeaderFromHandle. -->
<invoke function="BioSPI_GetHeaderFromHandle">
      <input name="BSPHandle" var="_bsphandle"/>
      <input name="Handle" var="newtemplate_handle"/>
      <output name="HeaderVersion" setvar="headerversion"/>
      <output name="ProcessedLevel" setvar="processedLevel"/>
      <output name="FormatOwner" setvar="formatowner"/>
      <output name="FormatType" setvar="formattype"/>
      <output name="Quality" setvar="quality"/>
      <output name="Purpose" setvar="purpose"/>
      <output name="ProductOwner" setvar="productowner" />
      <output name="ProductType" setvar="producttype" />
      <output name="Creation_Year" setvar="creationyear" />
      <output name="Creation_Month" setvar="creationmonth" />
```

```
            <output name="Creation_Day" setvar="creationday" />
            <output name="Creation_Hour" setvar="creationhour" />
            <output name="Creation_Minute" setvar="creationminute" />
            <output name="Creation_Second" setvar="creationsecond" />
            <output name="Expiration_Year" setvar="expirationyear" />
            <output name="Expiration_Month" setvar="expirationmonth" />
            <output name="Expiration_Day" setvar="expirationday" />
            <output name="SBFormatOwner" setvar="securityformatowner" />
            <output name="SBFormatType" setvar="securityformattype" />
            <output name="Index" setvar="index" />
            <return setvar="return"/>
        </invoke>

        <!-- Issue a conformity response.
                    If the condition specified in the <description> below is false, an UNDECIDED
                        conformity response is issued, otherwise a PASS conformity response is
                        issued.-->
        <assert_condition response_if_false="undecided">
            <description>
                The function BioSPI_GetHeaderFromHandle has returned BioAPI_OK.
            </description>
            <equal_to var1="return" var2="__BioAPI_OK"/>
        </assert_condition>

        <!-- Invoke the function BioSPI_GetBIRFromHandle passing the enrolled BIR handle. -->
        <invoke function="BioSPI_GetBIRFromHandle">
            <input name="BSPHandle" var="_bsphandle"/>
            <input name="Handle" var="newtemplate_handle"/>
            <output name="HeaderVersion" setvar="headerversion"/>
            <output name="ProcessedLevel" setvar="processedlevel"/>
            <output name="FormatOwner" setvar="formatowner"/>
            <output name="FormatType" setvar="formattype"/>
            <output name="Quality" setvar="quality"/>
            <output name="Purpose" setvar="purpose"/>
            <output name="ProductOwner" setvar="productowner" />
            <output name="ProductType" setvar="producttype" />
            <output name="Creation_Year" setvar="creationyear" />
            <output name="Creation_Month" setvar="creationmonth" />
            <output name="Creation_Day" setvar="creationday" />
            <output name="Creation_Hour" setvar="creationhour" />
            <output name="Creation_Minute" setvar="creationminute" />
            <output name="Creation_Second" setvar="creationsecond" />
            <output name="Expiration_Year" setvar="expirationyear" />
            <output name="Expiration_Month" setvar="expirationmonth" />
            <output name="Expiration_Day" setvar="expirationday" />
            <output name="SBFormatOwner" setvar="securityformatowner" />
            <output name="SBFormatType" setvar="securityformattype" />
            <output name="Index" setvar="index" />
            <output name="BiometricData" setvar="biometircdata" />
            <output name="SecurityBlock" setvar="securityblock"/>
            <return setvar="return"/>
        </invoke>

        <!-- Issue a conformity response.
                    If the condition specified in the <description> below is false, a FAIL
                        conformity response is issued, otherwise a PASS conformity response is
                        issued.-->
        <assert_condition>
            <description>
                The function BioSPI_GetBIRFromHandle has returned BioAPI_OK.
            </description>
            <equal_to var1="return" var2="__BioAPI_OK"/>
        </assert_condition>

        <!-- Invoke the functions BioSPI_BSPDetach and BioSPI_BSPUnload -->
        <invoke activity="DetachAndUnload"
                package="02c59458-0c46-1085-95d7-0002a5d5fd2e" >
            <input name="bspUuid" var="bspUuid"/>
            <input name="BSPHandle" var="_bsphandle"/>
        </invoke>
    </activity>

</package>
```

8.26 Assertion 8a - *BioSPI_EnableEvents_ValidParam*

Description: This assertion tests BioSPI_EnableEvents with valid input parameters.

Excerpts

Subclause 9.3.3.1

BioAPI_RETURN BioAPI BioSPI_EnableEvents

(BioAPI_HANDLE BSPHandle,

BioAPI_EVENT_MASK Events);

Subclause 8.3.1

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.

Return Value

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

References: 9.3.3.1 and 8.3.1.

Scenario:

1) Load the BSP under test

2) Attach the BSP under test

3) Call BioSPI_EnableEvents with a specified event mask.

4) Check the return code, which is expected to be BioAPI_OK.

5) In BioSPI_EventHandler, check if the enabled events can be received, and if the disabled events are not received.

6) Detach and unload the BSP under test.

If any of the intermediate operations fails, an UNDECIDED conformity response is issued.

Expected results: Enabled events are received and disabled events are not received.

Assertion language package

```
<package name="0333f628-0ccf-1085-aceb-0002a5d5fd2e">
      <author>
            ISO/IEC JTC1 SC37
      </author>

      <description>
            This package contains the assertion "BioSPI_EnableEvents_ValidParam" (see the
                              "description" element of the assertion below).
      </description>
```

```
<assertion name="BioSPI_EnableEvents_ValidParam" model="BSPTesting">
      <description>
            This assertion tests BioSPI_EnableEvents with valid input parameters.
            The relevant text in BioAPI 2.0 is quoted below from subclauses 9.3.3.1 and 8.3.1.
            ___________________________________________________________
            BioAPI_RETURN BioAPI BioSPI_EnableEvents
                  (BioAPI_HANDLE BSPHandle,
                  BioAPI_EVENT_MASK Events);

            NOTE: Details of the function definition are located in clause 8.3.1,
                        BioAPI_EnableEvents.

            ___________________________________________________________
            Subclause 8.3.1
                        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.

            Return Value
            A BioAPI_RETURN value indicating success or specifying a particular error
                        condition. The value BioAPI_OK indicates success. All other values
                        represent an error condition.
            ___________________________________________________________

            In order to determine conformance with respect to the text above, the following
                        steps are performed:

                  1)    Load the BSP under test
                  2)    Attach the BSP under test
                  3)    Call BioSPI_EnableEvents with a specified event mask.
                  4)    Check the return code, which is expected to be BioAPI_OK.
                  5)    In BioSPI_EventHandler, check if the enabled events can be received, and
                        if the disabled events are not received.
                  6)    Detach and unload the BSP under test.

            If any of the intermediate operations fails, an UNDECIDED conformity response is
                        issued.
      </description>

      <!-- UUID of the BSP under test -->
      <input name="_bspUuid"/>

      <!-- Timeout for the NOTIFY_INSERT event -->
      <input name="_inserttimeout"/>

      <!-- Indicates whether the BioAPI_NOTIFY_INSERT event notification is to be enabled -->
      <input name="_eventNotifyInsert"/>

      <!-- Indicates whether the BioAPI_NOTIFY_REMOVE event notification is to be enabled -->
      <input name="_eventNotifyRemove"/>

      <!-- Indicates whether the BioAPI_NOTIFY_FAULT event notification is to be enabled -->
      <input name="_eventNotifyFault"/>

      <!-- Indicates whether the BioAPI_NOTIFY_SOURCE_PRESENT event notification is to be
                        enabled -->
      <input name="_eventNotifySourcePresent"/>

      <!-- Indicates whether the BioAPI_NOTIFY_SOURCE_REMOVED event notification is to be
                        enabled -->
      <input name="_eventNotifySourceRemoved"/>

      <!-- Timeout -->
      <input name="_timeout"/>

      <!-- Invocation of the primary activity of this assertion with input parameter values
                        assigned from the assertion's parameters. -->
      <invoke activity="BioSPI_EnableEvents">
            <input name="bspUuid" var="_bspUuid"/>
            <input name="eventtimeouttime" var="_inserttimeout"/>
            <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"/>
            <input name="timeout" var="_timeout" />
      </invoke>

      <!-- Activity bound to a function of the framework callback interface exposed by the
                         testing component.  This activity will be automatically invoked on each
                         incoming call to the function to which it is bound. -->
      <bind activity="EventHandler"
                   function="BioSPI_EventHandler"/>
</assertion>

<activity name="BioSPI_EnableEvents">
      <input name="bspUuid"/>
      <input name="eventtimeouttime"/>
      <input name="eventnotifyinsert" />
      <input name="eventnotifyremove" />
      <input name="eventnotifyfault" />
      <input name="eventnotifysourcepresent" />
      <input name="eventnotifysourceremoved" />
      <input name="timeout" />

      <set name="_bsphandle" value="1" />

      <set name="_enableEventsCalled" value="false" />

      <!-- Initialize the global variable "_insert" to "false".  The activity "EventHandler"
                         will set this variable to "true" when a NOTIFY_INSERT event notification
                         is received, and will set it to "false" when a NOTIFY_REMOVE event
                         notification is received. -->
      <set name="_insert" value="false"/>

      <!-- Invoke the function BioSPI_BSPLoad. -->
      <invoke function="BioSPI_BSPLoad">
            <input name="BSPUuid" var="bspUuid"/>
            <input name="BioAPINotifyCallback" value="*"/>
            <input name="BFPEnumerationHandler" value="*"/>
            <input name="MemoryFreeHandler" value="*" />
            <return setvar="return"/>
      </invoke>

      <!-- Issue a conformity response.
                   If the condition specified in the <description> below is false, an UNDECIDED
                         conformity response is issued and the execution of the activity is
                         interrupted, otherwise a PASS conformity response is issued.-->
      <assert_condition response_if_false="undecided"
                   break_if_false="true">
            <description>
                   The function BioSPI_BSPLoad has returned BioAPI_OK.
            </description>
            <equal_to var1="return" var2="__BioAPI_OK"/>
      </assert_condition>

      <!-- Wait until the BioAPI_NOTIFY_INSERT event notification has been received, but no
                         longer than the specified maximum duration.-->
      <wait_until timeout_var="eventtimeouttime"
                   setvar="eventtimeoutflag" var="_insert"/>

      <!-- Issue a conformity response.
                   If the condition specified in the <description> below is false, an UNDECIDED
                         conformity response is issued and the execution of the activity is
                         interrupted, otherwise a PASS conformity response is issued.-->
      <assert_condition response_if_false="undecided"
                   break_if_false="true">
            <description>
                   The BioAPI_NOTIFY_INSERT event notification has been received within the
                         specified maximum duration
            </description>
            <not var="eventtimeoutflag"/>
      </assert_condition>

      <!-- Issue the conformance response.
```

```
            If the condition specified in the <description> below is false, a FAIL
                  conformity response is issued and the execution of the activity is
                  interrupted.-->
<assert_condition break_if_false="true">
      <description>
            The event notification received contains valid specification of the unit
                  category
      </description>
      <or>
            <equal_to var1="_unitCategory" var2="__BioAPI_CATEGORY_ARCHIVE"/>
            <equal_to var1="_unitCategory" var2="__BioAPI_CATEGORY_MATCHING_ALG"/>
            <equal_to var1="_unitCategory" var2="__BioAPI_CATEGORY_PROCESSING_ALG"/>
            <equal_to var1="_unitCategory" var2="__BioAPI_CATEGORY_SENSOR"/>
      </or>
</assert_condition>

<!-- Invoke the function BioSPI_BSPAttach and explicitly attach the only one unit. -->
<invoke function="BioSPI_BSPAttach">
      <only_if>
            <equal_to var1="_unitCategory" var2="__BioAPI_CATEGORY_ARCHIVE"/>
      </only_if>
      <input name="BSPUuid" var="bspUuid"/>
      <input name="Version" value="32"/>
      <input name="Unit_1_UnitCategory" var="_unitCategory" />
      <input name="Unit_1_UnitID" var="_unitID"/>
      <input name="Unit_2_UnitCategory" var="__BioAPI_CATEGORY_MATCHING_ALG" />
      <input name="Unit_2_UnitID" var="__BioAPI_DONT_INCLUDE"/>
      <input name="Unit_3_UnitCategory" var="__BioAPI_CATEGORY_PROCESSING_ALG" />
      <input name="Unit_3_UnitID" var="__BioAPI_DONT_INCLUDE"/>
      <input name="Unit_4_UnitCategory" var="__BioAPI_CATEGORY_SENSOR" />
      <input name="Unit_4_UnitID" var="__BioAPI_DONT_INCLUDE"/>
      <input name="NumUnits" value="4" />
      <input name="BSPHandle" var="_bsphandle"/>
      <return setvar="return"/>
</invoke>
<invoke function="BioSPI_BSPAttach">
      <only_if>
            <equal_to var1="_unitCategory" var2="__BioAPI_CATEGORY_MATCHING_ALG"/>
      </only_if>
      <input name="BSPUuid" var="bspUuid"/>
      <input name="Version" value="32"/>
      <input name="Unit_1_UnitCategory" var="__BioAPI_CATEGORY_ARCHIVE" />
      <input name="Unit_1_UnitID" var="__BioAPI_DONT_INCLUDE"/>
      <input name="Unit_2_UnitCategory" var="_unitCategory" />
      <input name="Unit_2_UnitID" var="_unitID"/>
      <input name="Unit_3_UnitCategory" var="__BioAPI_CATEGORY_PROCESSING_ALG" />
      <input name="Unit_3_UnitID" var="__BioAPI_DONT_INCLUDE"/>
      <input name="Unit_4_UnitCategory" var="__BioAPI_CATEGORY_SENSOR" />
      <input name="Unit_4_UnitID" var="__BioAPI_DONT_INCLUDE"/>
      <input name="NumUnits" value="4" />
      <input name="BSPHandle" var="_bsphandle"/>
      <return setvar="return"/>
</invoke>
<invoke function="BioSPI_BSPAttach">
      <only_if>
            <equal_to var1="_unitCategory" var2="__BioAPI_CATEGORY_PROCESSING_ALG"/>
      </only_if>
      <input name="BSPUuid" var="bspUuid"/>
      <input name="Version" value="32"/>
      <input name="Unit_1_UnitCategory" var="__BioAPI_CATEGORY_ARCHIVE" />
      <input name="Unit_1_UnitID" var="__BioAPI_DONT_INCLUDE"/>
      <input name="Unit_2_UnitCategory" var="__BioAPI_CATEGORY_MATCHING_ALG" />
      <input name="Unit_2_UnitID" var="__BioAPI_DONT_INCLUDE"/>
      <input name="Unit_3_UnitCategory" var="_unitCategory" />
      <input name="Unit_3_UnitID" var="_unitID"/>
      <input name="Unit_4_UnitCategory" var="__BioAPI_CATEGORY_SENSOR" />
      <input name="Unit_4_UnitID" var="__BioAPI_DONT_INCLUDE"/>
      <input name="NumUnits" value="4" />
      <input name="BSPHandle" var="_bsphandle"/>
      <return setvar="return"/>
</invoke>
<invoke function="BioSPI_BSPAttach">
      <only_if>
            <equal_to var1="_unitCategory" var2="__BioAPI_CATEGORY_SENSOR"/>
      </only_if>
      <input name="BSPUuid" var="bspUuid"/>
```

```
            <input name="Version" value="32"/>
            <input name="Unit_1_UnitCategory" var="__BioAPI_CATEGORY_ARCHIVE" />
            <input name="Unit_1_UnitID" var="__BioAPI_DONT_INCLUDE"/>
            <input name="Unit_2_UnitCategory" var="__BioAPI_CATEGORY_MATCHING_ALG" />
            <input name="Unit_2_UnitID" var="__BioAPI_DONT_INCLUDE"/>
            <input name="Unit_3_UnitCategory" var="__BioAPI_CATEGORY_PROCESSING_ALG" />
            <input name="Unit_3_UnitID" var="__BioAPI_DONT_INCLUDE"/>
            <input name="Unit_4_UnitCategory" var="_unitCategory" />
            <input name="Unit_4_UnitID" var="_unitID"/>
            <input name="NumUnits" value="4" />
            <input name="BSPHandle" var="_bsphandle"/>
            <return setvar="return"/>
      </invoke>

      <!-- Issue a conformity response.
                  If the condition specified in the <description> below is false, an UNDECIDED
                        conformity response is issued and the execution of the activity is
                        interrupted, otherwise a PASS conformity response is issued.-->
      <assert_condition response_if_false="undecided"
                  break_if_false="true">
            <description>
                  The function BioSPI_BSPAttach has returned BioAPI_OK.
            </description>
            <equal_to var1="return" var2="__BioAPI_OK"/>
      </assert_condition>

      <!-- Invoke the function BioSPI_EnableEvents to enable/disable events -->
      <invoke function="BioSPI_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>

      <!-- Issue a conformity response.
                  If the condition specified in the <description> below is false, an UNDECIDED
                        conformity response is issued and the execution of the activity is
                        interrupted, otherwise a PASS conformity response is issued.-->
      <assert_condition break_if_false="true">
            <description>
                  The function BioSPI_EnableEvents has returned BioAPI_OK.
            </description>
            <equal_to var1="return" var2="__BioAPI_OK"/>
      </assert_condition>

      <set name="_enableEventsCalled" value="true" />
      <set name="_eventFlag" value="false" />

      <!-- Wait until an event notification that is disabled has been received (mistakenly),
                        but no longer than the specified maximum duration. -->
      <wait_until timeout_var="timeout" var="_eventFlag"/>

      <!-- Issue a conformity response.
                  If the condition specified in the <description> below is false, a FAIL
                        conformity response is issued, otherwise a PASS conformity response is
                        issued.-->
      <assert_condition>
            <description>
                  The BSP has not sent any event notifications that are disabled.
            </description>
            <not var="_eventFlag" />
      </assert_condition>

      <!-- Invoke the functions BioSPI_BSPDetach and BioSPI_BSPUnload -->
      <invoke activity="DetachAndUnload"
                  package="02c59458-0c46-1085-95d7-0002a5d5fd2e" >
            <input name="bspUuid" var="bspUuid" />
            <input name="BSPHandle" var="_bsphandle" />
      </invoke>
</activity>
```

```
<!-- This activity will be invoked on incoming calls to the function BioSPI_ModuleEventHandler
                        exposed by the testing component.  In this activity, the global
                        variables "_unitID", "_insert", "_sourcePresent" and "_eventtype" are
                        set depending on the input parameter values. -->
<activity name="EventHandler" atomic="true">
      <input name="BSPUuid"/>
      <input name="UnitID"/>
      <input name="UnitSchema_BspUuid" />
      <input name="UnitSchema_UnitManagerUuid" />
      <input name="UnitSchema_UnitId" />
      <input name="UnitSchema_UnitCategory" />
      <input name="UnitSchema_UnitProperties" />
      <input name="UnitSchema_VendorInformation" />
      <input name="UnitSchema_EventNotifyInsert" />
      <input name="UnitSchema_EventNotifyRemove" />
      <input name="UnitSchema_EventNotifyFault" />
      <input name="UnitSchema_EventNotifySourcePresent" />
      <input name="UnitSchema_EventNotifySourceRemoved" />
      <input name="UnitSchema_UnitPropertyID" />
      <input name="UnitSchema_UnitProperty" />
      <input name="UnitSchema_HardwareVersion" />
      <input name="UnitSchema_FirmwareVersion" />
      <input name="UnitSchema_SoftwareVersion" />
      <input name="UnitSchema_HardwareSerialNumber" />
      <input name="UnitSchema_AuthenticatedHardware" />
      <input name="UnitSchema_MaxBspDbSize" />
      <input name="UnitSchema_MaxIdentify" />
      <input name="EventType" />
      <output name="return"/>

      <!-- Check if the received event notification is compatible with the event mask set by
                        BioSPI_EnableEvents -->
      <invoke activity="checkForUnexpectedEvent">
            <only_if>
                  <same_as var1="_enableEventsCalled" value2="true"/>
                  <existing var="_unitID"/>
            </only_if>
            <input name="UnitID" var="UnitID"/>
            <input name="EventType" var="EventType"/>
      </invoke>

      <!-- Set the global variable "_unitID" if:
                  - it is not set; and
                  - the event notification is BioAPI_NOTIFY_INSERT or
                        BioAPI_NOTIFY_SOURCE_PRESENT; -->
      <set name="_unitID" var="UnitID">
            <only_if>
                  <not>
                        <existing var="_unitID"/>
                  </not>
                  <or>
                        <equal_to var1="EventType"
                                    var2="__BioAPI_NOTIFY_INSERT"/>
                        <equal_to var1="EventType"
                                    var2="__BioAPI_NOTIFY_SOURCE_PRESENT"/>
                  </or>
            </only_if>
      </set>

      <!-- set the unit category -->
      <set name="_unitCategory" var="UnitSchema_UnitCategory">
            <only_if>
                  <not>
                        <existing var="_unitCategory"/>
                  </not>
                  <equal_to var1="EventType"
                              var2="__BioAPI_NOTIFY_INSERT"/>
            </only_if>
      </set>

      <invoke activity="EventHandlerSetGlobalData"
              package="02c59458-0c46-1085-95d7-0002a5d5fd2e"
                  break_on_break="true">
            <only_if>
                  <existing var="_unitID"/>
            </only_if>
```

```
                <input name="UnitID" var="UnitID"/>
                <input name="EventType" var="EventType"/>
            </invoke>

            <set name="return" var="__BioAPI_OK"/>
        </activity>

        <activity name="checkForUnexpectedEvent" atomic="true">
            <input name="UnitID"/>
            <input name="EventType"/>
            <set name="_eventFlag" value="true">
                <only_if>
                    <equal_to var1="_unitID" var2="UnitID"/>
                    <or>
                        <and>
                            <equal_to var1="EventType"
                                    var2="__BioAPI_NOTIFY_INSERT"/>
                            <not var="_eventNotifyInsert" />
                        </and>
                        <and>
                            <equal_to var1="EventType"
                                    var2="__BioAPI_NOTIFY_REMOVE"/>
                            <not var="_eventNotifyRemove" />
                        </and>
                        <and>
                            <equal_to var1="EventType"
                                    var2="__BioAPI_NOTIFY_FAULT"/>
                            <not var="_eventNotifyFault" />
                        </and>
                        <and>
                            <equal_to var1="EventType"
                                    var2="__BioAPI_NOTIFY_SOURCE_PRESENT"/>
                            <not var="_eventNotifySourcePresent" />
                        </and>
                        <and>
                            <equal_to var1="EventType"
                                    var2="__BioAPI_NOTIFY_SOURCE_REMOVED"/>
                            <not var="_eventNotifySourceRemoved" />
                        </and>
                    </or>
                </only_if>
            </set>
        </activity>
    </package>
```

8.27 Assertion 8b - *BioSPI_EnableEvents_InvalidBSPHandle*

Description: This assertion is to test BioSPI_EnableEvents with an invalid module handle.

Excerpts

Subclause 9.3.3.1

BioAPI_RETURN BioAPI BioSPI_EnableEvents

(BioAPI_HANDLE BSPHandle,

BioAPI_EVENT_MASK Events);

Subclause 8.3.1

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.

Return Value

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

Parameters: BSPHandle (input) - the handle of the attached biometric service provider.

References: 9.3.3.1 and 8.3.1.

Scenario:

1) Load the BSP under test

2) Attach the BSP under test

3) Call BioSPI_EnableEvents with an invalid BSP handle. The function is expected to return an error.

4) Detach and unload the BSP under test.

If any of the intermediate operations fails, an UNDECIDED conformity response is issued.

Expected results: The call to BioSPI_EnableEvents returns BioAPIERR_INVALID_BSP_HANDLE.

Assertion language package

```
<package name="04ed0838-0ccf-1085-b64e-0002a5d5fd2e">
      <author>
            ISO/IEC JTC1 SC37
      </author>

      <description>
            This package contains the assertion "BioSPI_EnableEvents_InvalidBSPHandle" (see the
                              "description" element of the assertion below).
      </description>

      <assertion name="BioSPI_EnableEvents_InvalidBSPHandle" model="BSPTesting">
            <description>
                  This assertion is to test BioSPI_EnableEvents with an invalid module handle.
                  The relevant text in BioAPI 2.0 is quoted below from subclauses 9.3.3.1, 8.3.1.1
                              and 8.3.1.2.
                  __________________________________________________________
                  BioAPI_RETURN BioAPI BioSPI_EnableEvents
                        (BioAPI_HANDLE BSPHandle,
                        BioAPI_EVENT_MASK Events);

                  NOTE: Details of the function definition are located in clause 8.3.1,
                              BioAPI_EnableEvents.

                  ______________________________________________________
                  Subclause 8.3.1.1:
                              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.

                  Return Value
                  A BioAPI_RETURN value indicating success or specifying a particular error
                              condition. The value BioAPI_OK indicates success. All other values
                              represent an error condition.
                  ______________________________________________________
                  Subclause 8.3.1.2:
                  Parameters: BSPHandle (input) - the handle of the attached biometric service
                              provider.
            ______________________________________________________
            In order to determine conformance with respect to the text above, the following
                        steps are performed:

                  1)    Load the BSP under test
                  2)    Attach the BSP under test
                  3)    Call BioSPI_EnableEvents with an invalid BSP handle.  The function is
                        expected to return an error.
                  4)    Detach and unload the BSP under test.

            If any of the intermediate operations fails, an UNDECIDED conformity response is
                        issued.
      </description>

      <!-- UUID of the BSP under test -->
      <input name="_bspUuid"/>

      <!-- Indicates whether the BioAPI_NOTIFY_INSERT event is to be enabled -->
      <input name="_eventNotifyInsert"/>

      <!-- Indicates whether the BioAPI_NOTIFY_REMOVE event is to be enabled -->
      <input name="_eventNotifyRemove"/>

      <!-- Indicates whether the BioAPI_NOTIFY_FAULT event is to be enabled -->
      <input name="_eventNotifyFault"/>

      <!-- Indicates whether the BioAPI_NOTIFY_SOURCE_PRESENT event is to be enabled -->
      <input name="_eventNotifySourcePresent"/>

      <!-- Indicates whether the BioAPI_NOTIFY_SOURCE_REMOVED event is to be enabled -->
      <input name="_eventNotifySourceRemoved"/>

      <!-- Timeout for the BioAPI_NOTIFY_INSERT event notification -->
      <input name="_inserttimeouttime" />

      <!-- Invocation of the primary activity of this assertion with input parameter values
                  assigned from the assertion's parameters. -->
      <invoke activity="BioSPI_EnableEvents">
            <input name="bspUuid" var="_bspUuid"/>
            <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"/>
            <input name="inserttimeouttime" var="_inserttimeouttime" />
      </invoke>

      <!-- Activity bound to a function of the framework callback interface exposed by the
                  testing component.  This activity will be automatically invoked on each
                  incoming call to the function to which it is bound. -->
      <bind activity="EventHandler"
                  package="02c59458-0c46-1085-95d7-0002a5d5fd2e"
                  function="BioSPI_EventHandler"/>
</assertion>

<activity name="BioSPI_EnableEvents">
      <input name="bspUuid"/>
      <input name="eventnotifyinsert" />
      <input name="eventnotifyremove" />
      <input name="eventnotifyfault" />
      <input name="eventnotifysourcepresent" />
      <input name="eventnotifysourceremoved" />
      <input name="inserttimeouttime" />

      <!-- This assertion will use BSPHandle "1" for all BioSPI calls that require it -->
      <set name="_bsphandle" value="1"/>

      <!-- Invoke the functions BioSPI_BSPLoad and BioSPI_BSPAttach exposed by the BSP under
                  test.
            The input value for the parameter "unitIDOrNull" is "0", therefore the
                  assertion will test a sensor unit chosen by the BSP. -->
      <invoke activity="LoadAndAttach"
                  package="02c59458-0c46-1085-95d7-0002a5d5fd2e"
                  break_on_break="true">
            <input name="bspUuid" var="bspUuid"/>
            <input name="bspVersion" value="32"/>
            <input name="unitIDOrNull" value="0"/>
            <input name="bspHandle" var="_bsphandle"/>
            <input name="eventtimeouttime" var="inserttimeouttime"/>
      </invoke>

      <!-- Invoke the function BioSPI_EnableEvents with an invalid module handle.-->
      <invoke function="BioSPI_EnableEvents" >
            <input name="BSPHandle" value="0" />
            <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>

      <!-- Issue a conformity response.
                  If the condition specified in the <description> below is false, a FAIL
                        conformity response is issued, otherwise a PASS conformity response is
                        issued.-->
      <assert_condition>
            <description>
                  The function BioSPI_EnableEvents has returned BioAPIERR_INVALID_BSP_HANDLE
            </description>
            <equal_to var1="return"
                        var2="__BioAPIERR_BSP_INVALID_BSP_HANDLE"/>
      </assert_condition>

      <!-- Invoke the functions BioSPI_BSPDetach and BioSPI_BSPUnload -->
      <invoke activity="DetachAndUnload"
                  package="02c59458-0c46-1085-95d7-0002a5d5fd2e" >
            <input name="bspUuid" var="bspUuid" />
            <input name="BSPHandle" var="_bsphandle" />
      </invoke>
</activity>

</package>
```

8.28 Assertion 9a - *BioSPI_Capture_AuditData*

Description: This assertion tests the function BioSPI_Capture with AuditData having a non-NULL value.

Excerpts

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);*

Subclause 8.4.1.1

If AuditData is not NULL, a BIR of type 'raw' may be returned. A BSP may return a handle value of BioAPI_UNSUPPORTED_BIR_HANDLE to indicate AuditData is not supported, or a value of BioAPI_INVALID_BIR_HANDLE to indicate that no audit data is available.

Parameters: AuditData (output/optional) - a handle to a BIR containing raw biometric data. This data may be used to provide human-identifiable data of the person at the device.

If the pointer is NULL on input, no audit data is collected.

Not all BSPs support the collection of audit data.

A BSP may return a handle value of BioAPI_UNSUPPORTED_BIR_HANDLE to indicate AuditData is not supported, or a value of BioAPI_INVALID_BIR_HANDLE to indicate that no audit data is available.

Subclause 7.47

BioAPI_OPTIONS_MASK

#define BioAPI_RAW (0x00000001)   If set, indicates that the BSP supports the return of raw/audit data.

Subclause A.4.6.2.1

Return of raw data.

Functions involving the capture of biometric data from a sensor may optionally support the return of this raw data for purposes of display or audit.

If supported, the output parameter AuditData will contain a pointer to this data.

If not supported, the BSP will return a value of -1.

References: 9.3.4.1, 8.4.1.1, 7.47, and A.4.6.2.1

Scenario:

1) Load the BSP under test

2) Attach the BSP under test

3) Invoke the function BioSPI_Capture with AuditData set to a non-NULL value

4) Invoke the function BioSPI_GetHeaderFromHandle to check the obtained AuditData BIR handle. If audit data is supported, the processed level of the audit data BIR is expected to be RAW. If audit data is not supported, the function is expected to return BioAPI_UNSUPPORTED_BIR_HANDLE.

5) Detach and unload the BSP under test.

Expected results: If audit data is supported, the processed level of the audit data BIR is RAW. Otherwise, the call to BioSPI_Capture returns the special handle value BioAPI_UNSUPPORTED_BIR_HANDLE.

Assertion language package

```
<package name="02704c50-0cd8-1085-96cb-0002a5d5fd2e">
      <author>
            ISO/IEC JTC1 SC37
      </author>

      <description>
            This package contains the assertion "BioSPI_Capture_AuditData" (see the "description"
                              element of the assertion below).
      </description>

      <assertion name="BioSPI_Capture_AuditData" model="BSPTesting">
            <description>
                  This assertion tests the function BioSPI_Capture with AuditData having a non-NULL
                              value.
                  The relevant text in BioAPI 2.0 is quoted below from subclauses 9.3.4.1, 8.4.1.1,
                              7.47 and A.4.6.2.1.
                  _________________________________________________________
                  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);

                  NOTE: Details of the function definition are located in clause 8.4.1,
                              BioAPI_Capture.

                  _________________________________________________________
                  Subclause 8.4.1.1:
                  If AuditData is not NULL, a BIR of type 'raw' may be returned. A BSP may return a
                              handle value of BioAPI_UNSUPPORTED_BIR_HANDLE to indicate AuditData is
                              not supported, or a value of BioAPI_INVALID_BIR_HANDLE to indicate that
                              no audit data is available.

                  Parameters: AuditData (output/optional) - a handle to a BIR containing raw
                              biometric data.
                  This data may be used to provide human-identifiable data of the person at the
                              sensor unit.
                  If the pointer is NULL on input, no audit data is collected.
                  Not all BSPs support the collection of audit data.
                  A BSP may return a handle value of BioAPI_UNSUPPORTED_BIR_HANDLE to indicate
                              AuditData is not supported, or a value of BioAPI_INVALID_BIR_HANDLE to
                              indicate that no audit data is available.
                  _________________________________________________________
                  Subclause 7.47: BioAPI_OPTIONS_MASK
                  #define BioAPI_RAW (0x00000001) If set, indicates that the BSP supports the return
                              of raw/audit data.
                  _________________________________________________________
                  Subclause A.4.6.2.1:
                  Return of raw data.
                  Functions involving the capture of biometric data from a sensor may optionally
                              support the return of this raw data for purposes of display or audit.
                  If supported, the output parameter AuditData will contain a pointer to this data.
                  If not supported, the BSP will return a value of -1.
                  _________________________________________________________

                  In order to determine conformance with respect to the text above, the following
                              steps are performed:

                        1)    Load the BSP under test
                        2)    Attach the BSP under test
                        3)    Invoke the function BioSPI_Capture with AuditData set to a non-NULL
                              value
                        4)    Invoke the function BioSPI_GetHeaderFromHandle to check the obtained
                              AuditData BIR handle.  If audit data is supported, the processed level
                              of the audit data BIR is expected to be RAW.  If audit data is not
                              supported, the function is expected to return
                              BioAPI_UNSUPPORTED_BIR_HANDLE.
                        5)    Detach and unload the BSP under test.
            </description>
```

```
        <!-- UUID of the BSP under test -->
        <input name="_bspUuid"/>

        <!-- Timeout for the BioAPI_NOTIFY_INSERT event -->
        <input name="_inserttimeout"/>

        <!-- Indicates whether the BSP under test does not claim support for the
                       BioAPI_NOTIFY_SOURCE_PRESENT event notification -->
        <input name="_noSourcePresentSupported" />

        <!-- Timeout for the BioAPI_NOTIFY_SOURCE_PRESENT event -->
        <input name="_sourcepresenttimeout"/>

        <!-- Timeout for BioSPI_Capture -->
        <input name="_capturetimeout"/>

        <!-- Indicates whether the BSP under test claims support for audit data -->
        <input name="_supportAuditData" />

        <!-- Invocation of the primary activity of this assertion with input parameter values
                       assigned from the assertion's parameters. -->
        <invoke activity="CapturedBIR_AuditDataPresent">
              <input name="bspUuid" var="_bspUuid"/>
              <input name="inserttimeouttime" var="_inserttimeout"/>
              <input name="nosourcepresentsupported"
                       var="_noSourcePresentSupported" />
              <input name="sourcepresenttimeouttime"
                       var="_sourcepresenttimeout"/>
              <input name="capturetimeouttime" var="_capturetimeout"/>
              <input name="supportAuditData" var="_supportAuditData" />
        </invoke>

        <!-- Activity bound to a function of the framework callback interface exposed by the
                       testing component.  This activity will be automatically invoked on each
                       incoming call to the function to which it is bound. -->
        <bind activity="EventHandler"
                    package="02c59458-0c46-1085-95d7-0002a5d5fd2e"
                    function="BioSPI_EventHandler"/>
  </assertion>

  <activity name="CapturedBIR_AuditDataPresent">
        <input name="bspUuid"/>
        <input name="inserttimeouttime"/>
        <input name="nosourcepresentsupported" />
        <input name="sourcepresenttimeouttime"/>
        <input name="capturetimeouttime"/>
        <input name="supportAuditData" />

        <!-- This assertion will use BSPHandle "1" for all BioSPI calls that require it -->
        <set name="_bsphandle" value="1"/>

        <!-- Invoke the functions BioSPI_BSPLoad and BioSPI_BSPAttach exposed by the BSP under
                       test.
                    The input value for the parameter "unitIDOrNull" is "0", therefore the
                       assertion will test a sensor unit chosen by the BSP. -->
        <invoke activity="LoadAndAttach"
                    package="02c59458-0c46-1085-95d7-0002a5d5fd2e"
                    break_on_break="true">
              <input name="bspUuid" var="bspUuid"/>
              <input name="bspVersion" value="32"/>
              <input name="unitIDOrNull" value="0"/>
              <input name="bspHandle" var="_bsphandle"/>
              <input name="eventtimeouttime" var="inserttimeouttime"/>
        </invoke>

        <set name="eventtimeoutflag" value="false"/>
        <!-- If the BSP under test claims support for the BioAPI_NOTIFY_SOURCE_PRESENT event
                       notification, wait until that notification has been received, but no
                       longer than the specified maximum duration.-->
        <wait_until timeout_var="sourcepresenttimeouttime"
                    setvar="eventtimeoutflag">
              <or var1="nosourcepresentsupported" var2="_sourcePresent" />
        </wait_until>
```

```
<!-- Issue a conformity response.
            If the condition specified in the <description> below is false, an UNDECIDED
                  conformity response is issued and the execution of the activity is
                  interrupted, otherwise a PASS conformity response is issued.-->
<assert_condition response_if_false="undecided"
            break_if_false="true">
      <description>
            Either the BSP under test does not claim support for the
                  BioAPI_NOTIFY_SOURCE_PRESENT event notification, or the event
                  notification has been received within the specified maximum duration.
      </description>
      <not var="eventtimeoutflag"/>
</assert_condition>

<!-- The BSP is ready to capture. Invoke the function BioSPI_Capture, with AuditData set
                  to a non-NULL value.  The handle of the captured BIR is stored in the
                  variable "capturedbir". -->
<invoke function="BioSPI_Capture">
      <input name="BSPHandle" var="_bsphandle"/>
      <input name="Purpose"
                  var="__BioAPI_PURPOSE_ENROLL_FOR_VERIFICATION_ONLY"/>
      <input name="Subtype" value="0"/>
      <input name="Timeout" var="capturetimeouttime"/>
      <output name="CapturedBIR" setvar="capturedbir_handle"/>
      <output name="AuditData" setvar="auditbir_handle"/>
      <return setvar="return"/>
</invoke>

<!-- Issue a conformity response.
            If the condition specified in the <description> below is false, an UNDECIDED
                  conformity response is issued and the execution of the activity is
                  interrupted, otherwise a PASS conformity response is issued.-->
<assert_condition response_if_false="undecided"
            break_if_false="true">
      <description>
            The function BioSPI_Capture has returned BioAPI_OK.
      </description>
      <equal_to var1="return" var2="__BioAPI_OK"/>
</assert_condition>

<!-- Issue a conformity response.
            If the condition specified in the <description> below is false, a FAIL
                  conformity response is issued and the execution of the activity is
                  interrupted, otherwise a PASS conformity response is issued.-->
<assert_condition break_if_false="true">
      <description>
            The output AuditData BIR handle is either a valid value or
                  BioAPI_INVALID_BIR_HANDLE (when audit data is supported), or
                  BioAPI_UNSUPPORTED_BIR_HANDLE (when audit data is not supported).
      </description>
      <or>
            <and>
                  <same_as var1="supportAuditData" value2="true" />
                  <or>
                        <equal_to var1="auditbir_handle"
                                    var2="__BioAPI_INVALID_BIR_HANDLE"/>
                        <greater_than_or_equal_to var1="auditbir_handle" value2="0" />
                  </or>
            </and>
            <and>
                  <same_as var1="supportAuditData" value2="false" />
                  <equal_to var1="auditbir_handle"
                              var2="__BioAPI_UNSUPPORTED_BIR_HANDLE" />
            </and>
      </or>
</assert_condition>

<!-- Invoke activity the check the processed level of the audit data BIR handle -->
<invoke activity="check_auditdata_type" >
      <only_if>
            <same_as var1="supportAuditData" value2="true" />
            <greater_than_or_equal_to var1="auditbir_handle" value2="0" />
      </only_if>
```

```
                <input name="bsphandle" var="_bsphandle"/>
                <input name="auditbirhandle" var="auditbir_handle" />
          </invoke>

          <!-- Invoke the functions BioSPI_BSPDetach and BioSPI_BSPUnload -->
          <invoke activity="DetachAndUnload"
                      package="02c59458-0c46-1085-95d7-0002a5d5fd2e" >
                <input name="bspUuid" var="bspUuid" />
                <input name="BSPHandle" var="_bsphandle" />
          </invoke>
    </activity>

    <activity name="check_auditdata_type">
          <input name="bsphandle" />
          <input name="auditbirhandle" />

          <!-- Invoke the function BioSPI_GetHeaderFromHandle.-->
          <invoke function="BioSPI_GetHeaderFromHandle">
                <input name="BSPHandle" var="_bsphandle"/>
                <input name="Handle" var="auditbirhandle"/>
                <output name="ProcessedLevel" setvar="processedLevel"/>
                <return setvar="return"/>
          </invoke>

          <!-- Issue a conformity response.
                      If the condition specified in the <description> below is false, an UNDECIDED
                            conformity response is issued and the execution of the activity is
                            interrupted, otherwise a PASS conformity response is issued.-->
          <assert_condition response_if_false="undecided"
                      break_if_false="true">
                <description>
                      The function BioSPI_GetHeaderFromHandle has returned BioAPI_OK.
                </description>
                <equal_to var1="return" var2="__BioAPI_OK"/>
          </assert_condition>

          <!-- Issue a conformity response.
                      If the condition specified in the <description> below is false, a FAIL
                            conformity response is issued, otherwise a PASS conformity response is
                            issued.-->
          <assert_condition>
                <description>
                      The processed level of the audit data BIR is RAW.
                </description>
                <equal_to var1="processedLevel"
                            var2="__BioAPI_BIR_DATA_TYPE_RAW"/>
          </assert_condition>
    </activity>

</package>
```

8.29 Assertion 9b - *BioSPI_Capture_ReturnQuality*

Description: This assertion invokes the function BioSPI_Capture and checks if the header of the captured BIR contains a valid quality value (in the range 0-100).

Excerpts

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);*

Subclause 8.4.1.1

This function captures samples for the purpose specified, and returns either an 'intermediate' type BIR (if the Process function needs to be called), or a 'processed' BIR (if not).

Subclause 7.49

BioAPI_QUALITY - A value indicating the quality of the biometric data in a BIR.

Quality measurements are reported as an integral value in the range 0-100 except as follows:

Value of -1: BioAPI_QUALITY was not set by the BSP (reference BSP vendor's documentation for explanation).

Value of -2: BioAPI_QUALITY is not supported by the BSP.

Subclause 7.47

#define BioAPI_QUALITY_INTERMEDIATE (0x00000004) If set, BSP supports the return of a quality value (in the BIR header) for intermediate biometric data.

#define BioAPI_QUALITY_PROCESSED (0x00000008) If set, BSP supports the return of quality value (in the BIR header) for processed biometric data.

Subclause A.4.6.2.2

Return of Quality.

Upon the new capture of biometric data from a sensor, the BSP may calculate a relative quality value associated with this data, which it will include in the header of the returned CapturedBIR (and the optional AuditData).

If supported, this header field will be filled with a positive value between 1 and 100.

If not supported, this field will be set to -2. This would occur during BioAPI_Capture and BioAPI_Enroll.

References: 9.3.4.1, 8.4.1.1, 7.49 , 7.47, and A.4.6.2.2

Scenario:

1) Load the BSP under test

2) Attach the BSP under test

3) Call BioSPI_Capture

4) Call BioSPI_GetHeaderFromHandle

5) Check the quality value of the returned BIR.

If any of the intermediate operations fails, an UNDECIDED conformity response is issued.

Expected results: The header of the captured BIR contains a valid quality value (in the range 0-100) if quality is supported, or -2 if quality is not supported.

Assertion language package

```
<package name="03f601f0-0cd8-1085-bd59-0002a5d5fd2e">
      <author>
            ISO/IEC JTC1 SC37
      </author>

      <description>
            This package contains the assertion "BioSPI_Capture_ReturnQuality" (see the "description"
                              element of the assertion below).
      </description>

      <assertion name="BioSPI_Capture_ReturnQuality" model="BSPTesting">
            <description>
                  This assertion invokes the function BioSPI_Capture and checks if the header of the
                              captured BIR contains a valid quality value (in the range 0-100).
                  The relevant text in BioAPI 2.0 is quoted below from subclauses 9.3.4.1, 8.4.1,
                              7.47, 7.49 and A.4.6.2.2.

                  ____________________________________________________________
                  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);

                  NOTE: Details of the function definition are located in clause 8.4.1,
                              BioAPI_Capture.

                  ____________________________________________________________
                  Subclause 7.49
                  BioAPI_QUALITY - A value indicating the quality of the biometric data in a BIR.

                  Quality measurements are reported as an integral value in the range 0-100 except as
                              follows:
                  Value of -1: BioAPI_QUALITY was not set by the BSP (reference BSP vendor's
                              documentation for explanation).
                  Value of -2: BioAPI_QUALITY is not supported by the BSP.
                  ____________________________________________________________
                  Subclause 7.47:
                  #define BioAPI_QUALITY_INTERMEDIATE (0x00000004)
                  If set, BSP supports the return of a quality value (in the BIR header) for
                              intermediate biometric data.
                  #define BioAPI_QUALITY_PROCESSED (0x00000008) \
                  If set, BSP supports the return of quality value (in the BIR header) for processed
                              biometric data.
                  ____________________________________________________________
                  Subclause A.4.6.2.2:
                  Return of Quality.
                  Upon the new capture of biometric data from a sensor, the BSP may calculate a
                              relative quality value associated with this data, which it will include
                              in the header of the returned CapturedBIR (and the optional AuditData).
                  If supported, this header field will be filled with a positive value between 1 and
                              100.
                  If not supported, this field will be set to -2. This would occur during
                              BioAPI_Capture and BioAPI_Enroll.
                  ____________________________________________________________

                  In order to determine conformance with respect to the text above, the following
                              steps are performed:

                        1)    Load the BSP under test
                        2)    Attach the BSP under test
                        3)    Call BioSPI_Capture
                        4)    Call BioSPI_GetHeaderFromHandle
                        5)    Check the quality value of the returned BIR.

                  If any of the intermediate operations fails, an UNDECIDED conformity response is
                              issued.
            </description>
```

```
        <!-- UUID of the BSP under test -->
        <input name="_bspUuid"/>

        <!-- Timeout for the BioAPI_NOTIFY_INSERT event -->
        <input name="_inserttimeout"/>

        <!-- Indicates whether the BSP under test does not claim support for the
                         BioAPI_NOTIFY_SOURCE_PRESENT event notification -->
        <input name="_noSourcePresentSupported" />

        <!-- Timeout for the BioAPI_NOTIFY_SOURCE_PRESENT event -->
        <input name="_sourcepresenttimeout"/>

        <!-- Timeout for BioSPI_Capture -->
        <input name="_capturetimeout"/>

        <!-- Indicates whether the BSP under test claims support for quality in an intermediate
                         BIR -->
        <input name="_intermediateQualitySupported" />

        <!-- Indicates whether the BSP under test claims support for quality in a processed BIR -
                         ->
        <input name="_processedQualitySupported" />

        <!-- Invocation of the primary activity of this assertion with input parameter values
                         assigned from the assertion's parameters. -->
        <invoke activity="BioSPI_Capture_ReturnQuality">
              <input name="bspUuid" var="_bspUuid"/>
              <input name="inserttimeouttime" var="_inserttimeout"/>
              <input name="nosourcepresentsupported"
                         var="_noSourcePresentSupported" />
              <input name="sourcepresenttimeouttime"
                         var="_sourcepresenttimeout"/>
              <input name="capturetimeouttime" var="_capturetimeout"/>
              <input name="intermediateQualitySupported"
                         var="_intermediateQualitySupported" />
              <input name="processedQualitySupported"
                         var="_processedQualitySupported" />

        </invoke>
        <!-- Activity bound to a function of the framework callback interface exposed by the
                         testing component. This activity will be automatically invoked on each
                         incoming call to the function to which it is bound. -->
              <bind activity="EventHandler"
                    package="02c59458-0c46-1085-95d7-0002a5d5fd2e"
                    function="BioSPI_EventHandler"/>
  </assertion>

  <activity name="BioSPI_Capture_ReturnQuality">
        <input name="bspUuid"/>
        <input name="inserttimeouttime"/>
        <input name="nosourcepresentsupported" />
        <input name="sourcepresenttimeouttime"/>
        <input name="capturetimeouttime"/>
        <input name="intermediateQualitySupported" />
        <input name="processedQualitySupported"  />

        <!-- This assertion will use BSPHandle "1" for all BioSPI calls that require it -->
        <set name="_bsphandle" value="1"/>

        <!-- Invoke the functions BioSPI_BSPLoad and BioSPI_BSPAttach exposed by the BSP under
                         test.
                    The input value for the parameter "unitIDOrNull" is "0", therefore the
                         assertion will test a sensor unit chosen by the BSP. -->
        <invoke activity="LoadAndAttach"
                    package="02c59458-0c46-1085-95d7-0002a5d5fd2e"
                    break_on_break="true">
              <input name="bspUuid" var="bspUuid"/>
              <input name="bspVersion" value="32"/>
              <input name="unitIDOrNull" value="0"/>
              <input name="bspHandle" var="_bsphandle"/>
              <input name="eventtimeouttime" var="inserttimeouttime"/>
        </invoke>

        <set name="eventtimeoutflag" value="false"/>
```

```
<!-- If the BSP under test claims support for the BioAPI_NOTIFY_SOURCE_PRESENT event
                  notification, wait until that notification has been received, but no
                  longer than the specified maximum duration.-->
<wait_until timeout_var="sourcepresenttimeouttime"
            setvar="eventtimeoutflag">
      <or var1="nosourcepresentsupported" var2="_sourcePresent" />
</wait_until>

<!-- Issue a conformity response.
            If the condition specified in the <description> below is false, an UNDECIDED
                  conformity response is issued and the execution of the activity is
                  interrupted, otherwise a PASS conformity response is issued.-->
<assert_condition response_if_false="undecided"
            break_if_false="true">
      <description>
            Either the BSP under test does not claim support for the
                  BioAPI_NOTIFY_SOURCE_PRESENT event notification, or the event
                  notification has been received within the specified maximum duration.
      </description>
      <not var="eventtimeoutflag"/>
</assert_condition>

<!-- The BSP is ready to capture. Invoke the function BioSPI_Capture for the purpose of
                  creating a template. The handle of the captured BIR is stored in the
                  variable "capturedbir". -->
<invoke function="BioSPI_Capture">
      <input name="BSPHandle" var="_bsphandle"/>
      <input name="Purpose"
                  var="__BioAPI_PURPOSE_ENROLL_FOR_VERIFICATION_ONLY"/>
      <input name="Subtype" value="0"/>
      <input name="Timeout" var="capturetimeouttime"/>
      <input name="no_AuditData" value="true"/>
      <output name="CapturedBIR" setvar="capturedbir_handle"/>
      <return setvar="return"/>
</invoke>

<!-- Issue a conformity response.
            If the condition specified in the <description> below is false, an UNDECIDED
                  conformity response is issued and the execution of the activity is
                  interrupted, otherwise a PASS conformity response is issued.-->
<assert_condition response_if_false="undecided"
            break_if_false="true">
      <description>
            The function BioSPI_Capture has returned BioAPI_OK
      </description>
      <equal_to var1="return" var2="__BioAPI_OK"/>
</assert_condition>

<!-- Invoke the function BioSPI_GetHeaderFromHandle. -->
<invoke function="BioSPI_GetHeaderFromHandle">
      <input name="BSPHandle" var="_bsphandle"/>
      <input name="Handle" var="capturedbir_handle"/>
      <output name="ProcessedLevel" setvar="processedLevel"/>
      <output name="Quality" setvar="quality"/>
      <return setvar="return"/>
</invoke>

<!-- Issue a conformity response.
            If the condition specified in the <description> below is false, an UNDECIDED
                  conformity response is issued and the execution of the activity is
                  interrupted, otherwise a PASS conformity response is issued.-->
<assert_condition response_if_false="undecided"
            break_if_false="true">
      <description>
            The function BioSPI_GetHeaderFromHandle has returned BioAPI_OK.
      </description>
      <equal_to var1="return" var2="__BioAPI_OK"/>
</assert_condition>

<invoke activity="check_quality_supported"
            package="02c59458-0c46-1085-95d7-0002a5d5fd2e"
            break_on_break="true" >
      <only_if>
            <same_as var1="processedQualitySupported"
                        value2="true"/>
            <same_as var1="intermediateQualitySupported"
```

```
                        value2="true"/>
            </only_if>
            <input name="quality" var="quality"/>
      </invoke>

      <invoke activity="check_quality_ps_ins"
                  break_on_break="true" >
            <only_if>
                  <same_as var1="processedQualitySupported"
                        value2="true" />
                  <same_as var1="intermediateQualitySupported"
                        value2="false" />
            </only_if>
            <input name="quality" var="quality" />
            <input name="processedLevel" var="processedLevel" />
      </invoke>

      <invoke activity="check_quality_pns_is"
                  break_on_break="true" >
            <only_if>
                  <same_as var1="processedQualitySupported"
                        value2="false" />
                  <same_as var1="intermediateQualitySupported"
                        value2="true" />
            </only_if>
            <input name="quality" var="quality" />
            <input name="processedLevel" var="processedLevel" />
      </invoke>

      <invoke activity="check_quality_not_supported"
                  package="02c59458-0c46-1085-95d7-0002a5d5fd2e"
                  break_on_break="true" >
            <only_if>
                  <same_as var1="processedQualitySupported"
                        value2="false"/>
                  <same_as var1="intermediateQualitySupported"
                        value2="false"/>
            </only_if>
            <input name="quality" var="quality"/>
      </invoke>

      <!-- Invoke the functions BioSPI_BSPDetach and BioSPI_BSPUnload -->
      <invoke activity="DetachAndUnload"
                  package="02c59458-0c46-1085-95d7-0002a5d5fd2e" >
            <input name="bspUuid" var="bspUuid" />
            <input name="BSPHandle" var="_bsphandle" />
      </invoke>
</activity>

<!-- This activity checks the value of the returned quality.  It is used if the BSP claims to
                        support processed quality but not intermediate quality. -->
<activity name="check_quality_ps_ins" >
      <input name="quality" />
      <input name="processedLevel" />

      <invoke activity="check_quality_not_supported"
                  package="02c59458-0c46-1085-95d7-0002a5d5fd2e" >
            <only_if>
                  <equal_to var1="processedLevel"
                        var2="__BioAPI_BIR_DATA_TYPE_INTERMEDIATE" />
            </only_if>
            <input name="quality" var="quality" />
      </invoke>

      <invoke activity="check_quality_supported"
                  package="02c59458-0c46-1085-95d7-0002a5d5fd2e" >
            <only_if>
                  <equal_to var1="processedLevel"
                        var2="__BioAPI_BIR_DATA_TYPE_PROCESSED" />
            </only_if>
            <input name="quality" var="quality" />
      </invoke>
</activity>
```

```
      <!-- This activity checks the value of the returned quality.  It is used if the BSP claims to
                              support intermediate quality but does not claim to support processed
                              quality. -->
      <activity name="check_quality_pns_is" >
            <input name="quality" />
            <input name="processedLevel" />

            <invoke activity="check_quality_supported"
                        package="02c59458-0c46-1085-95d7-0002a5d5fd2e" >
                  <only_if>
                        <equal_to var1="processedLevel"
                                    var2="__BioAPI_BIR_DATA_TYPE_INTERMEDIATE" />
                  </only_if>
                  <input name="quality" var="quality" />
            </invoke>

            <invoke activity="check_quality_not_supported"
                        package="02c59458-0c46-1085-95d7-0002a5d5fd2e" >
                  <only_if>
                        <equal_to var1="processedLevel"
                                    var2="__BioAPI_BIR_DATA_TYPE_PROCESSED" />
                  </only_if>
                  <input name="quality" var="quality" />
            </invoke>
      </activity>

</package>
```

8.30 Assertion 9c - *BioSPI_Capture_IntermediateProcessedBIR*

Description: This assertion checks if the BIR returned by the function BioSPI_Capture has a processed level of either INTERMEDIATE or PROCESSED.

Excerpts

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);*

Subclause 8.4.1.1

This function captures samples for the purpose specified, and returns either an 'intermediate' type BIR (if the Process function needs to be called), or a 'processed' BIR (if not).

Subclause 8.4.1.2

CapturedBIR (output) a handle to a BIR containing captured data. This data is either an 'intermediate' type BIR, (which can only be used by either the Process or CreateTemplate functions, depending on the purpose), or a 'processed' BIR, (which can be used directly by VerifyMatch or IdentifyMatch, depending on the purpose).

References: 9.3.4.1. 8.4.1.1, and 8.4.1.2

Scenario:

1) Load the BSP under test.

2) Attach the BSP under test.

3) Call BioSPI_Capture.

4) Call BioSPI_GetHeaderFromHandle.

5) Check the processed level of the returned BIR.

If any of the intermediate operations fails, an UNDECIDED conformity response is issued.

Expected results: The processed level of the returned BIR is either INTERMEDIATE or PROCESSED.

Assertion language package

```
<package name="055ddb08-0cd6-1085-a6d3-0002a5d5fd2e">
      <author>
            ISO/IEC JTC1 SC37
      </author>

      <description>
            This package contains the assertion "BioSPI_Capture_IntermediateProcessedBIR" (see the
                              "description" element of the assertion below).
      </description>

      <assertion name="BioSPI_Capture_IntermediateProcessedBIR" model="BSPTesting">
            <description>
                  This assertion checks if the BIR returned by the function BioSPI_Capture has a
                              processed level of either INTERMEDIATE or PROCESSED.
                  The relevant text in BioAPI 2.0 is quoted below from subclause 9.3.4.1. 8.4.1.1 and
                              8.4.1.2.

                  _______________________________________________________
                  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);

                  NOTE: Details of the function definition are located in clause 8.4.1,
                              BioAPI_Capture.

                  _______________________________________________________
                  Subclause 8.4.1.1:
                  This function captures samples for the purpose specified, and returns either an
                              'intermediate' type BIR (if the Process function needs to be called), or
                              a 'processed' BIR (if not).
                  _______________________________________________________
                  Subclause 8.4.1.2:
                  CapturedBIR (output) a handle to a BIR containing captured data. This data is
                              either an 'intermediate' type BIR, (which can only be used by either the
                              Process or CreateTemplate functions, depending on the purpose), or a
                              'processed' BIR, (which can be used directly by VerifyMatch or
                              IdentifyMatch, depending on the purpose).
                  _______________________________________________________

                  In order to determine conformance with respect to the text above, the following
                              steps are performed:

                        1)    Load the BSP under test.
                        2)    Attach the BSP under test.
                        3)    Call BioSPI_Capture.
                        4)    Call BioSPI_GetHeaderFromHandle.
                        5)    Check the processed level of the returned BIR.
      If any of the intermediate operations fails, an UNDECIDED conformity response is
                  issued.
   </description>

   <!-- UUID of the BSP under test -->
   <input name="_bspUuid"/>

   <!-- Timeout for the BioAPI_NOTIFY_INSERT event -->
   <input name="_inserttimeout"/>

   <!-- Indicates whether the BSP under test does not claim support for the
                  BioAPI_NOTIFY_SOURCE_PRESENT event notification -->
   <input name="_noSourcePresentSupported" />

   <!-- Timeout for the BioAPI_NOTIFY_SOURCE_PRESENT event -->
   <input name="_sourcepresenttimeout"/>

   <!-- Timeout for BioSPI_Capture -->
   <input name="_capturetimeout"/>

   <!-- Invocation of the primary activity of this assertion with input parameter values
                  assigned from the assertion's parameters. -->
   <invoke activity="CapturedBIR_Datatype">
      <input name="bspUuid" var="_bspUuid"/>
      <input name="inserttimeouttime" var="_inserttimeout"/>
      <input name="nosourcepresentsupported"
                  var="_noSourcePresentSupported" />
      <input name="sourcepresenttimeouttime"
                  var="_sourcepresenttimeout"/>
      <input name="capturetimeouttime" var="_capturetimeout"/>
   </invoke>

   <!-- Activity bound to a function of the framework callback interface exposed by the
                  testing component.  This activity will be automatically invoked on each
                  incoming call to the function to which it is bound. -->
   <bind activity="EventHandler"
            package="02c59458-0c46-1085-95d7-0002a5d5fd2e"
            function="BioSPI_EventHandler"/>
</assertion>

<activity name="CapturedBIR_Datatype">
   <input name="bspUuid"/>
   <input name="inserttimeouttime"/>
   <input name="nosourcepresentsupported" />
   <input name="sourcepresenttimeouttime"/>
   <input name="capturetimeouttime"/>

   <!-- This assertion will use BSPHandle "1" for all BioSPI calls that require it -->
   <set name="_bsphandle" value="1"/>

   <!-- Invoke the functions BioSPI_BSPLoad and BioSPI_BSPAttach exposed by the BSP under
                  test.
            The input value for the parameter "unitIDOrNull" is "0", therefore the
                  assertion will test a sensor unit chosen by the BSP. -->
   <invoke activity="LoadAndAttach"
            package="02c59458-0c46-1085-95d7-0002a5d5fd2e"
            break_on_break="true">
      <input name="bspUuid" var="bspUuid"/>
      <input name="bspVersion" value="32"/>
      <input name="unitIDOrNull" value="0"/>
      <input name="bspHandle" var="_bsphandle"/>
      <input name="eventtimeouttime" var="inserttimeouttime"/>
   </invoke>

   <set name="eventtimeoutflag" value="false"/>

   <!-- If the BSP under test claims support for the BioAPI_NOTIFY_SOURCE_PRESENT event
                  notification, wait until that notification has been received, but no
                  longer than the specified maximum duration.-->
   <wait_until timeout_var="sourcepresenttimeouttime"
            setvar="eventtimeoutflag">
      <or var1="nosourcepresentsupported" var2="_sourcePresent" />
   </wait_until>

   <!-- Issue a conformity response.
```

```
            If the condition specified in the <description> below is false, an UNDECIDED
                  conformity response is issued and the execution of the activity is
                  interrupted, otherwise a PASS conformity response is issued.-->
<assert_condition response_if_false="undecided"
            break_if_false="true">
      <description>
            Either the BSP under test does not claim support for the
                  BioAPI_NOTIFY_SOURCE_PRESENT event notification, or the event
                  notification has been received within the specified maximum duration.
      </description>
      <not var="eventtimeoutflag"/>
</assert_condition>

<!-- The BSP is ready to capture. Invoke the function BioSPI_Capture for the purpose of
                  creating a template. The handle of the captured BIR is stored in the
                  variable "capturedbir". -->
<invoke function="BioSPI_Capture">
      <input name="BSPHandle" var="_bsphandle"/>
      <input name="Purpose"
                  var="__BioAPI_PURPOSE_ENROLL_FOR_VERIFICATION_ONLY"/>
      <input name="Subtype" value="0" />
      <input name="Timeout" var="capturetimeouttime"/>
      <input name="no_AuditData" value="true"/>
      <output name="CapturedBIR" setvar="capturedbir_handle"/>
      <return setvar="return"/>
</invoke>

<!-- Issue a conformity response.
            If the condition specified in the <description> below is false, an UNDECIDED
                  conformity response is issued and the execution of the activity is
                  interrupted, otherwise a PASS conformity response is issued.-->
<assert_condition response_if_false="undecided"
            break_if_false="true">
      <description>
            The function BioSPI_Capture has returned BioAPI_OK
      </description>
      <equal_to var1="return" var2="__BioAPI_OK"/>
</assert_condition>

<!-- Invoke the function BioSPI_GetHeaderFromHandle. -->
<invoke function="BioSPI_GetHeaderFromHandle">
      <input name="BSPHandle" var="_bsphandle"/>
      <input name="Handle" var="capturedbir_handle"/>
      <output name="ProcessedLevel" setvar="processedLevel"/>
      <return setvar="return"/>
</invoke>

<!-- Issue a conformity response.
            If the condition specified in the <description> below is false, an UNDECIDED
                  conformity response is issued and the execution of the activity is
                  interrupted, otherwise a PASS conformity response is issued.-->
<assert_condition response_if_false="undecided"
            break_if_false="true">
      <description>
                  The function BioSPI_GetHeaderFromHandle has returned BioAPI_OK.
      </description>
      <equal_to var1="return" var2="__BioAPI_OK"/>
</assert_condition>

<!-- Issue a conformity response.
            If the condition specified in the <description> below is false, a FAIL
                  conformity response is issued, otherwise a PASS conformity response is
                  issued.-->
<assert_condition>
      <description>
            The processed level of the returned BIR is either INTERMEDIATE or PROCESSED.
      </description>
      <or>
            <equal_to var1="processedLevel"
                        var2="__BioAPI_BIR_DATA_TYPE_INTERMEDIATE"/>
            <equal_to var1="processedLevel"
                        var2="__BioAPI_BIR_DATA_TYPE_PROCESSED"/>
      </or>
</assert_condition>
```

```
            <!-- Invoke the functions BioSPI_BSPDetach and BioSPI_BSPUnload -->
            <invoke activity="DetachAndUnload"
                        package="02c59458-0c46-1085-95d7-0002a5d5fd2e" >
                  <input name="bspUuid" var="bspUuid" />
                  <input name="BSPHandle" var="_bsphandle" />
            </invoke>
      </activity>

</package>
```

8.31 Assertion 9d - *BioSPI_Capture_InvalidBSPHandle*

Description: This assertion checks if calling BioSPI_Capture with an invalid BSP handle returns an error BioAPIERR_INVALID_BSP_HANDLE.

Excerpts

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);*

Subclause 8.4.1.2

Parameters: BSPHandle (input) - The handle of the attached biometric service provider.

References: 9.3.4.1 and 8.4.1.2

Scenario:

1) Load the BSP under test

2) Attach the BSP under test

3) Call BioSPI_Capture with an invalid BSP handle.

4) Check the return code. It is expected to be BioAPIERR_INVALID_BSP_HANDLE.

5) Detach and unload the BSP under test.

If any of the intermediate operations fails, an UNDECIDED conformity response is issued.

Expected results: The call to BioSPI_Capture returns BioAPIERR_INVALID_BSP_HANDLE

Assertion language package

```
<package name="0244f2a8-0cf2-1085-8443-0002a5d5fd2e">
      <author>
            ISO/IEC JTC1 SC37
      </author>

      <description>
            This package contains the assertion "BioSPI_Capture_InvalidBSPHandle" (see the
                              "description" element of the assertion below).
      </description>

      <assertion name="BioSPI_Capture_InvalidBSPHandle" model="BSPTesting">
            <description>
                  This assertion checks if calling BioSPI_Capture with an invalid BSP handle returns
                              an error BioAPIERR_INVALID_BSP_HANDLE.
                  The relevant text in BioAPI 2.0 is quoted below from 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);

                  NOTE: Details of the function definition are located in clause 8.4.1,
                              BioAPI_Capture.
                  ________________________________________________________
                  Subclause 8.4.1.2:
                  Parameters: BSPHandle (input) - The handle of the attached biometric service
                              provider.
                  ________________________________________________________

                  In order to determine conformance with respect to the text above, the following
                              steps are performed:

                        1)    Load the BSP under test
                        2)    Attach the BSP under test
                        3)    Call BioSPI_Capture with an invalid BSP handle.
                        4)    Check the return code.  It is expected to be
                              BioAPIERR_INVALID_BSP_HANDLE.
                        5)    Detach and unload the BSP under test.

                  If any of the intermediate operations fails, an UNDECIDED conformity response is
                              issued.
            </description>

            <!-- UUID of the BSP under test -->
            <input name="_bspUuid"/>

            <!-- Timeout for the BioAPI_NOTIFY_INSERT event -->
            <input name="_inserttimeout"/>

            <!-- Indicates whether the BSP under test does not claim support for the
                              BioAPI_NOTIFY_SOURCE_PRESENT event notification -->
            <input name="_noSourcePresentSupported" />

            <!-- Timeout for the BioAPI_NOTIFY_SOURCE_PRESENT event -->
            <input name="_sourcepresenttimeout"/>

            <!-- Timeout for BioSPI_Capture -->
            <input name="_capturetimeout"/>

            <!-- Invocation of the primary activity of this assertion with input parameter values
                              assigned from the assertion's parameters. -->
            <invoke activity="BioSPI_Capture_InvalidBSPHandle">
                  <input name="bspUuid" var="_bspUuid"/>
                  <input name="inserttimeouttime" var="_inserttimeout"/>
                  <input name="nosourcepresentsupported"
                              var="_noSourcePresentSupported" />
                  <input name="sourcepresenttimeouttime"
                              var="_sourcepresenttimeout"/>
                  <input name="capturetimeouttime" var="_capturetimeout"/>
            </invoke>
      <!-- Activity bound to a function of the framework callback interface exposed by the
                  testing component.  This activity will be automatically invoked on each
                  incoming call to the function to which it is bound. -->
      <bind activity="EventHandler"
            package="02c59458-0c46-1085-95d7-0002a5d5fd2e"
            function="BioSPI_EventHandler"/>
</assertion>

<activity name="BioSPI_Capture_InvalidBSPHandle">
      <input name="bspUuid"/>
      <input name="inserttimeouttime"/>
      <input name="nosourcepresentsupported" />
      <input name="sourcepresenttimeouttime"/>
      <input name="capturetimeouttime"/>

      <!-- This assertion will use ModuleHandle "1" for all BioSPI calls that require it -->
      <set name="_bsphandle" value="1"/>

      <!-- Invoke the functions BioSPI_BSPLoad and BioSPI_BSPAttach exposed by the BSP under
                  test.
            The input value for the parameter "unitIDOrNull" is "0", therefore the
                  assertion will test a sensor unit chosen by the BSP. -->
      <invoke activity="LoadAndAttach"
            package="02c59458-0c46-1085-95d7-0002a5d5fd2e"
            break_on_break="true">
            <input name="bspUuid" var="bspUuid"/>
            <input name="bspVersion" value="32"/>
            <input name="unitIDOrNull" value="0"/>
            <input name="bspHandle" var="_bsphandle"/>
            <input name="eventtimeouttime" var="inserttimeouttime"/>
      </invoke>

      <set name="eventtimeoutflag" value="false"/>

      <!-- If the BSP under test claims support for the BioAPI_NOTIFY_SOURCE_PRESENT event
                  notification, wait until that notification has been received, but no
                  longer than the specified maximum duration.-->
      <wait_until timeout_var="sourcepresenttimeouttime"
            setvar="eventtimeoutflag">
            <or var1="nosourcepresentsupported" var2="_sourcePresent" />
      </wait_until>

      <!-- Issue a conformity response.
            If the condition specified in the <description> below is false, an UNDECIDED
                  conformity response is issued and the execution of the activity is
                  interrupted, otherwise a PASS conformity response is issued.-->
      <assert_condition response_if_false="undecided"
            break_if_false="true">
            <description>
            Either the BSP under test does not claim support for the
                  BioAPI_NOTIFY_SOURCE_PRESENT event notification, or the event
                  notification has been received within the specified maximum duration.
            </description>
            <not var="eventtimeoutflag"/>
      </assert_condition>

      <!-- The BSP is ready to capture. Invoke the function BioSPI_Capture for the purpose of
                  creating a template. The handle of the captured BIR is stored in the
                  variable "capturedbir". -->
      <invoke function="BioSPI_Capture">
            <input name="BSPHandle" value="0"/>
            <input name="Purpose"
                  var="__BioAPI_PURPOSE_ENROLL_FOR_VERIFICATION_ONLY"/>
            <input name="Subtype" value="0" />
            <input name="Timeout" var="capturetimeouttime"/>
            <input name="no_AuditData" value="true"/>
            <output name="CapturedBIR" setvar="capturedbir_handle"/>
            <return setvar="return"/>
      </invoke>

      <!-- Issue a conformity response.
            If the condition specified in the <description> below is false, a FAIL
                  conformity response is issued, otherwise a PASS conformity response is
                  issued.-->
      <assert_condition>
            <description>
```

```
                        The function BioSPI_Capture has returned BioAPIERR_INVALID_BSP_HANDLE
                  </description>
                  <equal_to var1="return"
                              var2="__BioAPIERR_BSP_INVALID_BSP_HANDLE"/>
            </assert_condition>

            <!-- Invoke the functions BioSPI_BSPDetach and BioSPI_BSPUnload -->
            <invoke activity="DetachAndUnload"
                        package="02c59458-0c46-1085-95d7-0002a5d5fd2e" >
                  <input name="bspUuid" var="bspUuid" />
                  <input name="BSPHandle" var="_bsphandle" />
            </invoke>
      </activity>

</package>
```

8.32 Assertion 10a - *BioSPI_CreateTemplate_PayloadSupported*

Description: This assertion tests BioSPI_CreateTemplate with valid parameters and payload.

Excerpts

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);*

Subclause 8.4.5.1

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.

Subclause 7.47

#define BioAPI_PAYLOAD (0x00000080)

If set, indicates that the BSP supports payload carry (accepts payload during enroll/process and returns payroll upon successful verify).

Subclause A.4.6.2.6

If supported, BSPs shall post to the component registry the maximum payload size it can accommodate. A maximum size of '0' indicates payload carry is not supported. If input payloads exceed this size, an error shall be generated.

References: 9.3.4.2, 8.4.5.1, 7.47, and A.4.6.2.6

Scenario:

1) Load the BSP under test

2) Attach the BSP under test

3) Call BioSPI_Capture with a purpose of BioAPI_PURPOSE_ENROLL_FOR_VERIFICATION_ONLY to obtain a BIR for enrollment.

4) Call BioSPI_CreateTemplate with a specified Payload.

5) Check the return code, which is expected to be BioAPI_OK.

6) Detach and unload the BSP under test.

If any of the intermediate operations fails, an UNDECIDED conformity response is issued.

Expected results: The call to BioSPI_CreateTemplate returns BioAPI_OK.

Assertion language package

```
<package name="04a01118-0cf9-1085-96d4-0002a5d5fd2e">
      <author>
            ISO/IEC JTC1 SC37
      </author>

      <description>
            This package contains the assertion "BioSPI_CreateTemplate_PayloadSupported" (see the
                              "description" element of the assertion below).
      </description>

      <assertion name="BioSPI_CreateTemplate_PayloadSupported" model="BSPTesting">
            <description>
                  This assertion tests BioSPI_CreateTemplate with valid parameters and payload.
                  The relevant text in BioAPI 2.0 is quoted below from subclauses  9.3.4.2,
                              A.4.6.2.6, C.3.4.4, and 7.47.
                  ___________________________________________________________
                  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);

                        NOTE: Details of the function definition are located in clause 8.4.2,
                              BioAPI_CreateTemplate.
                  ___________________________________________________________
                  Subclause A.4.6.2.6
                  If supported, BSPs shall post to the component registry the maximum payload size it
                              can accommodate.  A maximum size of '0' indicates payload carry is not
                              supported.  If input payloads exceed this size, an error shall be
                              generated.
                  ___________________________________________________________
                  Subclause 7.47:
                  #define BioAPI_PAYLOAD (0x00000080)
                  If set, indicates that the BSP supports payload carry (accepts payload during
                              enroll/process and returns payroll upon successful verify).

                  In order to determine conformance with respect to the text above, the following
                              steps are performed:

                        1)    Load the BSP under test
                        2)    Attach the BSP under test
                        3)    Call BioSPI_Capture with a purpose of
                              BioAPI_PURPOSE_ENROLL_FOR_VERIFICATION_ONLY to obtain a BIR for
                              enrollment.
                        4)    Call BioSPI_CreateTemplate with a specified Payload.
                        5)    Check the return code, which is expected to be BioAPI_OK.
                        6)    Detach and unload the BSP under test.
            If any of the intermediate operations fails, an UNDECIDED conformity response is
                        issued.
        </description>

        <!-- UUID of the BSP under test -->
        <input name="_bspUuid"/>

        <!-- Timeout for the BioAPI_NOTIFY_INSERT event -->
        <input name="_inserttimeout"/>

        <!-- Indicates whether the BSP under test does not claim support for the
                        BioAPI_NOTIFY_SOURCE_PRESENT event notification -->
        <input name="_noSourcePresentSupported" />

        <!-- Timeout for the BioAPI_NOTIFY_SOURCE_PRESENT event -->
        <input name="_sourcepresenttimeout"/>

        <!-- Timeout for BioSPI_Capture -->
        <input name="_capturetimeout"/>

        <!-- Indicates whether the BSP under test claims support for payload -->
        <input name="_supportPayload" />

        <!-- Payload must satisfy the constraint on the payload size posted by the BSP in the
                        component registry -->
        <input name="_payload" />

        <!-- Invocation of the primary activity of this assertion with input parameter values
                        assigned from the assertion's parameters. -->
        <invoke activity="BioSPI_CreateTemplate_PayloadSupported">
              <input name="bspUuid" var="_bspUuid"/>
              <input name="inserttimeouttime" var="_inserttimeout"/>
              <input name="nosourcepresentsupported"
                        var="_noSourcePresentSupported" />
              <input name="sourcepresenttimeouttime"
                        var="_sourcepresenttimeout"/>
              <input name="capturetimeouttime" var="_capturetimeout"/>
              <input name="supportpayload"  var="_supportPayload" />
              <input name="payload" var="_payload" />
        </invoke>

        <!-- Activity bound to a function of the framework callback interface exposed by the
                        testing component.  This activity will be automatically invoked on each
                        incoming call to the function to which it is bound. -->
        <bind activity="EventHandler"
                  package="02c59458-0c46-1085-95d7-0002a5d5fd2e"
                  function="BioSPI_EventHandler"/>
  </assertion>

  <activity name="BioSPI_CreateTemplate_PayloadSupported">
        <input name="bspUuid"/>
        <input name="inserttimeouttime"/>
        <input name="nosourcepresentsupported" />
        <input name="sourcepresenttimeouttime"/>
        <input name="capturetimeouttime"/>
        <input name="supportpayload" />
        <input name="payload" />

        <!-- This assertion will use ModuleHandle "1" for all BioSPI calls that require it -->
        <set name="_bsphandle" value="1"/>

        <!-- Invoke the functions BioSPI_BSPLoad and BioSPI_BSPAttach exposed by the BSP under
                        test.
                  The input value for the parameter "unitIDOrNull" is "0", therefore the
                        assertion will test a sensor unit chosen by the BSP. -->
        <invoke activity="LoadAndAttach"
                  package="02c59458-0c46-1085-95d7-0002a5d5fd2e"
                  break_on_break="true">
              <input name="bspUuid" var="bspUuid"/>
              <input name="bspVersion" value="32"/>
              <input name="unitIDOrNull" value="0"/>
              <input name="bspHandle" var="_bsphandle"/>
              <input name="eventtimeouttime" var="inserttimeouttime"/>
        </invoke>

        <set name="eventtimeoutflag" value="false"/>
```

```
<!-- If the BSP under test claims support for the BioAPI_NOTIFY_SOURCE_PRESENT event
                  notification, wait until that notification has been received, but no
                  longer than the specified maximum duration.-->
<wait_until timeout_var="sourcepresenttimeouttime"
            setvar="eventtimeoutflag">
      <or var1="nosourcepresentsupported" var2="_sourcePresent" />
</wait_until>

<!-- Issue a conformity response.
            If the condition specified in the <description> below is false, an UNDECIDED
                  conformity response is issued and the execution of the activity is
                  interrupted, otherwise a PASS conformity response is issued.-->
<assert_condition response_if_false="undecided"
            break_if_false="true">
      <description>
            Either the BSP under test does not claim support for the
                  BioAPI_NOTIFY_SOURCE_PRESENT event notification, or the event
                  notification has been received within the specified maximum duration.
      </description>
      <not var="eventtimeoutflag"/>
</assert_condition>

<!-- The BSP is ready to capture. Invoke the function BioSPI_Capture for the purpose of
                  creating a template. The handle of the captured BIR is stored in the
                  variable "capturedbir". -->
<invoke function="BioSPI_Capture">
      <input name="BSPHandle" var="_bsphandle"/>
      <input name="Purpose"
                  var="__BioAPI_PURPOSE_ENROLL_FOR_VERIFICATION_ONLY"/>
      <input name="Subtype" value="0"/>
      <input name="Timeout" var="capturetimeouttime"/>
      <input name="no_AuditData" value="true"/>
      <output name="CapturedBIR" setvar="capturedbir_handle"/>
      <return setvar="return"/>
</invoke>

<!-- Issue a conformity response.
            If the condition specified in the <description> below is false, an UNDECIDED
                  conformity response is issued and the execution of the activity is
                  interrupted, otherwise a PASS conformity response is issued.-->
<assert_condition response_if_false="undecided"
            break_if_false="true">
      <description>
            The function BioSPI_Capture has returned BioAPI_OK.
      </description>
      <equal_to var1="return" var2="__BioAPI_OK"/>
</assert_condition>

<!-- Check if the processed level of the captured BIR is PROCESSED.  If it is is
                  PROCESSED, then an UNDECIDED conformity response is issued and the
                  activity is interrupted. -->
<invoke activity="check_capturedBIR_datatype"
            package="02c59458-0c46-1085-95d7-0002a5d5fd2e"
            break_on_break="true">
      <input name="BSPHandle" var="_bsphandle"/>
      <input name="BIRHandle" var="capturedbir_handle"/>
</invoke>

<!-- Invoke the function BioSPI_CreateTemplate. -->
<invoke function="BioSPI_CreateTemplate">
      <input name="BSPHandle" var="_bsphandle"/>
      <input name="CapturedBIR_Form"
                  var="__BioAPI_BIR_HANDLE_INPUT" />
      <input name="CapturedBIR_BIRHandle"
                  var="capturedbir_handle"/>
      <input name="Payload" var="payload" />
      <output name="NewTemplate" setvar="newTemplate_handle"/>
      <return setvar="return"/>
</invoke>

<!-- Issue a conformity response.
            If the condition specified in the <description> below is false, a FAIL
                  conformity response is issued, otherwise a PASS conformity response is
                  issued.-->
<assert_condition>
      <description>
```

```
                        The function BioSPI_CreateTemplate has returned a proper value based on
                              whether the BSP supports a payload or not.
                  </description>
                  <or>
                        <and>
                              <equal_to var1="return" var2="__BioAPI_OK"/>
                              <same_as var1="supportpayload" value2="true"/>
                        </and>
                        <and>
                              <equal_to var1="return"
                                          var2="__BioAPIERR_BSP_UNABLE_TO_STORE_PAYLOAD" />
                              <same_as var1="supportpayload" value2="false"/>
                        </and>
                  </or>
            </assert_condition>

            <!-- Invoke the functions BioSPI_BSPDetach and BioSPI_BSPUnload -->
            <invoke activity="DetachAndUnload"
                        package="02c59458-0c46-1085-95d7-0002a5d5fd2e" >
                  <input name="bspUuid" var="bspUuid" />
                  <input name="BSPHandle" var="_bsphandle" />
            </invoke>
      </activity>

</package>
```

8.33 Assertion 10b - *BioSPI_CreateTemplate_BIRHeaderQuality*

Description: This assertion tests the function BioSPI_CreateTemplate with valid parameters and the returned quality value.

Excerpts

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);*

Subclause 7.49.3

Quality measurements are reported as an integral value in the range 0-100 except as follows:

Value of -1: BioAPI_QUALITY was not set by the BSP (reference BSP vendor's documentation for explanation).

Value of -2: BioAPI_QUALITY is not supported by the BSP.

Subclause 7.47

#define BioAPI_QUALITY_PROCESSED (0x00000008)

If set, BSP supports the return of quality value (in the BIR header) for processed biometric data.

Subclause A.4.6.2.2

Similarly, when a BIR is processed, another quality calculation may be performed and the quality value included in the header of the processedBIR (and the optional AdaptedBIR).

This would occur during BioAPI_CreateTemplate, BioAPI_Process, BioAPI_Verify, BioAPI_VerifyMatch, BioAPI_Enroll, and BioAPI_Import (ConstructedBIR) operations.

The BSP must post to the module registry whether or not it supports the calculation of quality measurements for each type of BIR - raw, intermediate, and processed.

References: 9.3.4.2, 7.49.3, 7.47, and A.4.6.2.2

Scenario:

1) Load the BSP under test

2) Attach the BSP under test

3) Call the function BioSPI_Capture to obtain a BIR

4) Call the function BioSPI_CreateTemplate.

5) Call BioSPI_GetHeaderFromHandle for the NewTemplate, check the Quality field, which is expected to be in the range 0-100.

6) Detach and Unload the BSP under test.

If any of the intermediate operations fails, an UNDECIDED conformity response is issued.

Expected results: The header of the captured BIR contains a valid quality value (in the range 0-100).

Assertion language package

```
<package name="00b5c728-0cfb-1085-8969-0002a5d5fd2e">
      <author>
            ISO/IEC JTC1 SC37
      </author>

      <description>
            This package contains the assertion "BioSPI_CreateTemplate_BIRHeaderQuality" (see the
                        "description" element of the assertion below).
      </description>

      <assertion name="BioSPI_CreateTemplate_BIRHeaderQuality" model="BSPTesting">
            <description>
                  This assertion tests the function BioSPI_CreateTemplate with valid parameters and
                        checks the returned quality value.
                  The relevant text in BioAPI 2.0 is quoted below from subclauses 9.3.4.2, 7.49.3,
                        7.47 and A.4.6.2.2.
                  _______________________________________________________
                  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);

                        NOTE: Details of the function definition are located in clause 8.4.2,
                              BioAPI_CreateTemplate.
      ________________________________________________________
      Subclause 7.49.3:
      Quality measurements are reported as an integral value in the range 0-100 except as
                  follows:
      Value of -1: BioAPI_QUALITY was not set by the BSP (reference BSP vendor's
                  documentation for explanation).
      Value of -2: BioAPI_QUALITY is not supported by the BSP.
      _______________________________________________________
      Subclause 7.47:
      #define BioAPI_QUALITY_PROCESSED (0x00000008)
      If set, BSP supports the return of quality value (in the BIR header) for processed
                  biometric data.
      Subclause A.4.6.2.2:
      Similarly, when a BIR is processed, another quality calculation may be performed
                  and the quality value included in the header of the processedBIR (and
                  the optional AdaptedBIR).
      This would occur during  BioAPI_CreateTemplate, BioAPI_Process, BioAPI_Verify,
                  BioAPI_VerifyMatch, BioAPI_Enroll, and BioAPI_Import (ConstructedBIR)
                  operations.
      The BSP must post to the module registry whether or not it supports the calculation
                  of quality measurements for each type of BIR - raw, intermediate, and
                  processed.
      _______________________________________________________

      In order to determine conformance with respect to the text above, the following
                  steps are performed:

            1)    Load the BSP under test
            2)    Attach the BSP under test
            3)    Call the function BioSPI_Capture to obtain a BIR
            4)    Call the function BioSPI_CreateTemplate.
            5)    Call BioSPI_GetHeaderFromHandle for the NewTemplate, check the Quality
                  field, which is expected to be in the range 0-100.
            6)    Detach and Unload the BSP under test.

      If any of the intermediate operations fails, an UNDECIDED conformity response is
                  issued.
</description>

<!-- UUID of the BSP under test -->
<input name="_bspUuid"/>

<!-- Timeout for the BioAPI_NOTIFY_INSERT event -->
<input name="_inserttimeout"/>

<!-- Indicates whether the BSP under test does not claim support for the
                  BioAPI_NOTIFY_SOURCE_PRESENT event notification -->
<input name="_noSourcePresentSupported"/>

<!-- Timeout for the BioAPI_NOTIFY_SOURCE_PRESENT event -->
<input name="_sourcepresenttimeout"/>

<!-- Timeout for BioSPI_Capture -->
<input name="_capturetimeout"/>

<!-- Indicates whether the BSP under test claims support for return of quality in a
                  processed BIR -->
<input name="_processedQualitySupported"/>

<!-- Invocation of the primary activity of this assertion with input parameter values
                  assigned from the assertion's parameters. -->
<invoke activity="BioSPI_CreateTemplate_ReturnQuality">
      <input name="bspUuid" var="_bspUuid"/>
      <input name="inserttimeouttime" var="_inserttimeout"/>
      <input name="nosourcepresentsupported"
                  var="_noSourcePresentSupported"/>
      <input name="sourcepresenttimeouttime"
                  var="_sourcepresenttimeout"/>
      <input name="capturetimeouttime" var="_capturetimeout"/>
      <input name="processedQualitySupported"
                  var="_processedQualitySupported"/>
</invoke>
```

```
      <!-- Activity bound to a function of the framework callback interface exposed by the
                  testing component.  This activity will be automatically invoked on each
                  incoming call to the function to which it is bound. -->
      <bind activity="EventHandler"
            package="02c59458-0c46-1085-95d7-0002a5d5fd2e"
            function="BioSPI_EventHandler"/>
</assertion>

<activity name="BioSPI_CreateTemplate_ReturnQuality">
      <input name="bspUuid"/>
      <input name="inserttimeouttime"/>
      <input name="nosourcepresentsupported"/>
      <input name="sourcepresenttimeouttime"/>
      <input name="capturetimeouttime"/>
      <input name="processedQualitySupported"/>

      <!-- This assertion will use BSPHandle "1" for all BioSPI calls that require it -->
      <set name="_bsphandle" value="1"/>

      <!-- Invoke the functions BioSPI_BSPLoad and BioSPI_BSPAttach exposed by the BSP under
                  test.
            The input value for the parameter "unitIDOrNull" is "0", therefore the
                  assertion will test a sensor unit chosen by the BSP. -->
      <invoke activity="LoadAndAttach"
            package="02c59458-0c46-1085-95d7-0002a5d5fd2e"
            break_on_break="true">
            <input name="bspUuid" var="bspUuid"/>
            <input name="bspVersion" value="32"/>
            <input name="unitIDOrNull" value="0"/>
            <input name="bspHandle" var="_bsphandle"/>
            <input name="eventtimeouttime" var="inserttimeouttime"/>
      </invoke>

      <set name="eventtimeoutflag" value="false"/>

      <!-- If the BSP under test claims support for the BioAPI_NOTIFY_SOURCE_PRESENT event
                  notification, wait until that notification has been received, but no
                  longer than the specified maximum duration.-->
      <wait_until timeout_var="sourcepresenttimeouttime"
            setvar="eventtimeoutflag">
            <or var1="nosourcepresentsupported" var2="_sourcePresent"/>
      </wait_until>

      <!-- Issue a conformity response.
            If the condition specified in the <description> below is false, an UNDECIDED
                  conformity response is issued and the execution of the activity is
                  interrupted, otherwise a PASS conformity response is issued.-->
      <assert_condition response_if_false="undecided"
            break_if_false="true">
            <description>
            Either the BSP under test does not claim support for the
                  BioAPI_NOTIFY_SOURCE_PRESENT event notification, or the event
                  notification has been received within the specified maximum duration.
            </description>
            <not var="eventtimeoutflag"/>
      </assert_condition>

      <!-- The BSP is ready to capture. Invoke the function BioSPI_Capture for the purpose of
                  creating a template. The handle of the captured BIR is stored in the
                  variable "capturedbir". -->
      <invoke function="BioSPI_Capture">
            <input name="BSPHandle" var="_bsphandle"/>
            <input name="Purpose"
                  var="__BioAPI_PURPOSE_ENROLL_FOR_VERIFICATION_ONLY"/>
            <input name="Subtype" value="0"/>
            <input name="Timeout" var="capturetimeouttime"/>
            <input name="no_AuditData" value="true"/>
            <output name="CapturedBIR" setvar="capturedbir_handle"/>
            <return setvar="return"/>
      </invoke>

      <!-- Issue a conformity response.
            If the condition specified in the <description> below is false, an UNDECIDED
                  conformity response is issued and the execution of the activity is
                  interrupted, otherwise a PASS conformity response is issued.-->
      <assert_condition response_if_false="undecided"
```

```
                break_if_false="true">
            <description>
                The function BioSPI_Capture has returned BioAPI_OK.
            </description>
            <equal_to var1="return" var2="__BioAPI_OK"/>
        </assert_condition>

        <!-- Check if the captured BIR's processed level is PROCESSED.  If it is PROCESSED, then
                        an UNDECIDED conformity response is issued and the execution of the
                        activity is interrupted. -->
        <invoke activity="check_capturedBIR_datatype"
                package="02c59458-0c46-1085-95d7-0002a5d5fd2e"
                break_on_break="true" >
            <input name="BSPHandle" var="_bsphandle"/>
            <input name="BIRHandle"  var="capturedbir_handle"/>
        </invoke>

        <!-- Invoke the function BioSPI_CreateTemplate. -->
        <invoke function="BioSPI_CreateTemplate">
            <input name="BSPHandle" var="_bsphandle"/>
            <input name="CapturedBIR_Form"
                        var="__BioAPI_BIR_HANDLE_INPUT"/>
            <input name="CapturedBIR_BIRHandle"
                        var="capturedbir_handle"/>
            <output name="NewTemplate" setvar="newTemplate_handle"/>
            <return setvar="return"/>
        </invoke>

        <!-- Issue a conformity response.
                If the condition specified in the <description> below is false, an UNDECIDED
                        conformity response is issued and execution of the activity is
                        interrupted, otherwise a PASS conformity response is issued.-->
        <assert_condition response_if_false="undecided"
                break_if_false="true">
            <description>
                The function BioSPI_CreateTemplate has returned BioAPI_OK.
            </description>
            <equal_to var1="return" var2="__BioAPI_OK"/>
        </assert_condition>

        <!-- Invoke the function BioSPI_GetHeaderFromHandle on the newly created template BIR. --
                        >
        <invoke function="BioSPI_GetHeaderFromHandle">
            <input name="BSPHandle" var="_bsphandle"/>
            <input name="Handle" var="newTemplate_handle"/>
            <output name="Quality" setvar="quality"/>
            <return setvar="return"/>
        </invoke>

        <!-- Issue a conformity response.
                If the condition specified in the <description> below is false, an UNDECIDED
                        conformity response is issued and the execution of the activity is
                        interrupted, otherwise a PASS conformity response is issued.-->
        <assert_condition response_if_false="undecided"
                break_if_false="true">
            <description>
                The function BioSPI_GetHeaderFromHandle has returned BioAPI_OK.
            </description>
        <equal_to var1="return" var2="__BioAPI_OK"/>
    </assert_condition>

        <invoke activity="check_quality_supported"
                package="02c59458-0c46-1085-95d7-0002a5d5fd2e"
                break_on_break="true">
            <only_if>
                <same_as var1="processedQualitySupported" value2="true"/>
            </only_if>
            <input name="quality" var="quality"/>
        </invoke>

        <invoke activity="check_quality_not_supported"
                package="02c59458-0c46-1085-95d7-0002a5d5fd2e"
                break_on_break="true">
            <only_if>
                <same_as var1="processedQualitySupported"
                            value2="false"/>
            </only_if>
            <input name="quality" var="quality"/>
        </invoke>

        <!-- Invoke the functions BioSPI_BSPDetach and BioSPI_BSPUnload -->
        <invoke activity="DetachAndUnload"
                package="02c59458-0c46-1085-95d7-0002a5d5fd2e" >
            <input name="bspUuid" var="bspUuid"/>
            <input name="BSPHandle" var="_bsphandle"/>
        </invoke>
    </activity>

</package>
```

8.34 Assertion 10c - *BioSPI_CreateTemplate_OutputBIRDataType*

Description: This assertion tests BioSPI_CreateTemplate with valid parameters. The new template BIR is expected to have the processed level PROCESSED.

Excerpts

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);*

Subclause 8.4.2.1

This function takes a BIR containing biometric data in intermediate form for the purpose of creating a new enrollment template.

References: 9.3.4.2 and 8.4.2.1.

Scenario:

1) Load the BSP under test

2) Attach the BSP under test

3) Call BioSPI_Capture with a purpose of BioAPI_PURPOSE_ENROLL_FOR_VERIFICATION_ONLY to obtain a BIR for enrollment.

4) Call BioSPI_CreateTemplate without StoredTemplate and with Payload set to 0

5) Check the return code, which is expected to be BioAPI_OK.

6) Call BioSPI_GetHeaderFromHandle on the new template BIR, expecting the processed level to be PROCESSED.

If any of the intermediate operations fails, an UNDECIDED conformity response is issued.

Expected results: The processed level of the new template BIR is PROCESSED.

Assertion language package

```
<package name="0193c730-0cf9-1085-b0a3-0002a5d5fd2e">
      <author>
            ISO/IEC JTC1 SC37
      </author>

      <description>
            This package contains the assertion "BioSPI_CreateTemplate_OutputBIRDataType" (see the
                        "description" element of the assertion below).
      </description>

      <assertion name="BioSPI_CreateTemplate_OutputBIRDataType" model="BSPTesting">
            <description>
                  This assertion tests BioSPI_CreateTemplate with valid parameters.  The new template
                        BIR is expected to have the processed level PROCESSED.
                  The relevant text in BioAPI 2.0 is quoted below from subclause 9.3.4.2 and 8.4.2.1.
                  _________________________________________________________
                  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);

                        NOTE: Details of the function definition are located in clause 8.4.2,
                              BioAPI_CreateTemplate.
                  _________________________________________________________
                  Subclause 8.4.2.1:
                  This function takes a BIR containing biometric data in intermediate form for the
                        purpose of creating a new enrollment template.
                  _________________________________________________________

                  In order to determine conformance with respect to the text above, the following
                        steps are performed:

                        1)    Load the BSP under test
                        2)    Attach the BSP under test
                        3)    Call BioSPI_Capture with a purpose of
                              BioAPI_PURPOSE_ENROLL_FOR_VERIFICATION_ONLY to obtain a BIR for
                              enrollment.
                        4)    Call BioSPI_CreateTemplate without StoredTemplate and with Payload set
                              to 0
                        5)    Check the return code, which is expected to be BioAPI_OK.
                        6)    Call BioSPI_GetHeaderFromHandle on the new template BIR, expecting the
                              processed level to be PROCESSED.

                  If any of the intermediate operations fails, an UNDECIDED conformity response is
                        issued.
            </description>

            <!-- UUID of the BSP under test -->
            <input name="_bspUuid"/>

            <!-- Timeout for the BioAPI_NOTIFY_INSERT event -->
            <input name="_inserttimeout"/>

            <!-- Indicates whether the BSP under test does not claim support for the
                        BioAPI_NOTIFY_SOURCE_PRESENT event notification -->
            <input name="_noSourcePresentSupported"/>

            <!-- Timeout for the BioAPI_NOTIFY_SOURCE_PRESENT event -->
            <input name="_sourcepresenttimeout"/>

            <!-- Timeout for BioSPI_Capture -->
            <input name="_capturetimeout"/>
      <!-- Invocation of the primary activity of this assertion with input parameter values
                  assigned from the assertion's parameters. -->
      <invoke activity="BioSPI_CreateTemplate">
            <input name="bspUuid" var="_bspUuid"/>
            <input name="inserttimeouttime" var="_inserttimeout"/>
            <input name="nosourcepresentsupported"
                        var="_noSourcePresentSupported"/>
            <input name="sourcepresenttimeouttime"
                        var="_sourcepresenttimeout"/>
            <input name="capturetimeouttime" var="_capturetimeout"/>
      </invoke>

      <!-- Activity bound to a function of the framework callback interface exposed by the
                  testing component.  This activity will be automatically invoked on each
                  incoming call to the function to which it is bound. -->
      <bind activity="EventHandler"
                  package="02c59458-0c46-1085-95d7-0002a5d5fd2e"
                  function="BioSPI_EventHandler"/>
</assertion>

<activity name="BioSPI_CreateTemplate">
      <input name="bspUuid"/>
      <input name="inserttimeouttime"/>
      <input name="nosourcepresentsupported"/>
      <input name="sourcepresenttimeouttime"/>
      <input name="capturetimeouttime"/>

      <!-- This assertion will use BSPHandle "1" for all BioSPI calls that require it -->
      <set name="_bsphandle" value="1"/>

      <!-- Invoke the functions BioSPI_BSPLoad and BioSPI_BSPAttach exposed by the BSP under
                  test.
            The input value for the parameter "unitIDOrNull" is "0", therefore the
                  assertion will test a sensor unit chosen by the BSP. -->
      <invoke activity="LoadAndAttach"
                  package="02c59458-0c46-1085-95d7-0002a5d5fd2e"
                  break_on_break="true">
            <input name="bspUuid" var="bspUuid"/>
            <input name="bspVersion" value="32"/>
            <input name="unitIDOrNull" value="0"/>
            <input name="bspHandle" var="_bsphandle"/>
            <input name="eventtimeouttime" var="inserttimeouttime"/>
      </invoke>

      <set name="eventtimeoutflag" value="false"/>

      <!-- If the BSP under test claims support for the BioAPI_NOTIFY_SOURCE_PRESENT event
                  notification, wait until that notification has been received, but no
                  longer than the specified maximum duration.-->
      <wait_until timeout_var="sourcepresenttimeouttime"
                  setvar="eventtimeoutflag">
            <or var1="nosourcepresentsupported" var2="_sourcePresent"/>
      </wait_until>

      <!-- Issue a conformity response.
            If the condition specified in the <description> below is false, an UNDECIDED
                  conformity response is issued and the execution of the activity is
                  interrupted, otherwise a PASS conformity response is issued.-->
      <assert_condition response_if_false="undecided"
                  break_if_false="true">
            <description>
                  Either the BSP under test does not claim support for the
                        BioAPI_NOTIFY_SOURCE_PRESENT event notification, or the event
                        notification has been received within the specified maximum duration.
            </description>
            <not var="eventtimeoutflag"/>
      </assert_condition>

      <!-- The BSP is ready to capture. Invoke the function BioSPI_Capture for the purpose of
                  creating a template. The handle of the captured BIR is stored in the
                  variable "capturedbir". -->
      <invoke function="BioSPI_Capture">
            <input name="BSPHandle" var="_bsphandle"/>
            <input name="Purpose"
                        var="__BioAPI_PURPOSE_ENROLL_FOR_VERIFICATION_ONLY"/>
            <input name="Subtype" value="0"/>
```

```
        <input name="Timeout" var="capturetimeouttime"/>
        <input name="no_AuditData" value="true"/>
        <output name="CapturedBIR" setvar="capturedbir_handle"/>
        <return setvar="return"/>
  </invoke>

  <!-- Issue a conformity response.
              If the condition specified in the <description> below is false, an UNDECIDED
                    conformity response is issued and the execution of the activity is
                    interrupted, otherwise a PASS conformity response is issued.-->
  <assert_condition response_if_false="undecided"
              break_if_false="true">
        <description>
              The function BioSPI_Capture has returned BioAPI_OK.
        </description>
        <equal_to var1="return" var2="__BioAPI_OK"/>
  </assert_condition>

  <!-- Check if the processed level of the captured BIR is PROCESSED. If it is PROCESSED,
                    then an UNDECIDED conformity response is issued and the execution of the
                    activity is interrupted. -->
  <invoke activity="check_capturedBIR_datatype"
              package="02c59458-0c46-1085-95d7-0002a5d5fd2e"
              break_on_break="true" >
        <input name="BSPHandle" var="_bsphandle"/>
        <input name="BIRHandle" var="capturedbir_handle"/>
  </invoke>

  <!-- Invoke the function BioSPI_CreateTemplate. -->
  <invoke function="BioSPI_CreateTemplate">
        <input name="BSPHandle" var="_bsphandle"/>
        <input name="CapturedBIR_Form" var="__BioAPI_BIR_HANDLE_INPUT" />
        <input name="CapturedBIR_BIRHandle" var="capturedbir_handle"/>
        <output name="NewTemplate" setvar="newTemplate_handle"/>
        <return setvar="return"/>
  </invoke>

  <!-- Issue a conformity response.
              If the condition specified in the <description> below is false, an UNDECIDED
                    conformity response is issued and the execution of the activity is
                    interrupted, otherwise a PASS conformity response is issued.-->
  <assert_condition response_if_false="undecided"
              break_if_false="true">
        <description>
              The function BioSPI_CreateTemplate has returned BioAPI_OK.
        </description>
        <equal_to var1="return" var2="__BioAPI_OK"/>
  </assert_condition>

  <!-- Invoke the function BioSPI_GetHeaderFromHandle on the newly created template BIR. --
              >
  <invoke function="BioSPI_GetHeaderFromHandle">
        <input name="BSPHandle" var="_bsphandle"/>
        <input name="Handle" var="newTemplate_handle"/>
        <output name="ProcessedLevel" setvar="processedLevel"/>
        <return setvar="return"/>
  </invoke>

  <!-- Issue a conformity response.
              If the condition specified in the <description> below is false, an UNDECIDED
                    conformity response is issued and the execution of the activity is
                    interrupted, otherwise a PASS conformity response is issued.-->
  <assert_condition response_if_false="undecided"
              break_if_false="true">
        <description>
              The function BioSPI_GetHeaderFromHandle has returned BioAPI_OK.
        </description>
        <equal_to var1="return" var2="__BioAPI_OK"/>
  </assert_condition>

  <!-- Issue a conformity response.
              If the condition specified in the <description> below is false, a FAIL
                    conformity response is issued, otherwise a PASS conformity response is
                    issued . -->
  <assert_condition>
        <description>
```

```
                    The processed level of the new template BIR is PROCESSED
                </description>
                <equal_to var1="processedLevel"
                        var2="__BioAPI_BIR_DATA_TYPE_PROCESSED" />
            </assert_condition>

            <!-- Invoke the functions BioSPI_BSPDetach and BioSPI_BSPeUnload -->
            <invoke activity="DetachAndUnload"
                    package="02c59458-0c46-1085-95d7-0002a5d5fd2e" >
                <input name="bspUuid" var="bspUuid" />
                <input name="BSPHandle" var="_bsphandle" />
            </invoke>
        </activity>

</package>
```

8.35 Assertion 10d - *BioSPI_CreateTemplate_OutputBIRPurpose*

Description: This assertion tests if the purpose of the created template is the same as the purpose of the captured BIR.

Excerpts

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);*

Subclause 8.4.2.1

This function takes a BIR containing biometric data in intermediate form for the purpose of creating a new enrollment template.

References: 9.3.4.2 and 8.4.2.1.

Scenario:

1) Load the BSP under test.

2) Attach the BSP under test.

3) Call the function BioSPI_Capture with a purpose of BioAPI_PURPOSE_ENROLL_FOR_VERIFICATION_ONLY to obtain a BIR for enrollment.

4) Call BioSPI_CreateTemplate without StoredTemplate and with Payload set to 0.

5) Check the return code, which is expected to be BioAPI_OK.

6) Call BioSPI_GetHeaderFromHandle for both captured BIR and the NewTemplate to compare them. They are expected to have the same purpose (BioAPI_PURPOSE_ENROLL_FOR_VERIFICATION_ONLY).

The assertion waits for the BioAPI_NOTIFY_SOURCE_PRESENT event notification (if the BSP claims support for this event).

If any of the intermediate operations fails, an UNDECIDED conformity response is issued.

Expected results: The purpose of the new template is the same as that of the captured BIR.

Assertion language package

```
<package name="03dbdaa0-0cf2-1085-99ed-0002a5d5fd2e">
      <author>
            ISO/IEC JTC1 SC37
      </author>

      <description>
            This package contains the assertion "BioSPI_CreateTemplate_OutputBIRPurpose" (see the
                        "description" element of the assertion below).
      </description>

      <assertion name="BioSPI_CreateTemplate_OutputBIRPurpose" model="BSPTesting">
            <description>
                  This assertion tests if the purpose of the created template is the same as the
                        purpose of the captured BIR.
                  The relevant text in BioAPI 2.0 is quoted below from subclause 9.3.4.2 and 8.4.2.1.
                  ________________________________________________________________________________
                  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);

                        NOTE: Details of the function definition are located in clause 8.4.2,
                              BioAPI_CreateTemplate.
                  ________________________________________________________________________________
                  Subclause 8.4.2.1:
                  This function takes a BIR containing biometric data in intermediate form for the
                        purpose of creating a new enrollment template.

                  In order to determine conformance with respect to the text above, the following
                        steps are performed:

                        1)    Load the BSP under test.
                        2)    Attach the BSP under test.
                        3)    Call the function BioSPI_Capture with a purpose of
                              BioAPI_PURPOSE_ENROLL_FOR_VERIFICATION_ONLY to obtain a BIR for
                              enrollment.
                        4)    Call BioSPI_CreateTemplate without StoredTemplate and with Payload set
                              to 0.
                        5)    Check the return code, which is expected to be BioAPI_OK.
                        6)    Call BioSPI_GetHeaderFromHandle for both captured BIR and the
                              NewTemplate to compare them. They are expected to have the same purpose
                              (BioAPI_PURPOSE_ENROLL_FOR_VERIFICATION_ONLY).

                  The assertion waits for the BioAPI_NOTIFY_SOURCE_PRESENT event notification (if the
                        BSP claims support for this event).

                  If any of the intermediate operations fails, an UNDECIDED conformity response is
                        issued.
            </description>

            <!-- UUID of the BSP under test -->
            <input name="_bspUuid"/>

            <!-- Timeout for the BioAPI_NOTIFY_INSERT event -->
            <input name="_inserttimeout"/>

            <!-- Indicates whether the BSP under test does not claim support for the
                        BioAPI_NOTIFY_SOURCE_PRESENT event notification -->
            <input name="_noSourcePresentSupported"/>
      <!-- Timeout for the BioAPI_NOTIFY_SOURCE_PRESENT event -->
      <input name="_sourcepresenttimeout"/>

      <!-- Timeout for BioSPI_Capture -->
      <input name="_capturetimeout"/>

      <!-- Invocation of the primary activity of this assertion with input parameter values
                  assigned from the assertion's parameters. -->
      <invoke activity="BioSPI_CreateTemplate">
            <input name="bspUuid" var="_bspUuid"/>
            <input name="inserttimeouttime" var="_inserttimeout"/>
            <input name="nosourcepresentsupported"
                  var="_noSourcePresentSupported"/>
            <input name="sourcepresenttimeouttime"
                  var="_sourcepresenttimeout"/>
            <input name="capturetimeouttime" var="_capturetimeout"/>
      </invoke>

      <!-- Activity bound to a function of the framework callback interface exposed by the
                  testing component.  This activity will be automatically invoked on each
                  incoming call to the function to which it is bound. -->
      <bind activity="EventHandler"
            package="02c59458-0c46-1085-95d7-0002a5d5fd2e"
            function="BioSPI_EventHandler"/>
</assertion>

<activity name="BioSPI_CreateTemplate">
      <input name="bspUuid"/>
      <input name="inserttimeouttime"/>
      <input name="nosourcepresentsupported"/>
      <input name="sourcepresenttimeouttime"/>
      <input name="capturetimeouttime"/>

      <!-- This assertion will use BSPHandle "1" for all BioSPI calls that require it -->
      <set name="_bsphandle" value="1"/>

      <!-- Invoke the functions BioSPI_BSPLoad and BioSPI_BSPAttach exposed by the BSP under
                  test.
            The input value for the parameter "unitIDOrNull" is "0", therefore the
                  assertion will test a sensor unit chosen by the BSP. -->
      <invoke activity="LoadAndAttach"
            package="02c59458-0c46-1085-95d7-0002a5d5fd2e"
            break_on_break="true">
            <input name="bspUuid" var="bspUuid"/>
            <input name="bspVersion" value="32"/>
            <input name="unitIDOrNull" value="0"/>
            <input name="bspHandle" var="_bsphandle"/>
            <input name="eventtimeouttime" var="inserttimeouttime"/>
      </invoke>

      <set name="eventtimeoutflag" value="false"/>

      <!-- If the BSP under test claims support for the BioAPI_NOTIFY_SOURCE_PRESENT event
                  notification, wait until that notification has been received, but no
                  longer than the specified maximum duration.-->
      <wait_until timeout_var="sourcepresenttimeouttime"
            setvar="eventtimeoutflag">
            <or var1="nosourcepresentsupported" var2="_sourcePresent"/>
      </wait_until>

      <!-- Issue a conformity response.
            If the condition specified in the <description> below is false, an UNDECIDED
                  conformity response is issued and the execution of the activity is
                  interrupted, otherwise a PASS conformity response is issued.-->
      <assert_condition response_if_false="undecided"
            break_if_false="true">
            <description>
                  Either the BSP under test does not claim support for the
                        BioAPI_NOTIFY_SOURCE_PRESENT event notification, or the event
                        notification has been received within the specified maximum duration.
            </description>
            <not var="eventtimeoutflag"/>
      </assert_condition>
```

```
<!-- The BSP is ready to capture.  Invoke the function BioSPI_Capture for the purpose of
                 creating a template.  The handle of the captured BIR is stored in the
                 variable "capturedbir". -->
<invoke function="BioSPI_Capture">
      <input name="BSPHandle" var="_bsphandle"/>
      <input name="Purpose"
                 var="__BioAPI_PURPOSE_ENROLL_FOR_VERIFICATION_ONLY"/>
      <input name="Subtype" value="0"/>
      <input name="Timeout" var="capturetimeouttime"/>
      <input name="no_AuditData" value="true"/>
      <output name="CapturedBIR" setvar="capturedbir_handle"/>
      <return setvar="return"/>
</invoke>

<!-- Issue a conformity response.
            If the condition specified in the <description> below is false, an UNDECIDED
                 conformity response is issued and the execution of the activity is
                 interrupted, otherwise a PASS conformity response is issued.-->
<assert_condition response_if_false="undecided"
            break_if_false="true">
      <description>
            The function BioSPI_Capture has returned BioAPI_OK.
      </description>
      <equal_to var1="return" var2="__BioAPI_OK"/>
</assert_condition>

<!-- Check if the processed level of the captured BIR is PROCESSED. If it is PROCESSED,
                 then an UNDECIDED conformity response is issued and the execution of the
                 activity is interrupted. -->
<invoke activity="check_capturedBIR_datatype"
            package="02c59458-0c46-1085-95d7-0002a5d5fd2e"
                        break_on_break="true" >
      <input name="BSPHandle" var="_bsphandle"/>
      <input name="BIRHandle" var="capturedbir_handle"/>
</invoke>

<!-- Invoke the function BioSPI_CreateTemplate. -->
<invoke function="BioSPI_CreateTemplate">
      <input name="BSPHandle" var="_bsphandle"/>
      <input name="CapturedBIR_Form"
                 var="__BioAPI_BIR_HANDLE_INPUT"/>
      <input name="CapturedBIR_BIRHandle"
                 var="capturedbir_handle"/>
      <output name="NewTemplate" setvar="newTemplate_handle"/>
      <return setvar="return"/>
</invoke>

<!-- Issue a conformity response.
            If the condition specified in the <description> below is false, an UNDECIDED
                 conformity response is issued and the execution of the activity is
                 interrupted, otherwise a PASS conformity response is issued.-->
<assert_condition response_if_false="undecided"
            break_if_false="true">
      <description>
            The function BioSPI_CreateTemplate has returned BioAPI_OK.
      </description>
      <equal_to var1="return" var2="__BioAPI_OK"/>
</assert_condition>

<!-- Invoke the function BioSPI_GetHeaderFromHandle for the captured BIR. -->
<invoke function="BioSPI_GetHeaderFromHandle">
      <input name="BSPHandle" var="_bsphandle"/>
      <input name="Handle" var="capturedbir_handle"/>
      <output name="Purpose" setvar="purpose1"/>
      <return setvar="return"/>
</invoke>

<!-- Issue a conformity response.
            If the condition specified in the <description> below is false, an UNDECIDED
                 conformity response is issued and the execution of the activity is
                 interrupted, otherwise a PASS conformity response is issued.-->
<assert_condition response_if_false="undecided"
            break_if_false="true">
      <description>
            The function BioSPI_GetHeaderFromHandle has returned BioAPI_OK.
      </description>
```

```
                <equal_to var1="return" var2="__BioAPI_OK"/>
          </assert_condition>

          <!-- Invoke the function BioSPI_GetHeaderFromHandle for the newly created template BIR. -
                      ->
          <invoke function="BioSPI_GetHeaderFromHandle">
                <input name="BSPHandle" var="_bsphandle"/>
                <input name="Handle" var="newTemplate_handle"/>
                <output name="Purpose" setvar="purpose2"/>
                <return setvar="return"/>
          </invoke>

          <!-- Issue a conformity response.
                      If the condition specified in the <description> below is false, an UNDECIDED
                            conformity response is issued and the execution of the activity is
                            interrupted, otherwise a PASS conformity response is issued.-->
          <assert_condition response_if_false="undecided"
                      break_if_false="true">
                <description>
                      The function BioSPI_GetHeaderFromHandle has returned BioAPI_OK.
                </description>
                <equal_to var1="return" var2="__BioAPI_OK"/>
          </assert_condition>

          <!-- Issue a conformity response.
                      If the condition specified in the <description> below is false, a FAIL
                            conformity response is issued, otherwise a PASS conformity response is
                            issued.-->
          <assert_condition>
                <description>
                      The purpose of the template created is the same as the purpose of the
                            captured BIR.
                </description>
                <equal_to var1="purpose1" var2="purpose2"/>
          </assert_condition>

          <!-- Invoke the functions BioSPI_BSPDetach and BioSPI_BSPUnload -->
          <invoke activity="DetachAndUnload"
                      package="02c59458-0c46-1085-95d7-0002a5d5fd2e" >
                <input name="bspUuid" var="bspUuid"/>
                <input name="BSPHandle" var="_bsphandle"/>
          </invoke>
    </activity>

</package>
```

8.36 Assertion 10e - *BioSPI_CreateTemplate_InputBIRDataType*

Description: This assertion tests BioSPI_CreateTemplate with an input BIR that has an invalid processed level.

Excerpts

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);*

Subclause 8.4.2.1

This function takes a BIR containing raw biometric data for the purpose of creating a new enrollment template. A new BIR is constructed from the CapturedBIR, and (optionally) it may perform an adaptation based on an existing StoredTemplate.

References: 9.3.4.2 and 8.4.2.1.

Scenario:

1) Load the BSP under test.

2) Attach the BSP under test.

3) Call BioSPI_Enroll with a purpose of BioAPI_PURPOSE_ENROLL_FOR_VERIFICATION_ONLY to obtain a processed template.

4) Call BioSPI_GetBIRHeaderFromHandle.

5) Check that the processed level of the BIR is PROCESSED.

6) Call BioSPI_CreateTemplate specifying the input BIR.

7) Check if the return value is different from BioAPI_OK.

8) Detach and unload the BSP under test.

If any of the intermediate operations fails, an UNDECIDED conformity response is issued.

Expected results: The call to BioSPI_CreateTemplate returns an error.

Assertion language package

```
<package name="6d543ea0-2ce9-11d9-9669-0800200c9a66">
      <author>
            ISO/IEC JTC1 SC37
      </author>

      <description>
            This package contains the assertion "BioSPI_CreateTemplate_InputBIRDataType" (see the
                              "description" element of the assertion below).
      </description>

      <assertion name="BioSPI_CreateTemplate_InputBIRDataType" model="BSPTesting">
            <description>
                  This assertion tests BioSPI_CreateTemplate with an input BIR that has an invalid
                              processed level.
                  The relevant text in BioAPI 2.0 is quoted below from subclause 9.3.4.2 and 8.4.2.1.
                  ________________________________________________________
                  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);

                        NOTE: Details of the function definition are located in clause 8.4.2,
                              BioAPI_CreateTemplate.
                  ________________________________________________________
                  Subclause 8.4.2.1:
                  This function takes a BIR containing raw biometric data for the purpose of creating
                              a new enrollment template. A new BIR is constructed from the
                              CapturedBIR, and (optionally) it may perform an adaptation based on an
                              existing StoredTemplate.
                  ________________________________________________________
```

```
          In order to determine conformance with respect to the text above, the following
                steps are performed:

             1)   Load the BSP under test.
             2)   Attach the BSP under test.
             3)   Call BioSPI_Enroll with a purpose of
                  BioAPI_PURPOSE_ENROLL_FOR_VERIFICATION_ONLY to obtain a processed
                  template.
             4)   Call BioSPI_GetBIRHeaderFromHandle.
             5)   Check that the processed level of the BIR is PROCESSED.
             6)   Call BioSPI_CreateTemplate specifying the input BIR.
             7)   Check if the return value is different from BioAPI_OK.
             8)   Detach and unload the BSP under test.

          If any of the intermediate operations fails, an UNDECIDED conformity response is
                issued.
      </description>

      <!-- UUID of the BSP under test -->
      <input name="_bspUuid"/>

      <!-- Timeout for the BioAPI_NOTIFY_INSERT event -->
      <input name="_inserttimeout"/>

      <!-- Indicates whether the BSP under test does not claim support for the
                BioAPI_NOTIFY_SOURCE_PRESENT event notification -->
      <input name="_noSourcePresentSupported" />

      <!-- Timeout for the BioAPI_NOTIFY_SOURCE_PRESENT event -->
      <input name="_sourcepresenttimeout"/>

      <!-- Timeout for BioSPI_Capture -->
      <input name="_capturetimeout"/>

      <!-- Invocation of the primary activity of this assertion with input parameter values
                assigned from the assertion's parameters. -->
      <invoke activity="BioSPI_CreateTemplate_InputBIRDataType">
          <input name="bspUuid" var="_bspUuid"/>
          <input name="inserttimeouttime" var="_inserttimeout"/>
          <input name="nosourcepresentsupported"
                var="_noSourcePresentSupported" />
          <input name="sourcepresenttimeouttime"
                var="_sourcepresenttimeout"/>
          <input name="capturetimeouttime" var="_capturetimeout"/>
      </invoke>

      <!-- Activity bound to a function of the framework callback interface exposed by the
                testing component.  This activity will be automatically invoked on each
                incoming call to the function to which it is bound. -->
      <bind activity="EventHandler"
             package="02c59458-0c46-1085-95d7-0002a5d5fd2e"
             function="BioSPI_EventHandler"/>
  </assertion>

  <activity name="BioSPI_CreateTemplate_InputBIRDataType">
      <input name="bspUuid"/>
      <input name="inserttimeouttime"/>
      <input name="nosourcepresentsupported" />
      <input name="sourcepresenttimeouttime"/>
      <input name="capturetimeouttime"/>

      <!-- This assertion will use BSPHandle "1" for all BioSPI calls that require it -->
      <set name="_bsphandle" value="1"/>

      <!-- Invoke the functions BioSPI_BSPLoad and BioSPI_BSPAttach exposed by the BSP under
                test.
             The input value for the parameter "unitIDOrNull" is "0", therefore the
                assertion will test a sensor unit chosen by the BSP. -->
      <invoke activity="LoadAndAttach"
             package="02c59458-0c46-1085-95d7-0002a5d5fd2e"
             break_on_break="true">
          <input name="bspUuid" var="bspUuid"/>
          <input name="bspVersion" value="32"/>
          <input name="unitIDOrNull" value="0"/>
          <input name="bspHandle" var="_bsphandle"/>
          <input name="eventtimeouttime" var="inserttimeouttime"/>
```

```
</invoke>

<set name="eventtimeoutflag" value="false"/>

<!-- If the BSP under test claims support for the BioAPI_NOTIFY_SOURCE_PRESENT event
                  notification, wait until that notification has been received, but no
                  longer than the specified maximum duration.-->
<wait_until timeout_var="sourcepresenttimeouttime"
            setvar="eventtimeoutflag">
      <or var1="nosourcepresentsupported" var2="_sourcePresent" />
</wait_until>

<!-- Issue a conformity response.
            If the condition specified in the <description> below is false, an UNDECIDED
                  conformity response is issued and the execution of the activity is
                  interrupted, otherwise a PASS conformity response is issued.-->
<assert_condition response_if_false="undecided"
            break_if_false="true">
      <description>
            Either the BSP under test does not claim support for the
                  BioAPI_NOTIFY_SOURCE_PRESENT event notification, or the event
                  notification has been received within the specified maximum duration.
      </description>
      <not var="eventtimeoutflag"/>
</assert_condition>

<!-- The BSP is ready to capture. Invoke the function BioSPI_Enroll for the purpose of
                  enrollment. -->
<invoke function="BioSPI_Enroll">
      <input name="BSPHandle" var="_bsphandle"/>
      <input name="Purpose"
                  var="__BioAPI_PURPOSE_ENROLL_FOR_VERIFICATION_ONLY"/>
      <input name="Subtype" value="0"/>
      <input name="Timeout" var="capturetimeouttime"/>
      <output name="NewTemplate" setvar="capturedbir_handle"/>
      <return setvar="return"/>
</invoke>

<!-- Issue a conformity response.
            If the condition specified in the <description> below is false, an UNDECIDED
                  conformity response is issued and the execution of the activity is
                  interrupted, otherwise a PASS conformity response is issued.-->
<assert_condition response_if_false="undecided"
            break_if_false="true">
      <description>
                  The function BioSPI_Enroll has returned BioAPI_OK
      </description>
      <equal_to var1="return" var2="__BioAPI_OK"/>
</assert_condition>

<!-- Invoke the function BioSPI_GetHeaderFromHandle. -->
<invoke function="BioSPI_GetHeaderFromHandle">
      <input name="BSPHandle" var="_bsphandle"/>
      <input name="Handle" var="capturedbir_handle"/>
      <output name="ProcessedLevel" setvar="processedLevel"/>
      <return setvar="return"/>
</invoke>

<!-- Issue a conformity response.
            If the condition specified in the <description> below is false, an UNDECIDED
                  conformity response is issued and the execution of the activity is
                  interrupted, otherwise a PASS conformity response is issued.-->
<assert_condition response_if_false="undecided"
            break_if_false="true">
      <description>
                  The function BioSPI_GetHeaderFromHandle has returned BioAPI_OK.
      </description>
      <equal_to var1="return" var2="__BioAPI_OK"/>
</assert_condition>

<!-- Issue a conformity response.
            If the condition specified in the <description> below is false, an UNDECIDED
                  conformity response is issued and the execution of the activity is
                  interrupted, otherwise a PASS conformity response is issued.-->
<assert_condition response_if_false="undecided"
            break_if_false="true">
```

```
                <description>
                    The processed level of the enrolled BIR is processed.
                </description>
                <equal_to var1="processedLevel" var2="__BioAPI_BIR_DATA_TYPE_PROCESSED"/>
            </assert_condition>

            <!-- Invoke the function BioSPI_CreateTemplate. -->
            <invoke function="BioSPI_CreateTemplate">
                <input name="BSPHandle" var="_bsphandle"/>
                <input name="CapturedBIR_Form" var="__BioAPI_BIR_HANDLE_INPUT" />
                <input name="CapturedBIR_BIRHandle" var="capturedbir_handle"/>
                <output name="NewTemplate" setvar="newTemplate_handle"/>
                <return setvar="return"/>
            </invoke>

            <!-- Issue a conformity response.
                    If the condition specified in the <description> below is false, a FAIL
                        conformity response is issued, otherwise a PASS conformity response is
                        issued.-->
            <assert_condition>
                <description>
                    The function BioSPI_CreateTemplate has returned an error.
                </description>
                <not_equal_to var1="return" var2="__BioAPI_OK"/>
            </assert_condition>

            <!-- Invoke the functions BioSPI_ModuleDetach and BioSPI_ModuleUnload -->
            <invoke activity="DetachAndUnload"
                    package="02c59458-0c46-1085-95d7-0002a5d5fd2e" >
                <input name="bspUuid" var="bspUuid" />
                <input name="BSPHandle" var="_bsphandle" />
            </invoke>
        </activity>

</package>
```

8.37 Assertion 10f - *BioSPI_CreateTemplate_Inconsistent_Purpose*

Description: This assertion invokes the function BioSPI_CreateTemplate with an invalid input BIR purpose.

Excerpts

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);*

Subclause 8.4.2.1

This function takes a BIR containing raw biometric data for the purpose of creating a new enrollment template. A new BIR is constructed from the CapturedBIR, and (optionally) it may perform an adaptation based on an existing StoredTemplate.

References: 9.3.4.2 and 8.4.2.1.

Scenario:

1) Load the BSP under test.

2) Attach the BSP under test.

3) Call BioSPI_Capture with a purpose of BioAPI_PURPOSE_VERIFY to obtain a BIR.

4) Call BioSPI_CreateTemplate specifying the input BIR.

5) Check if the return value is equal to BioAPIERR_INCONSISTENT_PURPOSE.

6) Detach and unload the BSP under test.

If any of the intermediate operations fails, an UNDECIDED conformity response is issued.

Expected results: The call to BioSPI_CreateTemplate returns BioAPIERR_INCONSISTENT_PURPOSE.

Assertion language package

```
<package name="28ec1620-e995-11d9-b1d1-0002a5d5c51b">
      <author>
            ISO/IEC JTC1 SC37
      </author>

      <description>
            This package contains the assertion "BioSPI_CreateTemplate_Inconsistent_Purpose" (see the
                              "description" element of the assertion below).
      </description>

      <assertion name="BioSPI_CreateTemplate_Inconsistent_Purpose" model="BSPTesting">
            <description>
                  This assertion invokes the function BioSPI_CreateTemplate with an invalid input BIR
                              purpose.
                  The relevant text in BioAPI 2.0 is quoted below from subclauses 9.3.4.2 and
                              8.4.2.1.
                  ________________________________________________________
                  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);
                  ________________________________________________________
                  Subclause 8.4.2.1:
                  This function takes a BIR containing raw biometric data for the purpose of creating
                              a new enrollment template. A new BIR is constructed from the
                              CapturedBIR, and (optionally) it may perform an adaptation based on an
                              existing StoredTemplate.
                  ________________________________________________________

                  In order to determine conformance with respect to the text above, the following
                              steps are performed:

                        1)    Load the BSP under test.
                        2)    Attach the BSP under test.
                        3)    Call BioSPI_Capture with a purpose of BioAPI_PURPOSE_VERIFY to obtain a
                              BIR.
                        4)    Call BioSPI_CreateTemplate specifying the input BIR.
                        5)    Check if the return value is equal to __BioAPIERR_INCONSISTENT_PURPOSE.
                        6)    Detach and unload the BSP under test.

                  If any of the intermediate operations fails, an UNDECIDED conformity response is
                              issued.
            </description>

            <!-- UUID of the BSP under test -->
            <input name="_bspUuid"/>
```

```
        <!-- Timeout for the BioAPI_NOTIFY_INSERT event -->
        <input name="_inserttimeout"/>

        <!-- Indicates whether the BSP under test does not claim support for the
                          BioAPI_NOTIFY_SOURCE_PRESENT event notification -->
        <input name="_noSourcePresentSupported"/>

        <!-- Timeout for the BioAPI_NOTIFY_SOURCE_PRESENT event -->
        <input name="_sourcepresenttimeout"/>

        <!-- Timeout for BioSPI_Capture -->
        <input name="_capturetimeout"/>

        <!-- Invocation of the primary activity of this assertion with input parameter values
                          assigned from the assertion's parameters. -->
        <invoke activity="BioSPI_CreateTemplate">
              <input name="bspUuid" var="_bspUuid"/>
              <input name="inserttimeouttime" var="_inserttimeout"/>
              <input name="nosourcepresentsupported"
                          var="_noSourcePresentSupported"/>
              <input name="sourcepresenttimeouttime"
                          var="_sourcepresenttimeout"/>
              <input name="capturetimeouttime" var="_capturetimeout"/>
        </invoke>

        <!-- Activity bound to a function of the framework callback interface exposed by the
                          testing component.  This activity will be automatically invoked on each
                          incoming call to the function to which it is bound. -->
        <bind activity="EventHandler"
                    package="02c59458-0c46-1085-95d7-0002a5d5fd2e"
                    function="BioSPI_EventHandler"/>
  </assertion>

  <activity name="BioSPI_CreateTemplate">
        <input name="bspUuid"/>
        <input name="inserttimeouttime"/>
        <input name="nosourcepresentsupported"/>
        <input name="sourcepresenttimeouttime"/>
        <input name="capturetimeouttime"/>

        <!-- This assertion will use BSPHandle "1" for all BioSPI calls that require it -->
        <set name="_bsphandle" value="1"/>

        <!-- Invoke the functions BioSPI_BSPLoad and BioSPI_BSPAttach exposed by the BSP under
                          test.
                    The input value for the parameter "unitIDOrNull" is "0", therefore the
                          assertion will test a sensor unit chosen by the BSP. -->
        <invoke activity="LoadAndAttach"
                    package="02c59458-0c46-1085-95d7-0002a5d5fd2e"
                    break_on_break="true">
              <input name="bspUuid" var="bspUuid"/>
              <input name="bspVersion" value="32"/>
              <input name="unitIDOrNull" value="0"/>
              <input name="bspHandle" var="_bsphandle"/>
              <input name="eventtimeouttime" var="inserttimeouttime"/>
        </invoke>

        <set name="eventtimeoutflag" value="false"/>

        <!-- If the BSP under test claims support for the BioAPI_NOTIFY_SOURCE_PRESENT event
                          notification, wait until that notification has been received, but no
                          longer than the specified maximum duration.-->
        <wait_until timeout_var="sourcepresenttimeouttime"
                    setvar="eventtimeoutflag">
              <or var1="nosourcepresentsupported" var2="_sourcePresent"/>
        </wait_until>

        <!-- Issue a conformity response.
                    If the condition specified in the <description> below is false, an UNDECIDED
                          conformity response is issued and the execution of the activity is
                          interrupted, otherwise a PASS conformity response is issued.-->
        <assert_condition response_if_false="undecided"
                    break_if_false="true">
              <description>
```

```
                    Either the BSP under test does not claim support for the
                          BioAPI_NOTIFY_SOURCE_PRESENT event notification, or the event
                          notification has been received within the specified maximum duration.
              </description>
              <not var="eventtimeoutflag"/>
        </assert_condition>

        <!-- The BSP is ready to capture. Invoke the function BioSPI_Capture for the purpose of
                          creating a template. The handle of the captured BIR is stored in the
                          variable "capturedbir". -->
        <invoke function="BioSPI_Capture">
        <input name="BSPHandle" var="_bsphandle"/>
        <input name="Purpose"
                    var="__BioAPI_PURPOSE_VERIFY"/>
        <input name="Subtype" value="0"/>
        <input name="Timeout" var="capturetimeouttime"/>
        <input name="no_AuditData" value="true"/>
        <output name="CapturedBIR" setvar="capturedbir_handle"/>
        <return setvar="return"/>
  </invoke>

        <!-- Issue a conformity response.
                    If the condition specified in the <description> below is false, an UNDECIDED
                          conformity response is issued and the execution of the activity is
                          interrupted, otherwise a PASS conformity response is issued.-->
        <assert_condition response_if_false="undecided"
                    break_if_false="true">
              <description>
                          The function BioSPI_Capture has returned BioAPI_OK.
              </description>
              <equal_to var1="return" var2="__BioAPI_OK"/>
        </assert_condition>

        <!-- Check if the processed level of the captured BIR is PROCESSED.  If it is PROCESSED,
                          then an UNDECIDED conformity response is issued and the execution of the
                          activity is interrupted. -->
        <invoke activity="check_capturedBIR_datatype"
                    package="02c59458-0c46-1085-95d7-0002a5d5fd2e"
                    break_on_break="true" >
              <input name="BSPHandle" var="_bsphandle"/>
              <input name="BIRHandle" var="capturedbir_handle"/>
        </invoke>

        <!-- Invoke the function BioSPI_CreateTemplate. -->
        <invoke function="BioSPI_CreateTemplate">
              <input name="BSPHandle" var="_bsphandle"/>
              <input name="CapturedBIR_Form" var="__BioAPI_BIR_HANDLE_INPUT"/>
              <input name="CapturedBIR_BIRHandle" var="capturedbir_handle"/>
              <output name="NewTemplate" setvar="newTemplate_handle"/>
              <return setvar="return"/>
        </invoke>

        <!-- Issue a conformity response.
                    If the condition specified in the <description> below is false, a FAIL
                          conformity response is issued, otherwise a PASS conformity response is
                          issued.-->
        <assert_condition>
              <description>
                    The function BioSPI_CreateTemplate has returned
                          BioAPIERR_INCONSISTENT_PURPOSE.
              </description>
              <equal_to var1="return"
                          var2="__BioAPIERR_BSP_INCONSISTENT_PURPOSE"/>
        </assert_condition>

        <!-- Invoke the functions BioSPI_BSPDetach and BioSPI_BSPUnload -->
        <invoke activity="DetachAndUnload"
                    package="02c59458-0c46-1085-95d7-0002a5d5fd2e" >
              <input name="bspUuid" var="bspUuid"/>
              <input name="BSPHandle" var="_bsphandle"/>
        </invoke>
  </activity>
</package>
```

8.38 Assertion 11a - *BioSPI_Process_ValidParam*

Description: This assertion tests BioSPI_Process with valid parameters and the returned processed level.

Excerpts

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);*

Subclause 8.4.3.1

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_FUNCTION_NOT_SUPPORTED.

References: 9.3.4.3 and 8.4.3.1.

Scenario:

1) Load the BSP under test

2) Attach the BSP under test

3) Call BioSPI_Capture to obtain an intermediate BIR.

4) Call BioSPI_Process to generate a processed BIR. The function is expected to return BioAPI_OK.

5) Call BioSPI_GetHeaderFromHandle for the processed BIR. The processed level is expected to be PROCESSED.

6) Detach and unload the BSP under test.

If any of the intermediate operations fails, an UNDECIDED conformity response is issued.

Expected results: The BIR generated by BioSPI_Process has a processed level of PROCESSED.

Assertion language package

```
<package name="4ec34700-e9a0-11d9-8fc8-0002a5d5c51b">
      <author>
            ISO/IEC JTC1 SC37
      </author>

      <description>
            This package contains the assertion "BioSPI_Process_ValidParam" (see the "description"
                              element of the assertion below).
      </description>

      <assertion name="BioSPI_Process_ValidParam" model="BSPTesting">
            <description>
                  This assertion tests BioSPI_Process with valid parameters and checks the returned
                              processed level.
                  The relevant text in BioAPI 2.0 is quoted below from subclause 8.4.3.1.
```

```
          ___________________________________________________________
          BioAPI_RETURN BioAPI BioSPI_Process
                (BioAPI_HANDLE  BSPHandle,
                const BioAPI_INPUT_BIR  *CapturedBIR,
                const BioAPI_BIR_BIOMETRIC_DATA_FORMAT  *OutputFormat,
                BioAPI_BIR_HANDLE  *ProcessedBIR);
          ___________________________________________________________
          Subclause 8.4.3.1:
          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_FUNCTION_NOT_SUPPORTED.
          ____________________________________________________________

          In order to determine conformance with respect to the text above, the following
                      steps are performed:

                1)    Load the BSP under test
                2)    Attach the BSP under test
                3)    Call BioSPI_Capture to obtain an intermediate BIR.
                4)    Call BioSPI_Process to generate a processed BIR.  The function is
                      expected to return BioAPI_OK.
                5)    Call BioSPI_GetHeaderFromHandle for the processed BIR.  The processed
                      level is expected to be PROCESSED.
                6)    Detach and unload the BSP under test.

          If any of the intermediate operations fails, an UNDECIDED conformity response is
                      issued.
    </description>

    <!-- UUID of the BSP under test -->
    <input name="_bspUuid"/>

    <!-- Timeout for the BioAPI_NOTIFY_INSERT event -->
    <input name="_inserttimeout"/>

    <!-- Indicates whether the BSP under test does not claim support for the
                      BioAPI_NOTIFY_SOURCE_PRESENT event notification -->
    <input name="_noSourcePresentSupported" />

    <!-- Timeout for the BioAPI_NOTIFY_SOURCE_PRESENT event -->
    <input name="_sourcepresenttimeout"/>

    <!-- Timeout for BioSPI_Capture -->
    <input name="_capturetimeout"/>

    <!-- Invocation of the primary activity of this assertion with input parameter values
                      assigned from the assertion's parameters. -->
    <invoke activity="BioSPI_Process">
          <input name="bspUuid" var="_bspUuid"/>
          <input name="inserttimeouttime" var="_inserttimeout"/>
          <input name="nosourcepresentsupported"
                      var="_noSourcePresentSupported" />
          <input name="sourcepresenttimeouttime"
                      var="_sourcepresenttimeout"/>
          <input name="capturetimeouttime" var="_capturetimeout"/>
    </invoke>

    <!-- Activity bound to a function of the framework callback interface exposed by the
                      testing component.  This activity will be automatically invoked on each
                      incoming call to the function to which it is bound. -->
    <bind activity="EventHandler"
                package="02c59458-0c46-1085-95d7-0002a5d5fd2e"
                function="BioSPI_EventHandler"/>
</assertion>

<activity name="BioSPI_Process">
    <input name="bspUuid"/>
    <input name="inserttimeouttime"/>
    <input name="nosourcepresentsupported" />
    <input name="sourcepresenttimeouttime"/>
    <input name="capturetimeouttime"/>
```

```
<!-- This assertion will use BSPHandle "1" for all BioSPI calls that require it -->
<set name="_bsphandle" value="1"/>

      <!-- Invoke the functions BioSPI_BSPLoad and BioSPI_BSPAttach exposed by the BSP under
                        test.
                  The input value for the parameter "unitIDOrNull" is "0", therefore the
                        assertion will test a sensor unit chosen by the BSP. -->
      <invoke activity="LoadAndAttach"
                  package="02c59458-0c46-1085-95d7-0002a5d5fd2e"
                  break_on_break="true">
            <input name="bspUuid" var="bspUuid"/>
            <input name="bspVersion" value="32"/>
            <input name="unitIDOrNull" value="0"/>
            <input name="bspHandle" var="_bsphandle"/>
            <input name="eventtimeouttime" var="inserttimeouttime"/>
      </invoke>

      <set name="eventtimeoutflag" value="false"/>
      <!-- If the BSP under test claims support for the BioAPI_NOTIFY_SOURCE_PRESENT event
                        notification, wait until that notification has been received, but no
                        longer than the specified maximum duration.-->
      <wait_until timeout_var="sourcepresenttimeouttime"
                  setvar="eventtimeoutflag">
            <or var1="nosourcepresentsupported" var2="_sourcePresent" />
      </wait_until>

      <!-- Issue a conformity response.
                  If the condition specified in the <description> below is false, an UNDECIDED
                        conformity response is issued and the execution of the activity is
                        interrupted, otherwise a PASS conformity response is issued.-->
      <assert_condition response_if_false="undecided"
                  break_if_false="true">
            <description>
                  Either the BSP under test does not claim support for the
                        BioAPI_NOTIFY_SOURCE_PRESENT event notification, or the event
                        notification has been received within the specified maximum duration.
            </description>
            <not var="eventtimeoutflag"/>
      </assert_condition>

      <!-- The BSP is ready to capture. Invoke the function BioSPI_Capture for the purpose of
                        creating a template. The handle of the captured BIR is stored in the
                        variable "capturedbir". -->
      <invoke function="BioSPI_Capture">
            <input name="BSPHandle" var="_bsphandle"/>
            <input name="Purpose" var="__BioAPI_PURPOSE_VERIFY"/>
            <input name="Subtype" value="0"/>
            <input name="Timeout" var="capturetimeouttime"/>
            <input name="no_AuditData" value="true"/>
            <output name="CapturedBIR" setvar="capturedbir_handle"/>
            <return setvar="return"/>
      </invoke>

      <!-- Issue a conformity response.
                  If the condition specified in the <description> below is false, an UNDECIDED
                        conformity response is issued and the execution of the activity is
                        interrupted, otherwise a PASS conformity response is issued.-->
      <assert_condition response_if_false="undecided"
                  break_if_false="true">
            <description>
                  The function BioSPI_Capture has returned BioAPI_OK
            </description>
            <equal_to var1="return" var2="__BioAPI_OK"/>
      </assert_condition>

      <!-- Check if the processed level of the captured BIR is PROCESSED.  If it is PROCESSED,
                        then an UNDECIDED conformity response is issued and the execution of the
                        activity is interrupted. -->
      <invoke activity="check_capturedBIR_datatype"
                  package="02c59458-0c46-1085-95d7-0002a5d5fd2e"
                  break_on_break="true" >
            <input name="BSPHandle" var="_bsphandle" />
            <input name="BIRHandle"  var="capturedbir_handle" />
      </invoke>
```

```
            <!-- Invoke the function BioSPI_Process.-->
            <invoke function="BioSPI_Process">
                  <input name="BSPHandle" var="_bsphandle"/>
                  <input name="CapturedBIR_Form" var="__BioAPI_BIR_HANDLE_INPUT" />
                  <input name="CapturedBIR_BIRHandle" var="capturedbir_handle"/>
                  <output name="ProcessedBIR" setvar="processedbir_handle"/>
                  <return setvar="return"/>
            </invoke>

            <!-- Issue a conformity response.
                        If the condition specified in the <description> below is false, a FAIL
                              conformity response is issued, otherwise a PASS conformity response is
                              issued.-->
            <assert_condition>
                  <description>
                        The function BioSPI_Process has returned BioAPI_OK.
                  </description>
                  <equal_to var1="return" var2="__BioAPI_OK"/>
            </assert_condition>

            <!-- Invoke the function BioSPI_GetHeaderFromHandle. -->
            <invoke function="BioSPI_GetHeaderFromHandle">
                  <input name="BSPHandle" var="_bsphandle"/>
                  <input name="Handle" var="processedbir_handle"/>
                  <output name="ProcessedLevel" setvar="processedLevel"/>
                  <return setvar="return"/>
            </invoke>

            <!-- Issue a conformity response.
                        If the condition specified in the <description> below is false, an UNDECIDED
                              conformity response is issued and the execution of the activity is
                              interrupted, otherwise a PASS conformity response is issued.-->
            <assert_condition response_if_false="undecided"
                        break_if_false="true">
                  <description>
                              The function BioSPI_GetHeaderFromHandle has returned BioAPI_OK.
                  </description>
                  <equal_to var1="return" var2="__BioAPI_OK"/>
            </assert_condition>

            <!-- Issue a conformity response.
                        If the condition specified in the <description> below is false, a FAIL
                              conformity response is issued, otherwise a PASS conformity response is
                              issued.-->
            <assert_condition>
                  <description>
                        The processed level of the processed BIR is processed.
                  </description>
                  <equal_to var1="processedLevel" var2="__BioAPI_BIR_DATA_TYPE_PROCESSED"/>
            </assert_condition>

            <!-- Invoke the functions BioSPI_BSPDetach and BioSPI_BSPUnload -->
            <invoke activity="DetachAndUnload"
                        package="02c59458-0c46-1085-95d7-0002a5d5fd2e" >
                  <input name="bspUuid" var="bspUuid" />
                  <input name="BSPHandle" var="_bsphandle" />
            </invoke>
      </activity>

</package>
```

8.39 Assertion 11b - *BioSPI_Process_BIRHeaderQuality*

Description: This assertion tests the BioSPI_Process with valid parameters and if the returned quality is valid.

Excerpts

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);*

Subclause 8.4.3.1

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_FUNCTION_NOT_SUPPORTED.

Subclause 7.49.3

Quality measurements are reported as an integral value in the range 0-100 except as follows:

Value of -1: BioAPI_QUALITY was not set by the BSP (reference BSP vendor's documentation for explanation).

Value of -2: BioAPI_QUALITY is not supported by the BSP.

Subclause 7.47

#define BioAPI_QUALITY_PROCESSED (0x00000008)

If set, BSP supports the return of quality value (in the BIR header) for processed biometric data.

Subclause A.4.6.2.2

Return of quality.

Similarly, when a BIR is processed, another quality calculation may be performed and the quality value included in the header of the processedBIR (and the optional AdaptedBIR).

This would occur during BioAPI_CreateTemplate, BioAPI_Process, BioAPI_Verify, BioAPI_VerifyMatch, BioAPI_Enroll, and BioAPI_Import (ConstructedBIR) operations.

The BSP must post to the module registry whether or not it supports the calculation of quality measurements for each type of BIR - raw, intermediate, and processed.

References: 9.3.4.3, 8.4.3.1, 7.49.3, 7.47, and A.4.6.2.2

Scenario:

1) Load the BSP under test

2) Attach the BSP under test

3) Call BioSPI_Capture to obtain an intermediate BIR.

4) Call BioSPI_Process to generate a processed BIR. The function is expected to return BioAPI_OK.

5) Call BioSPI_GetHeaderFromHandle for the processed BIR. Check if the quality value is valid.

6) Detach and unload the BSP under test.

If any of the intermediate operations fails, an UNDECIDED conformity response is issued.

Expected results: The BIR generated by BioSPI_Process has a valid quality value.

Assertion language package

```
<package name="211668e0-e9a6-11d9-bcc8-0002a5d5c51b">
      <author>
            ISO/IEC JTC1 SC37
      </author>

      <description>
            This package contains the assertion "BioSPI_Process_BIRHeaderQuality" (see the
                              "description" element of the assertion below).
      </description>

      <assertion name="BioSPI_Process_BIRHeaderQuality" model="BSPTesting">
            <description>
                  This assertion tests the BioSPI_Process with valid parameters and checks if the
                              returned quality is valid.
                  The relevant text in BioAPI 2.0 is quoted below from subclauses 8.4.3.1 and 7.49.3
                  ________________________________________________________
                  BioAPI_RETURN BioAPI BioSPI_Process
                        (BioAPI_HANDLE  BSPHandle,
                        const BioAPI_INPUT_BIR  *CapturedBIR,
                        const BioAPI_BIR_BIOMETRIC_DATA_FORMAT  *OutputFormat,
                        BioAPI_BIR_HANDLE  *ProcessedBIR);
                  ________________________________________________________
                  Subclause 8.4.3.1:
                  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_FUNCTION_NOT_SUPPORTED.
                  ________________________________________________________
                  Subclause 7.49.3
                  Quality measurements are reported as an integral value in the range 0-100 except as
                              follows:
                  Value of -1: BioAPI_QUALITY was not set by the BSP (reference BSP vendor's
                              documentation for explanation).
                  Value of -2: BioAPI_QUALITY is not supported by the BSP.
                  ________________________________________________________
                  Subclause 7.47:
                  #define BioAPI_QUALITY_PROCESSED (0x00000008)
                  If set, BSP supports the return of quality value (in the BIR header) for processed
                              biometric data.
                  ________________________________________________________
                  Subclause A.4.6.2.2
                  Return of quality.
                  Similarly, when a BIR is processed, another quality calculation may be performed
                              and the quality value included in the header of the processedBIR (and
                              the optional AdaptedBIR).
                  This would occur during  BioAPI_CreateTemplate, BioAPI_Process, BioAPI_Verify,
                              BioAPI_VerifyMatch, BioAPI_Enroll, and BioAPI_Import (ConstructedBIR)
                              operations.
            The BSP must post to the module registry whether or not it supports the calculation
                        of quality measurements for each type of BIR - raw, intermediate, and
                        processed.
            ____________________________________________________________
            In order to determine conformance with respect to the text above, the following
                        steps are performed:

                  1)    Load the BSP under test
                  2)    Attach the BSP under test
                  3)    Call BioSPI_Capture to obtain an intermediate BIR.
                  4)    Call BioSPI_Process to generate a processed BIR. The function is
                        expected to return BioAPI_OK.
                  5)    Call BioSPI_GetHeaderFromHandle for the processed BIR.  Check if the
                        quality value is valid.
                  6)    Detach and unload the BSP under test.

            If any of the intermediate operations fails, an UNDECIDED conformity response is
                        issued.
      </description>

      <!-- UUID of the BSP under test -->
      <input name="_bspUuid"/>

      <!-- Timeout for the BioAPI_NOTIFY_INSERT event -->
      <input name="_inserttimeout"/>

      <!-- Indicates whether the BSP under test does not claim support for the
                  BioAPI_NOTIFY_SOURCE_PRESENT event notification -->
      <input name="_noSourcePresentSupported" />

      <!-- Timeout for the BioAPI_NOTIFY_SOURCE_PRESENT event -->
      <input name="_sourcepresenttimeout"/>

      <!-- Timeout for BioSPI_Capture -->
      <input name="_capturetimeout"/>

      <!-- Indicates whether the BSP under test claims support for quality in a processed BIR -
                  ->
      <input name="_processedQualitySupported" />

      <!-- Invocation of the primary activity of this assertion with input parameter values
                  assigned from the assertion's parameters. -->
      <invoke activity="BioSPI_Process">
            <input name="bspUuid" var="_bspUuid"/>
            <input name="inserttimeouttime" var="_inserttimeout"/>
            <input name="nosourcepresentsupported"
                  var="_noSourcePresentSupported" />
            <input name="sourcepresenttimeouttime"
                  var="_sourcepresenttimeout"/>
            <input name="capturetimeouttime" var="_capturetimeout"/>
            <input name="processedQualitySupported"
                  var="_processedQualitySupported" />
      </invoke>

      <!-- Activity bound to a function of the framework callback interface exposed by the
                  testing component.  This activity will be automatically invoked on each
                  incoming call to the function to which it is bound. -->
      <bind activity="EventHandler"
                package="02c59458-0c46-1085-95d7-0002a5d5fd2e"
                function="BioSPI_EventHandler"/>
</assertion>

<activity name="BioSPI_Process">
      <input name="bspUuid"/>
      <input name="inserttimeouttime"/>
      <input name="nosourcepresentsupported" />
      <input name="sourcepresenttimeouttime"/>
      <input name="capturetimeouttime"/>
      <input name="processedQualitySupported" />

      <!-- This assertion will use BSPHandle "1" for all BioSPI calls that require it -->
      <set name="_bsphandle" value="1"/>

      <!-- Invoke the functions BioSPI_BSPLoad and BioSPI_BSPAttach exposed by the BSP under
                  test.
            The input value for the parameter "unitIDOrNull" is "0", therefore the
                  assertion will test a sensor unit chosen by the BSP. -->
<invoke activity="LoadAndAttach"
            package="02c59458-0c46-1085-95d7-0002a5d5fd2e"
            break_on_break="true">
      <input name="bspUuid" var="bspUuid"/>
      <input name="bspVersion" value="32"/>
      <input name="unitIDOrNull" value="0"/>
      <input name="bspHandle" var="_bsphandle"/>
      <input name="eventtimeouttime" var="inserttimeouttime"/>
</invoke>

<set name="eventtimeoutflag" value="false"/>

<!-- If the BSP under test claims support for the BioAPI_NOTIFY_SOURCE_PRESENT event
                  notification, wait until that notification has been received, but no
                  longer than the specified maximum duration.-->
<wait_until timeout_var="sourcepresenttimeouttime"
            setvar="eventtimeoutflag">
      <or var1="nosourcepresentsupported" var2="_sourcePresent" />
</wait_until>

<!-- Issue a conformity response.
            If the condition specified in the <description> below is false, an UNDECIDED
                  conformity response is issued and the execution of the activity is
                  interrupted, otherwise a PASS conformity response is issued.-->
<assert_condition response_if_false="undecided"
            break_if_false="true">
      <description>
            Either the BSP under test does not claim support for the
                  BioAPI_NOTIFY_SOURCE_PRESENT event notification, or the event
                  notification has been received within the specified maximum duration.
      </description>
      <not var="eventtimeoutflag"/>
</assert_condition>

<!-- The BSP is ready to capture. Invoke the function BioSPI_Capture for the purpose of
                  creating a template. The handle of the captured BIR is stored in the
                  variable "capturedbir". -->
<invoke function="BioSPI_Capture">
      <input name="BSPHandle" var="_bsphandle"/>
      <input name="Purpose" var="__BioAPI_PURPOSE_VERIFY"/>
      <input name="Subtype" value="0"/>
      <input name="Timeout" var="capturetimeouttime"/>
      <input name="no_AuditData" value="true"/>
      <output name="CapturedBIR" setvar="capturedbir_handle"/>
      <return setvar="return"/>
</invoke>

<!-- Issue a conformity response.
            If the condition specified in the <description> below is false, an UNDECIDED
                  conformity response is issued and the execution of the activity is
                  interrupted, otherwise a PASS conformity response is issued.-->
<assert_condition response_if_false="undecided"
            break_if_false="true">
      <description>
            The function BioSPI_Capture has returned BioAPI_OK
      </description>
      <equal_to var1="return" var2="__BioAPI_OK"/>
</assert_condition>

<!-- Check if the processed level of the captured BIR is PROCESSED.  If it is PROCESSED,
                  then an UNDECIDED conformity response is issued and the execution of the
                  activity is interrupted. -->
<invoke activity="check_capturedBIR_datatype"
            package="02c59458-0c46-1085-95d7-0002a5d5fd2e"
            break_on_break="true" >
      <input name="BSPHandle" var="_bsphandle" />
      <input name="BIRHandle"  var="capturedbir_handle" />
</invoke>

<!-- Invoke the function BioSPI_Process. -->
<invoke function="BioSPI_Process">
      <input name="BSPHandle" var="_bsphandle"/>
      <input name="CapturedBIR_Form"
                  var="__BioAPI_BIR_HANDLE_INPUT" />
            <input name="CapturedBIR_BIRHandle"
                        var="capturedbir_handle"/>
            <output name="ProcessedBIR" setvar="processedbir_handle"/>
            <return setvar="return"/>
      </invoke>

      <!-- Issue a conformity response.
                  If the condition specified in the <description> below is false, an UNDECIDED
                        conformity response is issued and the execution of the activity is
                        interrupted, otherwise a PASS conformity response is issued.-->
      <assert_condition response_if_false="undecided"
                  break_if_false="true">
            <description>
                  The function BioSPI_Process has returned BioAPI_OK.
            </description>
            <equal_to var1="return" var2="__BioAPI_OK"/>
      </assert_condition>

      <!-- Invoke the function BioSPI_GetHeaderFromHandle on the processed BIR. -->
      <invoke function="BioSPI_GetHeaderFromHandle">
            <input name="BSPHandle" var="_bsphandle"/>
            <input name="Handle" var="processedbir_handle"/>
            <output name="Quality" setvar="quality"/>
            <return setvar="return"/>
      </invoke>

      <!-- Issue a conformity response.
                  If the condition specified in the <description> below is false, an UNDECIDED
                        conformity response is issued and the execution of the activity is
                        interrupted, otherwise a PASS conformity response is issued.-->
      <assert_condition response_if_false="undecided"
                  break_if_false="true">
            <description>
                        The function BioSPI_GetHeaderFromHandle has returned BioAPI_OK
            </description>
            <equal_to var1="return" var2="__BioAPI_OK"/>
      </assert_condition>

      <invoke activity="check_quality_supported"
                  package="02c59458-0c46-1085-95d7-0002a5d5fd2e"
                  break_on_break="true">
            <only_if>
                  <same_as var1="processedQualitySupported" value2="true" />
            </only_if>
            <input name="quality" var="quality" />
      </invoke>

      <invoke activity="check_quality_not_supported"
                  package="02c59458-0c46-1085-95d7-0002a5d5fd2e"
                  break_on_break="true" >
            <only_if>
                  <same_as var1="processedQualitySupported"
                              value2="false" />
            </only_if>
            <input name="quality" var="quality" />
      </invoke>

      <!-- Invoke the functions BioSPI_BSPDetach and BioSPI_BSPUnload -->
      <invoke activity="DetachAndUnload"
                  package="02c59458-0c46-1085-95d7-0002a5d5fd2e" >
            <input name="bspUuid" var="bspUuid" />
            <input name="BSPHandle" var="_bsphandle" />
      </invoke>
</activity>

</package>
```

8.40 Assertion 11c - *BioSPI_Process_OutputBIRPurpose*

Description: This assertion tests the BioSPI_Process with valid parameters and if the purpose of the processed BIR is the same as the purpose of the captured BIR.

Excerpts

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);*

Subclause 8.4.3.1

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_FUNCTION_NOT_SUPPORTED.

Subclause 7.12.3

e) The Process, CreateTemplate, and ProcessWithAuxData functions do not have Purpose as an input parameter, but read the Purpose field from the BIR header of the input CapturedBIR.

f) The Process function may accept as input any intermediate BIR with a Purpose of Verify or Identify, and shall output only BIRs with the same purpose as the input BIR.

References: 9.3.4.3, 8.4.3.1, and 7.12.3

Scenario:

1) Load the BSP under test.

2) Attach the BSP under test.

3) Call BioSPI_Capture with a purpose of BioAPI_PURPOSE_ENROLL_FOR_VERIFICATION_ONLY to obtain a BIR for enrollment.

4) Call BioSPI_Process.

5) Check the return code, which is expected to be BioAPI_OK.

6) Call BioSPI_GetHeaderFromHandle for both capturedBIR and Processed BIR to compare them. They are expected to have the same purpose.

If any of the intermediate operations fails, an UNDECIDED conformity response is issued.

Expected results: The BIR generated by BioSPI_Process has the purpose BioAPI_PURPOSE_ENROLL_FOR_VERIFICATION_ONLY.

Assertion language package

```
<package name="e1bb4f20-ed61-11d9-9344-0002a5d5c51b">
      <author>
            ISO/IEC JTC1 SC37
      </author>

      <description>
            This package contains the assertion "BioSPI_Process_OutputBIRPurpose" (see the
                              "description" element of the assertion below).
      </description>

      <assertion name="BioSPI_Process_OutputBIRPurpose" model="BSPTesting">
            <description>
                  This assertion tests the BioSPI_Process with valid parameters and checks if the
                              purpose of the processed BIR is the same as the purpose of the captured
                              BIR.
                  The relevant text in BioAPI 2.0 is quoted below from subclause 8.4.3.1 and 7.12.3
                  ___________________________________________________________
                  BioAPI_RETURN BioAPI BioSPI_Process
                        (BioAPI_HANDLE  BSPHandle,
                        const BioAPI_INPUT_BIR  *CapturedBIR,
                        const BioAPI_BIR_BIOMETRIC_DATA_FORMAT  *OutputFormat,
                        BioAPI_BIR_HANDLE   *ProcessedBIR);

                  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_FUNCTION_NOT_SUPPORTED.
                  ___________________________________________________________
                  Subclause 7.12.3:
                  e) The Process, CreateTemplate, and ProcessWithAuxData functions do not have
                              Purpose as an input parameter, but read the Purpose field from the BIR
                              header of the input CapturedBIR.
                  f) The Process function may accept as input any intermediate BIR with a Purpose of
                              Verify or Identify, and shall output only BIRs with the same purpose as
                              the input BIR.
                  ___________________________________________________________

                  In order to determine conformance with respect to the text above, the following
                              steps are performed:

                        1)    Load the BSP under test.
                        2)    Attach the BSP under test.
                        3)    Call BioSPI_Capture with a purpose of
                              BioAPI_PURPOSE_ENROLL_FOR_VERIFICATION_ONLY to obtain a BIR for
                              enrollment.
                        4)    Call BioSPI_Process.
                        5)    Check the return code, which is expected to be BioAPI_OK.
                        6)    Call BioSPI_GetHeaderFromHandle for both capturedBIR and Processed BIR
                              to compare them.  They are expected to have the same purpose.

                  If any of the intermediate operations fails, an UNDECIDED conformity response is
                              issued.
            </description>

            <!-- UUID of the BSP under test -->
            <input name="_bspUuid"/>

            <!-- Timeout for the BioAPI_NOTIFY_INSERT event -->
            <input name="_inserttimeout"/>

            <!-- Indicates whether the BSP under test does not claim support for the
                              BioAPI_NOTIFY_SOURCE_PRESENT event notification -->
            <input name="_noSourcePresentSupported"/>

            <!-- Timeout for the BioAPI_NOTIFY_SOURCE_PRESENT event -->
            <input name="_sourcepresenttimeout"/>

            <!-- Timeout for BioSPI_Capture -->
            <input name="_capturetimeout"/>

            <!-- Invocation of the primary activity of this assertion with input parameter values
                              assigned from the assertion's parameters. -->
```

```
        <invoke activity="BioSPI_Process">
              <input name="bspUuid" var="_bspUuid"/>
              <input name="inserttimeouttime" var="_inserttimeout"/>
              <input name="nosourcepresentsupported"
                          var="_noSourcePresentSupported"/>
              <input name="sourcepresenttimeouttime"
                          var="_sourcepresenttimeout"/>
              <input name="capturetimeouttime" var="_capturetimeout"/>
        </invoke>

        <!-- Activity bound to a function of the framework callback interface exposed by the
                          testing component.  This activity will be automatically invoked on each
                          incoming call to the function to which it is bound. -->
        <bind activity="EventHandler"
                    package="02c59458-0c46-1085-95d7-0002a5d5fd2e"
                    function="BioSPI_EventHandler"/>
  </assertion>

  <activity name="BioSPI_Process">
        <input name="bspUuid"/>
        <input name="inserttimeouttime"/>
        <input name="nosourcepresentsupported"/>
        <input name="sourcepresenttimeouttime"/>
        <input name="capturetimeouttime"/>

        <!-- This assertion will use BSPHandle "1" for all BioSPI calls that require it -->
        <set name="_bsphandle" value="1"/>

        <!-- Invoke the functions BioSPI_BSPLoad and BioSPI_BSPAttach exposed by the BSP under
                          test.
                    The input value for the parameter "unitIDOrNull" is "0", therefore the
                          assertion will test a sensor unit chosen by the BSP. -->
        <invoke activity="LoadAndAttach"
                    package="02c59458-0c46-1085-95d7-0002a5d5fd2e"
                    break_on_break="true">
              <input name="bspUuid" var="bspUuid"/>
              <input name="bspVersion" value="32"/>
              <input name="unitIDOrNull" value="0"/>
              <input name="bspHandle" var="_bsphandle"/>
              <input name="eventtimeouttime" var="inserttimeouttime"/>
        </invoke>

        <set name="eventtimeoutflag" value="false"/>

        <!-- If the BSP under test claims support for the BioAPI_NOTIFY_SOURCE_PRESENT event
                          notification, wait until that notification has been received, but no
                          longer than the specified maximum duration.-->
        <wait_until timeout_var="sourcepresenttimeouttime"
                    setvar="eventtimeoutflag">
              <or var1="nosourcepresentsupported" var2="_sourcePresent"/>
        </wait_until>

        <!-- Issue a conformity response.
                    If the condition specified in the <description> below is false, an UNDECIDED
                          conformity response is issued and the execution of the activity is
                          interrupted, otherwise a PASS conformity response is issued.-->
        <assert_condition response_if_false="undecided"
                    break_if_false="true">
              <description>
                    Either the BSP under test does not claim support for the
                          BioAPI_NOTIFY_SOURCE_PRESENT event notification, or the event
                          notification has been received within the specified maximum duration.
              </description>
              <not var="eventtimeoutflag"/>
        </assert_condition>

        <!-- The BSP is ready to capture.  Invoke the function BioSPI_Capture for the purpose of
                          verification.  The handle of the captured BIR is stored in the variable
                          "capturedbir". -->
        <invoke function="BioSPI_Capture">
              <input name="BSPHandle" var="_bsphandle"/>
              <input name="Purpose" var="__BioAPI_PURPOSE_VERIFY"/>
              <input name="Subtype" value="0"/>
              <input name="Timeout" var="capturetimeouttime"/>
              <input name="no_AuditData" value="true"/>
              <output name="CapturedBIR" setvar="capturedbir_handle"/>
```

```
      <return setvar="return"/>
</invoke>

<!-- Issue a conformity response.
            If the condition specified in the <description> below is false, an UNDECIDED
                  conformity response is issued and the execution of the activity is
                  interrupted, otherwise a PASS conformity response is issued.-->
<assert_condition response_if_false="undecided"
            break_if_false="true">
      <description>
            The function BioSPI_Capture has returned BioAPI_OK.
      </description>
      <equal_to var1="return" var2="__BioAPI_OK"/>
</assert_condition>

<!-- Check if the processed level of the captured BIR is PROCESSED.  If it is PROCESSED,
                  then an UNDECIDED conformity response is issued and the execution of the
                  activity is interrupted. -->
<invoke activity="check_capturedBIR_datatype"
            package="02c59458-0c46-1085-95d7-0002a5d5fd2e"
            break_on_break="true" >
      <input name="BSPHandle" var="_bsphandle"/>
      <input name="BIRHandle" var="capturedbir_handle"/>
</invoke>

<!-- Invoke the function BioSPI_Process.-->
<invoke function="BioSPI_Process">
      <input name="BSPHandle" var="_bsphandle"/>
      <input name="CapturedBIR_Form"
                  var="__BioAPI_BIR_HANDLE_INPUT"/>
      <input name="CapturedBIR_BIRHandle"
                  var="capturedbir_handle"/>
      <output name="ProcessedBIR" setvar="processedbir_handle"/>
      <return setvar="return"/>
</invoke>

<!-- Issue a conformity response.
            If the condition specified in the <description> below is false, an UNDECIDED
                  conformity response is issued and the execution of the activity is
                  interrupted, otherwise a PASS conformity response is issued.-->
<assert_condition response_if_false="undecided"
            break_if_false="true">
      <description>
            The function BioSPI_Process has returned BioAPI_OK.
      </description>
      <equal_to var1="return" var2="__BioAPI_OK"/>
</assert_condition>

<!-- Invoke the function BioSPI_GetHeaderFromHandle for the captured BIR. -->
<invoke function="BioSPI_GetHeaderFromHandle">
      <input name="BSPHandle" var="_bsphandle"/>
      <input name="Handle" var="capturedbir_handle"/>
      <output name="Purpose" setvar="purpose1"/>
      <return setvar="return"/>
</invoke>

<!-- Issue a conformity response.
            If the condition specified in the <description> below is false, an UNDECIDED
                  conformity response is issued and the execution of the activity is
                  interrupted, otherwise a PASS conformity response is issued.-->
<assert_condition response_if_false="undecided"
            break_if_false="true">
      <description>
            The function BioSPI_GetHeaderFromHandle has returned BioAPI_OK.
      </description>
      <equal_to var1="return" var2="__BioAPI_OK"/>
</assert_condition>

<!-- Invoke the function BioSPI_GetHeaderFromHandle for the newly created processed BIR.
                  -->
<invoke function="BioSPI_GetHeaderFromHandle">
      <input name="BSPHandle" var="_bsphandle"/>
      <input name="Handle" var="processedbir_handle"/>
      <output name="Purpose" setvar="purpose2"/>
      <return setvar="return"/>
</invoke>
```

```
            <!-- Issue a conformity response.
                        If the condition specified in the <description> below is false, an UNDECIDED
                              conformity response is issued and the execution of the activity   is
                              interrupted, otherwise a PASS conformity response is issued.-->
            <assert_condition response_if_false="undecided"
                        break_if_false="true">
                  <description>
                        The function BioSPI_GetHeaderFromHandle has returned BioAPI_OK.
                  </description>
                  <equal_to var1="return" var2="__BioAPI_OK"/>
            </assert_condition>

            <!-- Issue a conformity response.
                        If the condition specified in the <description> below is false, a FAIL
                              conformity response is issued, otherwise a PASS conformity response is
                              issued.-->
            <assert_condition>
                  <description>
                        The purpose of the processed BIR is the same as the purpose of the captured
                              BIR.
                  </description>
                  <equal_to var1="purpose1" var2="purpose2"/>
            </assert_condition>

            <!-- Invoke the functions BioSPI_BSPDetach and BioSPI_BSPUnload -->
            <invoke activity="DetachAndUnload"
                        package="02c59458-0c46-1085-95d7-0002a5d5fd2e" >
                  <input name="bspUuid" var="bspUuid"/>
                  <input name="BSPHandle" var="_bsphandle"/>
            </invoke>
      </activity>

</package>
```

8.41 Assertion 11d - *BioSPI_Process_BuildsProcessedBIR*

Description: This assertion tests BioSPI_Process with valid parameters and the returned processed level.

Excerpts

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);*

Subclause 8.4.3.1

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_FUNCTION_NOT_SUPPORTED.

References: 9.3.4.3 and 8.4.3.1

Scenario:

1) Load the BSP under test.

2) Attach the BSP under test.

3) Call BioSPI_Capture to obtain an intermediate BIR.

4) Call BioSPI_Process to generate a processed BIR. The function is expected to return BioAPI_OK.

5) Call BioSPI_GetHeaderFromHandle for the processed BIR. Check if the data type is valid.

6) Detach and unload the BSP under test.

If any of the intermediate operations fails, an UNDECIDED conformity response is issued.

Expected results: The BIR generated by BioSPI_Process from an intermediate BIR is a processed BIR.

Assertion language package

```
<package name="f2ce6540-ed66-11d9-9618-0002a5d5c51b">
      <author>
            ISO/IEC JTC1 SC37
      </author>

      <description>
            This package contains the assertion "BioSPI_Process_BuildsProcessedBIR" (see the
                        "description" element of the assertion below).
      </description>

      <assertion name="BioSPI_Process_BuildsProcessedBIR" model="BSPTesting">
            <description>
                  This assertion tests BioSPI_Process with valid parameters and checks the returned
                        processed level.
                  The relevant text in BioAPI 2.0 is quoted below from subclauses 8.4.3.1. and
                        C.3.4.2
                  ___________________________________________________________
                  BioAPI_RETURN BioAPI BioSPI_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_FUNCTION_NOT_SUPPORTED.
                  ___________________________________________________________

                  Subclause C.3.4.2:
                  It always takes an 'intermediate' BIR as input, and may complete the processing of
                        the biometric data into 'final' form suitable for its intended purpose.
                  ___________________________________________________________

                  In order to determine conformance with respect to the text above, the following
                        steps are performed:

                        1)    Load the BSP under test.
                        2)    Attach the BSP under test.
                        3)    Call BioSPI_Capture to obtain an intermediate BIR.
                        4)    Call BioSPI_Process to generate a processed BIR.  The function is
                              expected to return BioAPI_OK.
                        5)    Call BioSPI_GetHeaderFromHandle for the processed BIR.  Check if the
                              data type is valid.
                        6)    Detach and unload the BSP under test.

                  If any of the intermediate operations fails, an UNDECIDED conformity response is
                        issued.
            </description>

            <!-- UUID of the BSP under test -->
            <input name="_bspUuid"/>

            <!-- Timeout for the BioAPI_NOTIFY_INSERT event -->
            <input name="_inserttimeout"/>
```

```
        <!-- Indicates whether the BSP under test does not claim support for the
                        BioAPI_NOTIFY_SOURCE_PRESENT event notification -->
        <input name="_noSourcePresentSupported" />

        <!-- Timeout for the BioAPI_NOTIFY_SOURCE_PRESENT event -->
        <input name="_sourcepresenttimeout"/>

        <!-- Timeout for BioSPI_Capture -->
        <input name="_capturetimeout"/>

        <!-- Invocation of the primary activity of this assertion with input parameter values
                        assigned from the assertion's parameters. -->
        <invoke activity="BioSPI_Process">
              <input name="bspUuid" var="_bspUuid"/>
              <input name="inserttimeouttime" var="_inserttimeout"/>
              <input name="nosourcepresentsupported"
                        var="_noSourcePresentSupported"/>
              <input name="sourcepresenttimeouttime"
                        var="_sourcepresenttimeout"/>
              <input name="capturetimeouttime" var="_capturetimeout"/>
        </invoke>

        <!-- Activity bound to a function of the framework callback interface exposed by the
                        testing component.  This activity will be automatically invoked on each
                        incoming call to the function to which it is bound. -->
        <bind activity="EventHandler"
                  package="02c59458-0c46-1085-95d7-0002a5d5fd2e"
                  function="BioSPI_EventHandler"/>
  </assertion>

  <activity name="BioSPI_Process">
        <input name="bspUuid"/>
        <input name="inserttimeouttime"/>
        <input name="nosourcepresentsupported" />
        <input name="sourcepresenttimeouttime"/>
        <input name="capturetimeouttime"/>

        <!-- This assertion will use BSPHandle "1" for all BioSPI calls that require it -->
        <set name="_bsphandle" value="1"/>

        <!-- Invoke the functions BioSPI_BSPLoad and BioSPI_BSPAttach exposed by the BSP under
                        test.
                  The input value for the parameter "unitIDOrNull" is "0", therefore the
                        assertion will test a sensor unit chosen by the BSP. -->
        <invoke activity="LoadAndAttach"
                  package="02c59458-0c46-1085-95d7-0002a5d5fd2e"
                  break_on_break="true">
              <input name="bspUuid" var="bspUuid"/>
              <input name="bspVersion" value="32"/>
              <input name="unitIDOrNull" value="0"/>
              <input name="bspHandle" var="_bsphandle"/>
              <input name="eventtimeouttime" var="inserttimeouttime"/>
        </invoke>

        <set name="eventtimeoutflag" value="false"/>

        <!-- If the BSP under test claims support for the BioAPI_NOTIFY_SOURCE_PRESENT event
                        notification, wait until that notification has been received, but no
                        longer than the specified maximum duration.-->
        <wait_until timeout_var="sourcepresenttimeouttime"
                  setvar="eventtimeoutflag">
              <or var1="nosourcepresentsupported" var2="_sourcePresent" />
        </wait_until>

        <!-- Issue a conformity response.
                  If the condition specified in the <description> below is false, an UNDECIDED
                        conformity response is issued and the execution of the activity is
                        interrupted, otherwise a PASS conformity response is issued.-->
        <assert_condition response_if_false="undecided"
                  break_if_false="true">
              <description>
                  Either the BSP under test does not claim support for the
                        BioAPI_NOTIFY_SOURCE_PRESENT event notification, or the event
                        notification has been received within the specified maximum duration.
        </description>
        <not var="eventtimeoutflag"/>
  </assert_condition>

  <!-- The BSP is ready to capture. Invoke the function BioSPI_Capture for the purpose of
                  creating a template. The handle of the captured BIR is stored in the
                  variable "capturedbir". -->
  <invoke function="BioSPI_Capture">
        <input name="BSPHandle" var="_bsphandle"/>
        <input name="Purpose" var="__BioAPI_PURPOSE_VERIFY"/>
        <input name="Subtype" value="0"/>
        <input name="Timeout" var="capturetimeouttime"/>
        <input name="no_AuditData" value="true"/>
        <output name="CapturedBIR" setvar="capturedbir_handle"/>
        <return setvar="return"/>
  </invoke>

  <!-- Issue a conformity response.
            If the condition specified in the <description> below is false, an UNDECIDED
                  conformity response is issued and the execution of the activity is
                  interrupted. -->
  <assert_condition response_if_false="undecided"
            break_if_false="true">
        <description>
            The function BioSPI_Capture has returned BioAPI_OK
        </description>
        <equal_to var1="return" var2="__BioAPI_OK"/>
  </assert_condition>

  <!-- Check if the processed level of the captured BIR is PROCESSED.  If it is PROCESSED,
                  then an UNDECIDED conformity response is issued and the execution of the
                  activity is interrupted. -->
  <invoke activity="check_capturedBIR_datatype"
            package="02c59458-0c46-1085-95d7-0002a5d5fd2e"
            break_on_break="true" >
        <input name="BSPHandle" var="_bsphandle" />
        <input name="BIRHandle"  var="capturedbir_handle" />
  </invoke>

  <!-- Invoke the function BioSPI_Process.-->
  <invoke function="BioSPI_Process">
        <input name="BSPHandle" var="_bsphandle"/>
        <input name="CapturedBIR_Form"
                  var="__BioAPI_BIR_HANDLE_INPUT" />
        <input name="CapturedBIR_BIRHandle"
                  var="capturedbir_handle"/>
        <output name="ProcessedBIR" setvar="processedbir_handle"/>
        <return setvar="return"/>
  </invoke>

  <!-- Issue a conformity response.
            If the condition specified in the <description> below is false, an UNDECIDED
                  conformity response is issued and the execution of the activity is
                  interrupted, otherwise a PASS conformity response is issued.-->
  <assert_condition response_if_false="undecided"
            break_if_false="true">
        <description>
            The function BioSPI_Process has returned BioAPI_OK.
        </description>
        <equal_to var1="return" var2="__BioAPI_OK"/>
  </assert_condition>

  <!-- Invoke the function BioSPI_GetHeaderFromHandle on the processed BIR. -->
  <invoke function="BioSPI_GetHeaderFromHandle">
        <input name="BSPHandle" var="_bsphandle"/>
        <input name="Handle" var="processedbir_handle"/>
        <output name="ProcessedLevel" setvar="processedLevel"/>
        <return setvar="return"/>
  </invoke>

  <!-- Issue a conformity response.
            If the condition specified in the <description> below is false, an UNDECIDED
                  conformity response is issued and the execution of the activity is
                  interrupted, otherwise a PASS conformity response is issued.-->
            <assert_condition response_if_false="undecided"
                        break_if_false="true">
                  <description>
                              The function BioSPI_GetHeaderFromHandle has returned BioAPI_OK.
                  </description>
                  <equal_to var1="return" var2="__BioAPI_OK"/>
            </assert_condition>

            <!-- Issue a conformity response.
                        If the condition specified in the <description> below is false, a FAIL
                              conformity response is issued, otherwise a PASS conformity response is
                              issued.-->
            <assert_condition>
                  <description>
                        The processed level of the output BIR is PROCESSED.
                  </description>
                  <equal_to var1="processedLevel"
                              var2="__BioAPI_BIR_DATA_TYPE_PROCESSED"/>
            </assert_condition>

            <!-- Invoke the functions BioSPI_BSPDetach and BioSPI_BSPUnload -->
            <invoke activity="DetachAndUnload"
                        package="02c59458-0c46-1085-95d7-0002a5d5fd2e" >
                  <input name="bspUuid" var="bspUuid" />
                  <input name="BSPHandle" var="_bsphandle" />
            </invoke>
      </activity>

</package>
```

8.42 Assertion 11e - *BioSPI_Process_InputBIRDataType*

Description: This assertion tests the BioSPI_Process with an input BIR having a processed level of PROCESSED and if the BioSPI_Process call fails.

Excerpts

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);*

Subclause 8.4.3.1

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_FUNCTION_NOT_SUPPORTED.

References: 9.3.4.3 and 8.4.3.1

Scenario:

1) Load the BSP under test.

2) Attach the BSP under test.

3) Call BioSPI_Capture with a purpose of BioAPI_PURPOSE_ENROLL_FOR_VERIFICATION_ONLY to obtain a BIR for enrollment.

4) Call BioSPI_GetBIRFromHandle.

5) Call BioSPI_Process if BioSPI_Capture has returned intermediate BIR.

6) Call BioSPI_Process specifying the input BIR that is processed BIR.

7) Check that the return value is different from BioAPI_OK.

8) Detach and Unload the BSP under test.

If any of the intermediate operations fails, an UNDECIDED conformity response is issued.

Expected results: The call to BioSPI_Process returns an error.

Assertion language package

```
<package name="3cf96080-ed6b-11d9-9acf-0002a5d5c51b">
      <author>
            ISO/IEC JTC1 SC37
      </author>

      <description>
            This package contains the assertion "BioSPI_Process_InputBIRDataType" (see the
                        "description" element of the assertion below).
      </description>

      <assertion name="BioSPI_Process_InputBIRDataType" model="BSPTesting">
            <description>
                  This assertion tests the BioSPI_Process with an input BIR having a processed level
                        of PROCESSED and checks if the BioSPI_Process call fails.
                  The relevant text in BioAPI 2.0 is quoted below from subclauses C.3.4.2 and
                        2.5.3.3.
                  ________________________________________________________
                  BioAPI_RETURN BioAPI BioSPI_Process
                        (BioAPI_HANDLE  BSPHandle,
                        const BioAPI_INPUT_BIR  *CapturedBIR,
                        const BioAPI_BIR_BIOMETRIC_DATA_FORMAT  *OutputFormat,
                        BioAPI_BIR_HANDLE  *ProcessedBIR);

                  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_FUNCTION_NOT_SUPPORTED.
                  ________________________________________________________
                  Subclause C.3.4.2:
                  It always takes an 'intermediate' BIR as input, and may complete the processing of
                        the biometric data into 'final' form suitable for its intended purpose.
                  ________________________________________________________

                  In order to determine conformance with respect to the text above, the following
                        steps are performed:

                  1)    Load the BSP under test.
                  2)    Attach the BSP under test.
                  3)    Call BioSPI_Capture with a purpose of
                        BioAPI_PURPOSE_ENROLL_FOR_VERIFICATION_ONLY to obtain a BIR for
                        enrollment.
                  4)    Call BioSPI_GetBIRFromHandle.
                  5)    Call BioSPI_Process if BioSPI_Capture has returned intermediate BIR.
                  6)    Call BioSPI_Process specifying the input BIR that is processed BIR.
                  7)    Check that the return value is different from BioAPI_OK.
                  8)    Detach and Unload the BSP under test.

                  If any of the intermediate operations fails, an UNDECIDED conformity response is
                        issued.
            </description>
```

```
        <!-- UUID of the BSP under test -->
        <input name="_bspUuid"/>

        <!-- Timeout for the BioAPI_NOTIFY_INSERT event -->
        <input name="_inserttimeout"/>

        <!-- Indicates whether the BSP under test does not claim support for the
                    BioAPI_NOTIFY_SOURCE_PRESENT event notification -->
        <input name="_noSourcePresentSupported" />

        <!-- Timeout for the BioAPI_NOTIFY_SOURCE_PRESENT event -->
        <input name="_sourcepresenttimeout"/>

        <!-- Timeout for BioSPI_Capture -->
        <input name="_capturetimeout"/>

        <!-- Invocation of the primary activity of this assertion with input parameter values
                    assigned from the assertion's parameters. -->
        <invoke activity="BioSPI_Process_InputBIRDataType">
              <input name="bspUuid" var="_bspUuid"/>
              <input name="inserttimeouttime" var="_inserttimeout"/>
              <input name="nosourcepresentsupported"
                    var="_noSourcePresentSupported" />
              <input name="sourcepresenttimeouttime"
                    var="_sourcepresenttimeout"/>
              <input name="capturetimeouttime" var="_capturetimeout"/>
        </invoke>

        <!-- Activity bound to a function of the framework callback interface exposed by the
                    testing component.  This activity will be automatically invoked on each
                    incoming call to the function to which it is bound. -->
        <bind activity="EventHandler"
                package="02c59458-0c46-1085-95d7-0002a5d5fd2e"
                function="BioSPI_EventHandler"/>
  </assertion>

  <activity name="BioSPI_Process_InputBIRDataType">
        <input name="bspUuid"/>
        <input name="inserttimeouttime"/>
        <input name="nosourcepresentsupported" />
        <input name="sourcepresenttimeouttime"/>
        <input name="capturetimeouttime"/>

        <!-- This assertion will use BSPHandle "1" for all BioSPI calls that require it -->
        <set name="_bsphandle" value="1"/>

        <!-- Invoke the functions BioSPI_BSPLoad and BioSPI_BSPAttach exposed by the BSP under
                    test.
                The input value for the parameter "unitIDOrNull" is "0", therefore the
                    assertion will test a sensor unit chosen by the BSP. -->
        <invoke activity="LoadAndAttach"
                package="02c59458-0c46-1085-95d7-0002a5d5fd2e"
                break_on_break="true">
              <input name="bspUuid" var="bspUuid"/>
              <input name="bspVersion" value="32"/>
              <input name="unitIDOrNull" value="0"/>
              <input name="bspHandle" var="_bsphandle"/>
              <input name="eventtimeouttime" var="inserttimeouttime"/>
        </invoke>

        <set name="eventtimeoutflag" value="false"/>

        <!-- If the BSP under test claims support for the BioAPI_NOTIFY_SOURCE_PRESENT event
                    notification, wait until that notification has been received, but no
                    longer than the specified maximum duration.-->
        <wait_until timeout_var="sourcepresenttimeouttime"
                setvar="eventtimeoutflag">
              <or var1="nosourcepresentsupported" var2="_sourcePresent" />
        </wait_until>

        <!-- Issue a conformity response.
                If the condition specified in the <description> below is false, an UNDECIDED
                    conformity response is issued and the execution of the activity is
                    interrupted, otherwise a PASS conformity response is issued.-->
```

```
<assert_condition response_if_false="undecided"
            break_if_false="true">
      <description>
            Either the BSP under test does not claim support for the
                  BioAPI_NOTIFY_SOURCE_PRESENT event notification, or the event
                  notification has been received within the specified maximum duration.
      </description>
      <not var="eventtimeoutflag"/>
</assert_condition>

<!-- The BSP is ready to capture. Invoke the function BioSPI_Capture for the purpose of
                  creating a template. The handle of the captured BIR is stored in the
                  variable "capturedbir". -->
<invoke function="BioSPI_Capture">
      <input name="BSPHandle" var="_bsphandle"/>
      <input name="Purpose" var="__BioAPI_PURPOSE_VERIFY"/>
      <input name="Subtype" value="0"/>
      <input name="Timeout" var="capturetimeouttime"/>
      <input name="no_AuditData" value="true"/>
      <output name="CapturedBIR" setvar="capturedbir_handle"/>
      <return setvar="return"/>
</invoke>

<!-- Issue a conformity response.
            If the condition specified in the <description> below is false, an UNDECIDED
                  conformity response is issued and the execution of the activity is
                  interrupted, otherwise a PASS conformity response is issued.-->
<assert_condition response_if_false="undecided"
            break_if_false="true">
      <description>
            The function BioSPI_Capture has returned BioAPI_OK
      </description>
      <equal_to var1="return" var2="__BioAPI_OK"/>
</assert_condition>

<!-- Invoke the function BioSPI_GetHeaderFromHandle on the captured BIR. -->
<invoke function="BioSPI_GetHeaderFromHandle">
      <input name="BSPHandle" var="_bsphandle"/>
      <input name="Handle" var="capturedbir_handle"/>
      <output name="ProcessedLevel" setvar="processedLevel"/>
      <return setvar="return"/>
</invoke>

<!-- Issue a conformity response.
            If the condition specified in the <description> below is false, an UNDECIDED
                  conformity response is issued and the execution of the activity is
                  interrupted, otherwise a PASS conformity response is issued.-->
<assert_condition response_if_false="undecided"
            break_if_false="true">
      <description>
                  The function BioSPI_GetHeaderFromHandle has returned BioAPI_OK
      </description>
      <equal_to var1="return" var2="__BioAPI_OK"/>
</assert_condition>

<!-- Use captured BIR if it is processed. -->
<set name="_processedbir_handle" var="capturedbir_handle">
      <only_if>
            <equal_to var1="processedLevel"
                        var2="__BioAPI_BIR_DATA_TYPE_PROCESSED" />
      </only_if>
</set>

<!-- Complete processing of intermediate BIR. -->
<invoke activity="process_bir"
            package="02c59458-0c46-1085-95d7-0002a5d5fd2e"
            break_on_break="true" >
      <only_if>
            <not_equal_to var1="processedLevel"
                        var2="__BioAPI_BIR_DATA_TYPE_PROCESSED" />
      </only_if>
      <input name="BSPHandle" var="_bsphandle" />
      <input name="CapturedBIR_BIRHandle"  var="capturedbir_handle" />
</invoke>
```

```
            <!-- Invoke the function BioSPI_Process with processed input BIR. -->
            <invoke function="BioSPI_Process">
                  <input name="BSPHandle" var="_bsphandle"/>
                  <input name="CapturedBIR_Form"
                              var="__BioAPI_BIR_HANDLE_INPUT" />
                  <input name="CapturedBIR_BIRHandle"
                              var="_processedbir_handle"/>
                  <output name="ProcessedBIR" setvar="processedbir_handle"/>
                  <return setvar="return"/>
            </invoke>

            <!-- Issue a conformity response.
                        If the condition specified in the <description> below is false, a FAIL
                              conformity response is issued, otherwise a PASS conformity response is
                              issued.-->
            <assert_condition>
                  <description>
                        The function BioSPI_Process has returned an error code.
                  </description>
                  <not_equal_to var1="return" var2="__BioAPI_OK"/>
            </assert_condition>

            <!-- Invoke the functions BioSPI_BSPDetach and BioSPI_BSPUnload -->
            <invoke activity="DetachAndUnload"
                        package="02c59458-0c46-1085-95d7-0002a5d5fd2e" >
                  <input name="bspUuid" var="bspUuid" />
                  <input name="BSPHandle" var="_bsphandle" />
            </invoke>
      </activity>

</package>
```

8.43 Assertion 12a - *BioSPI_VerifyMatch_ValidParam*

Description: This assertion checks if calling BioSPI_VerifyMatch with valid input parameters returns BioAPI_OK.

Excerpts

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);*

Subclause 8.4.5.1

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.

References: 9.3.4.5 and 8.4.5.1

Scenario:

1) Load the BSP under test.

2) Attach the BSP under test.

3) Call BioSPI_Enroll to create a template.

4) Do another capture.

5) Call BioSPI_Process on the captured BIR if its processed level is INTERMEDIATE.

6) Use the processed BIR to match against the stored template.

7) Check the return code.

If any of the intermediate operations fails, an UNDECIDED conformity response is issued.

Expected results: The call to BioSPI_VerifyMatch returns BioAPI_OK.

Assertion language package

```
<package name="688aad60-ee30-11d9-a62c-0002a5d5c51b">
      <author>
            ISO/IEC JTC1 SC37
      </author>

      <description>
            This package contains the assertion "BioSPI_VerifyMatch_ValidParam" (see the
                              "description" element of the assertion below).
      </description>

      <assertion name="BioSPI_VerifyMatch_ValidParam" model="BSPTesting">
            <description>
                  This assertion checks if calling BioSPI_VerifyMatch with valid input parameters
                              returns BioAPI_OK.
                  The relevant text in BioAPI 2.0 is quoted below from subclause and 8.4.5.1
                  ________________________________________________________
                  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);

                  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.
                  ________________________________________________________

                  In order to determine conformance with respect to the text above, the following
                              steps are performed:

                        1)    Load the BSP under test.
                        2)    Attach the BSP under test.
                        3)    Call BioSPI_Enroll to create a template.
                        4)    Do another capture.
                        5)    Call BioSPI_Process on the captured BIR if its processed level is
                              INTERMEDIATE.
                        6)    Use the processed BIR to match against the stored template.
                        7)    Check the return code.

                  If any of the intermediate operations fails, an UNDECIDED conformity response is
                              issued.
            </description>
```

```
        <!-- UUID of the BSP under test -->
        <input name="_bspUuid"/>

        <!-- Timeout for the BioAPI_NOTIFY_INSERT event -->
        <input name="_inserttimeout"/>

        <!-- Indicates whether the BSP under test does not claim support for the
                          BioAPI_NOTIFY_SOURCE_PRESENT event notification -->
        <input name="_noSourcePresentSupported" />

        <!-- Timeout for the BioAPI_NOTIFY_SOURCE_PRESENT event -->
        <input name="_sourcepresenttimeout"/>

        <!-- Timeout for BioSPI_Capture and BioSPI_Enroll -->
        <input name="_capturetimeout"/>

        <!-- MaxFMRRequested for BioSPI_VerifyMatch -->
        <input name="_maxFMRRequested" />

        <!-- Invocation of the primary activity of this assertion with input parameter values
                          assigned from the assertion's parameters. -->
        <invoke activity="BioSPI_VerifyMatch_ValidParam">
              <input name="bspUuid" var="_bspUuid"/>
              <input name="inserttimeouttime" var="_inserttimeout"/>
              <input name="nosourcepresentsupported"
                          var="_noSourcePresentSupported" />
              <input name="sourcepresenttimeouttime"
                          var="_sourcepresenttimeout"/>
              <input name="capturetimeouttime" var="_capturetimeout"/>
              <input name="maxFMRRequested" var="_maxFMRRequested"/>
        </invoke>

        <!-- Activity bound to a function of the framework callback interface exposed by the
                          testing component.  This activity will be automatically invoked on each
                          incoming call to the function to which it is bound. -->
        <bind activity="EventHandler"
                    package="02c59458-0c46-1085-95d7-0002a5d5fd2e"
                    function="BioSPI_EventHandler"/>
  </assertion>

  <activity name="BioSPI_VerifyMatch_ValidParam">
        <input name="bspUuid"/>
        <input name="inserttimeouttime"/>
        <input name="nosourcepresentsupported" />
        <input name="sourcepresenttimeouttime"/>
        <input name="capturetimeouttime"/>
        <input name="maxFMRRequested" />

        <!-- This assertion will use BSPHandle "1" for all BioSPI calls that require it -->
        <set name="_bsphandle" value="1"/>

        <!-- Invoke the functions BioSPI_BSPLoad and BioSPI_BSPAttach exposed by the BSP under
                          test.
                    The input value for the parameter "unitIDOrNull" is "0", therefore the
                          assertion will test a sensor unit chosen by the BSP. -->
        <invoke activity="LoadAndAttach"
                    package="02c59458-0c46-1085-95d7-0002a5d5fd2e"
                    break_on_break="true">
              <input name="bspUuid" var="bspUuid"/>
              <input name="bspVersion" value="32"/>
              <input name="unitIDOrNull" value="0"/>
              <input name="bspHandle" var="_bsphandle"/>
              <input name="eventtimeouttime" var="inserttimeouttime"/>
        </invoke>

        <set name="eventtimeoutflag" value="false"/>

        <!-- If the BSP under test claims support for the BioAPI_NOTIFY_SOURCE_PRESENT event
                          notification, wait until that notification has been received, but no
                          longer than the specified maximum duration.-->
        <wait_until timeout_var="sourcepresenttimeouttime"
                    setvar="eventtimeoutflag">
              <or var1="nosourcepresentsupported" var2="_sourcePresent" />
        </wait_until>

        <!-- Issue a conformity response.
```

```
            If the condition specified in the <description> below is false, an UNDECIDED
                  conformity response is issued and the execution of the activity is
                  interrupted, otherwise a PASS conformity response is issued.-->
<assert_condition response_if_false="undecided"
            break_if_false="true">
      <description>
            Either the BSP under test does not claim support for the
                  BioAPI_NOTIFY_SOURCE_PRESENT event notification, or the event
                  notification has been received within the specified maximum duration.
      </description>
      <not var="eventtimeoutflag"/>
</assert_condition>

<!-- Invoke the function BioSPI_Enroll to create a template -->
<invoke function="BioSPI_Enroll">
      <input name="BSPHandle" var="_bsphandle"/>
      <input name="Purpose"
                  var="__BioAPI_PURPOSE_ENROLL_FOR_VERIFICATION_ONLY"/>
      <input name="Subtype" value="0"/>
      <input name="Timeout" var="capturetimeouttime"/>
      <output name="NewTemplate" setvar="template_handle"/>
      <return setvar="return"/>
</invoke>

<!-- Issue a conformity response.
            If the condition specified in the <description> below is false, an UNDECIDED
                  conformity response is issued, otherwise a PASS conformity response is
                  issued.-->
<assert_condition response_if_false="undecided" >
      <description>
            The function BioSPI_Enroll has returned BioAPI_OK
      </description>
      <equal_to var1="return" var2="__BioAPI_OK"/>
</assert_condition>

<!-- Invoke the function BioSPI_Capture again to capture another BIR -->
<invoke function="BioSPI_Capture">
      <input name="BSPHandle" var="_bsphandle"/>
      <input name="Purpose" var="__BioAPI_PURPOSE_VERIFY"/>
      <input name="Subtype" value="0"/>
      <input name="Timeout" var="capturetimeouttime"/>
      <input name="no_AuditData" value="true"/>
      <output name="CapturedBIR" setvar="capturedbir_handle"/>
      <return setvar="return"/>
</invoke>

<!-- Issue a conformity response.
            If the condition specified in the <description> below is false, an UNDECIDED
                  conformity response is issued and the execution of the activity is
                  interrupted, otherwise a PASS conformity response is issued.-->
<assert_condition response_if_false="undecided"
            break_if_false="true">
      <description>
                  The function BioSPI_Capture has returned BioAPI_OK
      </description>
      <equal_to var1="return" var2="__BioAPI_OK"/>
</assert_condition>

<!-- Invoke the function BioSPI_GetHeaderFromHandle on the captured BIR. -->
<invoke function="BioSPI_GetHeaderFromHandle">
      <input name="BSPHandle" var="_bsphandle"/>
      <input name="Handle" var="capturedbir_handle"/>
      <output name="ProcessedLevel" setvar="processedLevel"/>
      <return setvar="return"/>
</invoke>
```

```
            <!-- Issue a conformity response.
                        If the condition specified in the <description> below is false, an UNDECIDED
                              conformity response is issued and the execution of the activity is
                              interrupted, otherwise a PASS conformity response is issued.-->
            <assert_condition response_if_false="undecided"
                        break_if_false="true">
                  <description>
                        The function BioSPI_GetHeaderFromHandle has returned BioAPI_OK
                  </description>
                  <equal_to var1="return" var2="__BioAPI_OK"/>
            </assert_condition>

            <!-- Initialize the global variable "_processedbir_handle" to the value of the variable
                              "capturedbir_handle" -->
            <set name="_processedbir_handle" var="capturedbir_handle" />

            <!-- If the processed level of the captured BIR is INTERMEDIATE, invoke the function
                              BioSPI_Process.-->
            <invoke activity="process_bir"
                        package="02c59458-0c46-1085-95d7-0002a5d5fd2e"
                                    break_on_break="true" >
                  <only_if>
                        <not_equal_to var1="processedLevel"
                                    var2="__BioAPI_BIR_DATA_TYPE_PROCESSED" />
                  </only_if>
                  <input name="BSPHandle" var="_bsphandle"/>
                  <input name="CapturedBIR_BIRHandle" var="capturedbir_handle"/>
            </invoke>

            <!-- Invoke BioSPI_VerifyMatch to verify the processed BIR against the stored template --
                              >
            <invoke function="BioSPI_VerifyMatch" >
                  <input name="BSPHandle" var="_bsphandle" />
                  <input name="MaxFMRRequested" var="maxFMRRequested" />
                  <input name="ProcessedBIR_Form"
                              var="__BioAPI_BIR_HANDLE_INPUT" />
                  <input name="ProcessedBIR_BIRHandle"
                              var="_processedbir_handle" />
                  <input name="ReferenceTemplate_Form"
                              var="__BioAPI_BIR_HANDLE_INPUT" />
                  <input name="ReferenceTemplate_BIRHandle"
                              var="template_handle" />
                  <input name="no_AdaptedBIR" value="true" />
                  <input name="no_Payload" value="true" />
                  <output name="Result" setvar="result" />
                  <output name="FMRAchieved" setvar="fmrAchieved" />
                  <return setvar="return"/>
            </invoke>

            <!-- Issue a conformity response.
                        If the condition specified in the <description> below is false, a FAIL
                              conformity response is issued, otherwise a PASS conformity response is
                              issued.-->
            <assert_condition>
                  <description>
                        The function BioSPI_VerifyMatch has returned BioAPI_OK
                  </description>
                  <equal_to var1="return" var2="__BioAPI_OK"/>
            </assert_condition>

            <!-- Invoke the functions BioSPI_BSPDetach and BioSPI_BSPUnload -->
            <invoke activity="DetachAndUnload"
                        package="02c59458-0c46-1085-95d7-0002a5d5fd2e" >
                  <input name="bspUuid" var="bspUuid" />
                  <input name="BSPHandle" var="_bsphandle" />
            </invoke>
      </activity>

</package>
```

8.44 Assertion 12b - *BioSPI_VerifyMatch_Payload*

Description: This assertion tests the support of payload in the function BioSPI_VerifyMatch. The function is expected to return BioAPI_OK.

Excerpts

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);*

Subclause 8.4.5.1

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.

Parameters: Payload (output/optional) - If a payload is associated with the ReferenceTemplate, it is returned in an allocated BioAPI_DATA structure if the FMRAchieved satisfies the policy of the BSP.

Subclause 7.47

#define BioAPI_PAYLOAD (0x00001000)

If set, the BSP supports payload carry (accepts payload during Enroll/CreateTemplate and returns payroll upon successful Verify/VerifyMatch).

References: 9.3.4.5, 8.4.5.1, and 7.47

Scenario:

1) Load the BSP under test.

2) Attach the BSP under test.

3) Call BioSPI_Enroll to create a template.

4) Do another capture.

5) Call BioSPI_Process on the captured BIR if its processed level is INTERMEDIATE.

6) Call BioSPI_VerifyMatch to verify the captured BIR against the stored template BIR.

7) Check the output parameter 'Payload', which is expected to be the same as the input parameter payload to BioSPI_Enroll.

8) Detach and unload the BSP under test.

If any of the intermediate operations fails, an UNDECIDED conformity response is issued.

Expected results: The call to BioSPI_VerifyMatch returns BioAPI_OK. In case of match, it returns a payload that is the same as the original payload.

Assertion language package

```
<package name="692ebe20-ee47-11d9-bd34-0002a5d5c51b">
      <author>
            ISO/IEC JTC1 SC37
      </author>

      <description>
            This package contains the assertion "BioSPI_VerifyMatch_Payload" (see the "description"
                              element of the assertion below).
      </description>

      <assertion name="BioSPI_VerifyMatch_Payload" model="BSPTesting">
            <description>
                  This assertion the support of payload in the function BioSPI_VerifyMatch.  The
                              function is expected to return BioAPI_OK.
                  The relevant text in BioAPI 2.0 is quoted below from subclauses 8.4.5.1, C.53 and
                              7.47.
                  ________________________________________________________
                  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);

                  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.
                  ________________________________________________________
                  Subclause 7.47:
                  #define BioAPI_PAYLOAD (0x00001000)
                  If set, the BSP supports payload carry (accepts payload during
                              Enroll/CreateTemplate and returns payroll upon successful
                              Verify/VerifyMatch).
                  Subclause C.53:
                  The 'payload' is released to the application on successful verification of the
                              template (during a Verify or VerifyMatch operation). The BSP may have a
                              policy of only releasing the 'payload' if the Actual FMR achieved is
                              below a certain threshold (this threshold being recorded in the BSP's
                              registry entry).
                  Subclause 8.4.5.1:
                  Parameters: Payload (output/optional) - If a payload is associated with the
                              ReferenceTemplate, it is returned in an allocated BioAPI_DATA structure
                              if the FMRAchieved satisfies the policy of the BSP.
                  ________________________________________________________

                  In order to determine conformance with respect to the text above, the following
                              steps are performed:

                        1)    Load the BSP under test.
                        2)    Attach the BSP under test.
                        3)    Call BioSPI_Enroll to create a template.
                        4)    Do another capture.
                        5)    Call BioSPI_Process on the captured BIR if its processed level is
                              INTERMEDIATE.
                        6)    Call BioSPI_VerifyMatch to verify the captured BIR against the stored
                              template BIR.
```

```
                7)    Check the output parameter 'Payload', which is expected to be the same
                      as the input parameter payload to BioSPI_Enroll.
                8)    Detach and unload the BSP under test.

          If any of the intermediate operations fails, an UNDECIDED conformity response is
                      issued.
      </description>

      <!-- UUID of the BSP under test -->
      <input name="_bspUuid"/>

      <!-- Timeout for the BioAPI_NOTIFY_INSERT event -->
      <input name="_inserttimeout"/>

      <!-- Indicates whether the BSP under test does not claim support for the
                      BioAPI_NOTIFY_SOURCE_PRESENT event notification -->
      <input name="_noSourcePresentSupported" />

      <!-- Timeout for the BioAPI_NOTIFY_SOURCE_PRESENT event -->
      <input name="_sourcepresenttimeout"/>

      <!-- Timeout for BioSPI_Capture and BioSPI_Enroll -->
      <input name="_capturetimeout"/>

      <!-- MaxFMRRequested for BioSPI_VerifyMatch -->
      <input name="_maxFMRRequested" />

      <!-- Indicates whether the BSP claims support for the payload.-->
      <input name="_payloadSupported"/>

      <!--Payload policy -->
      <input name="_payloadPolicy" />

      <!-- Payload -->
      <input name="_payload" />

      <!-- Invocation of the primary activity of this assertion with input parameter values
                      assigned from the assertion's parameters. -->
      <invoke activity="BioSPI_VerifyMatch">
            <input name="bspUuid" var="_bspUuid"/>
            <input name="inserttimeouttime" var="_inserttimeout"/>
            <input name="nosourcepresentsupported"
                      var="_noSourcePresentSupported" />
            <input name="sourcepresenttimeouttime"
                      var="_sourcepresenttimeout"/>
            <input name="capturetimeouttime" var="_capturetimeout"/>
            <input name="maxFMRRequested" var="_maxFMRRequested"/>
            <input name="payloadSupported" var="_payloadSupported"/>
            <input name="payloadPolicy" var="_payloadPolicy"/>
            <input name="payload" var="_payload"/>
      </invoke>

      <!-- Activity bound to a function of the framework callback interface exposed by the
                      testing component.  This activity will be automatically invoked on each
                      incoming call to the function to which it is bound. -->
      <bind activity="EventHandler"
                package="02c59458-0c46-1085-95d7-0002a5d5fd2e"
                function="BioSPI_EventHandler"/>
</assertion>

<activity name="BioSPI_VerifyMatch">
      <input name="bspUuid"/>
      <input name="inserttimeouttime"/>
      <input name="nosourcepresentsupported" />
      <input name="sourcepresenttimeouttime"/>
      <input name="capturetimeouttime"/>
      <input name="maxFMRRequested" />
      <input name="payloadSupported"/>
      <input name="payloadPolicy" />
      <input name="payload" />

      <!-- This assertion will use BSPHandle "1" for all BioSPI calls that require it -->
            <set name="_bsphandle" value="1"/>

      <!-- Invoke the functions BioSPI_BSPLoad and BioSPI_BSPAttach exposed by the BSP under
                      test.
                The input value for the parameter "unitIDOrNull" is "0", therefore the
                      assertion will test a sensor unit chosen by the BSP. -->
    <invoke activity="LoadAndAttach"
                package="02c59458-0c46-1085-95d7-0002a5d5fd2e"
                break_on_break="true">
          <input name="bspUuid" var="bspUuid"/>
          <input name="bspVersion" value="32"/>
          <input name="unitIDOrNull" value="0"/>
          <input name="bspHandle" var="_bsphandle"/>
          <input name="eventtimeouttime" var="inserttimeouttime"/>
    </invoke>

    <set name="eventtimeoutflag" value="false"/>

    <!-- If the BSP under test claims support for the BioAPI_NOTIFY_SOURCE_PRESENT event
                      notification, wait until that notification has been received, but no
                      longer than the specified maximum duration.-->
    <wait_until timeout_var="sourcepresenttimeouttime"
                setvar="eventtimeoutflag">
          <or var1="nosourcepresentsupported" var2="_sourcePresent" />
    </wait_until>

    <!-- Issue a conformity response.
                If the condition specified in the <description> below is false, an UNDECIDED
                      conformity response is issued and the execution of the activity is
                      interrupted, otherwise a PASS conformity response is issued.-->
    <assert_condition response_if_false="undecided"
                break_if_false="true">
          <description>
                Either the BSP under test does not claim support for the
                      BioAPI_NOTIFY_SOURCE_PRESENT event notification, or the event
                      notification has been received within the specified maximum duration.
          </description>
    <not var="eventtimeoutflag"/>
</assert_condition>

<!-- Call BioSPI_Enroll to create a template -->
    <invoke function="BioSPI_Enroll">
          <input name="BSPHandle" var="_bsphandle"/>
          <input name="Purpose"
                      var="__BioAPI_PURPOSE_ENROLL_FOR_VERIFICATION_ONLY"/>
          <input name="Subtype" value="0"/>
          <input name="Timeout" var="capturetimeouttime"/>
          <input name="Payload" var="payload" />
          <output name="NewTemplate" setvar="template_handle"/>
          <return setvar="return"/>
    </invoke>

    <!-- Issue a conformity response.
                If the condition specified in the <description> below is false, an UNDECIDED
                      conformity response is issued, otherwise a PASS conformity response is
                      issued.-->
    <assert_condition response_if_false="undecided" break_if_false="true">
          <description>
                The function BioSPI_Enroll has returned BioAPI_OK
          </description>
          <equal_to var1="return" var2="__BioAPI_OK"/>
    </assert_condition>

    <!-- Invoke the function BioSPI_Capture again to capture another BIR -->
    <invoke function="BioSPI_Capture">
          <input name="BSPHandle" var="_bsphandle"/>
          <input name="Purpose" var="__BioAPI_PURPOSE_VERIFY"/>
          <input name="Subtype" value="0"/>
          <input name="Timeout" var="capturetimeouttime"/>
          <input name="no_AuditData" value="true"/>
          <output name="CapturedBIR" setvar="capturedbir_handle"/>
          <return setvar="return"/>
    </invoke>

    <!-- Issue a conformity response.
                If the condition specified in the <description> below is false, an UNDECIDED
                      conformity response is issued and the execution of the activity is
                      interrupted, otherwise a PASS conformity response is issued.-->
    <assert_condition response_if_false="undecided"
                break_if_false="true">
```

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

<!-- Invoke the function BioSPI_GetHeaderFromHandle on the captured BIR. -->
<invoke function="BioSPI_GetHeaderFromHandle">
      <input name="BSPHandle" var="_bsphandle"/>
      <input name="Handle" var="capturedbir_handle"/>
      <output name="ProcessedLevel" setvar="processedLevel"/>
      <return setvar="return"/>
</invoke>

<!-- Issue a conformity response.
            If the condition specified in the <description> below is false, an UNDECIDED
                  conformity response is issued and the execution of the activity is
                  interrupted, otherwise a PASS conformity response is issued.-->
<assert_condition response_if_false="undecided"
            break_if_false="true">
      <description>
            The function BioSPI_GetHeaderFromHandle has returned BioAPI_OK
      </description>
      <equal_to var1="return" var2="__BioAPI_OK"/>
</assert_condition>

<!-- Set the global variable "_processedbir_handle" to the value of the variable
                  "capturedbir_handle" -->
<set name="_processedbir_handle" var="capturedbir_handle" />

<!-- If the processed level of the captured BIR is INTERMEDIATE, process the BIR -->
<invoke activity="process_bir"
            package="02c59458-0c46-1085-95d7-0002a5d5fd2e"
            break_on_break="true" >
      <only_if>
            <not_equal_to var1="processedLevel"
                        var2="__BioAPI_BIR_DATA_TYPE_PROCESSED" />
      </only_if>
      <input name="BSPHandle" var="_bsphandle"/>
      <input name="CapturedBIR_BIRHandle" var="capturedbir_handle"/>
</invoke>

<!-- Invoke BioSPI_VerifyMatch to verify the processed BIR against the stored template --
                  >
<invoke function="BioSPI_VerifyMatch" >
      <input name="BSPHandle" var="_bsphandle" />
      <input name="MaxFMRRequested" var="maxFMRRequested" />
      <input name="ProcessedBIR_Form"
                  var="__BioAPI_BIR_HANDLE_INPUT" />
      <input name="ProcessedBIR_BIRHandle"
                  var="_processedbir_handle" />
      <input name="ReferenceTemplate_Form"
                  var="__BioAPI_BIR_HANDLE_INPUT" />
      <input name="ReferenceTemplate_BIRHandle"
                  var="template_handle" />
      <input name="no_AdaptedBIR" value="true" />
      <output name="Payload" setvar="output_payload" />
      <output name="Result" setvar="result" />
      <output name="FMRAchieved" setvar="fmrAchieved" />
      <return setvar="return"/>
</invoke>

<!-- Issue a conformity response.
            If the condition specified in the <description> below is false, a FAIL
                  conformity response is issued, otherwise a PASS conformity response is
                  issued.-->
<assert_condition>
      <description>
            The function BioSPI_VerifyMatch has returned BioAPI_OK.
      </description>
      <equal_to var1="return" var2="__BioAPI_OK"/>
</assert_condition>

<!-- Check the returned payload (BSP claiming to support the payload) -->
<invoke activity="payloadSupport_checkPayload" package="02c59458-0c46-1085-95d7-
                  0002a5d5fd2e" break_on_break="true">
            <only_if>
                  <same_as var1="payloadSupported" value2="true"/>
            </only_if>
            <input name="inputPayload" var="payload" />
            <input name="outputPayload" var="output_payload"/>
            <input name="result"  var="result"/>
            <input name="payloadPolicy" var="payloadPolicy"/>
            <input name="fmrAchieved" var="fmrAchieved"/>
      </invoke>

      <!-- Check the returned payload (BSP not claiming to support the payload) -->
      <invoke activity="payloadNotSupport_checkPayload" package="02c59458-0c46-1085-95d7-
                  0002a5d5fd2e" break_on_break="true">
            <only_if>
                  <same_as var1="payloadSupported" value2="false"/>
            </only_if>
            <input name="outputPayload"  var="output_payload"/>
      </invoke>

      <!-- Invoke the functions BioSPI_BSPDetach and BioSPI_BSPUnload -->
      <invoke activity="DetachAndUnload"
                  package="02c59458-0c46-1085-95d7-0002a5d5fd2e" >
            <input name="bspUuid" var="bspUuid" />
            <input name="BSPHandle" var="_bsphandle" />
      </invoke>
   </activity>

</package>
```

8.45 Assertion 12c - *BioSPI_VerifyMatch_Inconsistent_Purpose*

Description: This assertion test BioSPI_VerifyMatch with a BIR whose purpose is invalid for the function. The function is expected to return BioAPIERR_BSP_INCONSISTENT_PURPOSE.

Excerpts

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);*

Subclause 8.4.5.1

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.

Subclause 11.2.3

#define BioAPIERR_INCONSISTENT_PURPOSE (0x000115)

The purpose recorded in the BIR and the requested purpose are inconsistent with the function being performed.

References: 9.3.4.5, 8.4.5.1, and 11.2.3

Scenario:

1) Load the BSP under test.

2) Attach the BSP under test.

3) Call BioSPI_Enroll to create a template.

4) Call BioSPI_Enroll to create another template.

5) Call BioSPI_VerifyMatch with the two templates passed as input, both having the purpose of BioAPI_PURPOSE_ENROLL_FOR_VERIFICATION_ONLY.

6) Check the return code, which is expected to be BioAPIERR_BSP_INCONSISTENT_PURPOSE.

7) Detach and unload the BSP under test.

If any of the intermediate operations fails, an UNDECIDED conformity response is issued.

Expected results: The call to BioSPI_VerifyMatch returns BioAPIERR_BSP_INCONSISTENT_PURPOSE.

Assertion language package

```
<package name="9108ec70-2e9b-11d9-9669-0800200c9a66">
      <author>
            ISO/IEC JTC1 SC37
      </author>

      <description>
            This package contains the assertion "BioSPI_VerifyMatch_Inconsistent_Purpose" (see the
                        "description" element of the assertion below).
      </description>

      <assertion name="BioSPI_VerifyMatch_Inconsistent_Purpose" model="BSPTesting">
            <description>
                  This assertion test BioSPI_VerifyMatch with a BIR whose purpose is invalid for the
                              function. The function is expected to return
                              BioAPIERR_BSP_INCONSISTENT_PURPOSE.
                  The relevant text in BioAPI 2.0 is quoted below from subclauses 8.4.5.1 and 11.2.3.
                  _________________________________________________________
                  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);

                  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.
            ________________________________________________________
            Subclause 11.2.3:
            #define BioAPIERR_INCONSISTENT_PURPOSE (0x000115)
            The purpose recorded in the BIR and the requested purpose are inconsistent with the
                        function being performed.
            ________________________________________________________

            In order to determine conformance with respect to the text above, the following
                        steps are performed:

                  1)    Load the BSP under test.
                  2)    Attach the BSP under test.
                  3)    Call BioSPI_Enroll to create a template.
                  4)    Call BioSPI_Enroll to create another template.
                  5)  Call BioSPI_VerifyMatch with the two templates passed as input, both
                        having the purpose of BioAPI_PURPOSE_ENROLL_FOR_VERIFICATION_ONLY.
                  6)    Check the return code, which is expected to be
                        BioAPIERR_BSP_INCONSISTENT_PURPOSE.
                  7)    Detach and unload the BSP under test.

            If any of the intermediate operations fails, an UNDECIDED conformity response is
                        issued.
      </description>

      <!-- UUID of the BSP under test -->
      <input name="_bspUuid"/>

      <!-- Timeout for the BioAPI_NOTIFY_INSERT event -->
      <input name="_inserttimeout"/>

      <!-- Indicates whether the BSP under test does not claim support for the
                        BioAPI_NOTIFY_SOURCE_PRESENT event notification -->
      <input name="_noSourcePresentSupported" />

      <!-- Timeout for the BioAPI_NOTIFY_SOURCE_PRESENT event -->
      <input name="_sourcepresenttimeout"/>

      <!-- Timeout for BioSPI_Capture -->
      <input name="_capturetimeout"/>

      <!-- MaxFARRequested for BioSPI_VerifyMatch -->
      <input name="_maxFMRRequested" />

      <!-- Invocation of the primary activity of this assertion with input parameter values
                        assigned from the assertion's parameters. -->
      <invoke activity="BioSPI_VerifyMatch">
            <input name="bspUuid" var="_bspUuid"/>
            <input name="inserttimeouttime" var="_inserttimeout"/>
            <input name="nosourcepresentsupported"
                        var="_noSourcePresentSupported" />
            <input name="sourcepresenttimeouttime"
                        var="_sourcepresenttimeout"/>
            <input name="capturetimeouttime" var="_capturetimeout"/>
            <input name="maxFMRRequested" var="_maxFMRRequested"/>
      </invoke>

      <!-- Activity bound to a function of the framework callback interface exposed by the
                        testing component.  This activity will be automatically invoked on each
                        incoming call to the function to which it is bound. -->
      <bind activity="EventHandler"
                  package="02c59458-0c46-1085-95d7-0002a5d5fd2e"
                  function="BioSPI_EventHandler"/>
</assertion>

<activity name="BioSPI_VerifyMatch">
      <input name="bspUuid"/>
      <input name="inserttimeouttime"/>
      <input name="nosourcepresentsupported" />
      <input name="sourcepresenttimeouttime"/>
      <input name="capturetimeouttime"/>
      <input name="maxFMRRequested" />

      <!-- This assertion will use BSPHandle "1" for all BioSPI calls that require it -->
      <set name="_bsphandle" value="1"/>
```

```
<!-- Invoke the functions BioSPI_BSPLoad and BioSPI_BSPAttach exposed by the BSP under
                  test.
            The input value for the parameter "unitIDOrNull" is "0", therefore the
                  assertion will test a sensor unit chosen by the BSP. -->
<invoke activity="LoadAndAttach"
            package="02c59458-0c46-1085-95d7-0002a5d5fd2e"
            break_on_break="true">
      <input name="bspUuid" var="bspUuid"/>
      <input name="bspVersion" value="32"/>
      <input name="unitIDOrNull" value="0"/>
      <input name="bspHandle" var="_bsphandle"/>
      <input name="eventtimeouttime" var="inserttimeouttime"/>
</invoke>

<set name="eventtimeoutflag" value="false"/>

<!-- If the BSP under test claims support for the BioAPI_NOTIFY_SOURCE_PRESENT event
                  notification, wait until that notification has been received, but no
                  longer than the specified maximum duration.-->
<wait_until timeout_var="sourcepresenttimeouttime"
            setvar="eventtimeoutflag">
      <or var1="nosourcepresentsupported" var2="_sourcePresent" />
</wait_until>

<!-- Issue a conformity response.
            If the condition specified in the <description> below is false, an UNDECIDED
                  conformity response is issued and the execution of the activity is
                  interrupted, otherwise a PASS conformity response is issued.-->
<assert_condition response_if_false="undecided"
            break_if_false="true">
      <description>
            Either the BSP under test does not claim support for the
                  BioAPI_NOTIFY_SOURCE_PRESENT event notification, or the event
                  notification has been received within the specified maximum duration.
      </description>
      <not var="eventtimeoutflag"/>
</assert_condition>

<!-- Invoke the function BioSPI_Enroll to create a template.-->
<invoke function="BioSPI_Enroll">
      <input name="BSPHandle" var="_bsphandle"/>
      <input name="Purpose"
                  var="__BioAPI_PURPOSE_ENROLL_FOR_VERIFICATION_ONLY"/>
      <input name="Subtype" value="0"/>
      <input name="Timeout" var="capturetimeouttime"/>
      <output name="NewTemplate" setvar="template_handle1"/>
      <return setvar="return"/>
</invoke>

<!-- Issue a conformity response.
            If the condition specified in the <description> below is false, an UNDECIDED
                  conformity response is issued, otherwise a PASS conformity response is
                  issued.-->
<assert_condition response_if_false="undecided" break_if_false="true">
      <description>
            The function BioSPI_Enroll has returned BioAPI_OK
      </description>
      <equal_to var1="return" var2="__BioAPI_OK"/>
</assert_condition>

<!-- Invoke the function BioSPI_Enroll again to create another template -->
<invoke function="BioSPI_Enroll">
      <input name="BSPHandle" var="_bsphandle"/>
      <input name="Purpose"
                  var="__BioAPI_PURPOSE_ENROLL_FOR_VERIFICATION_ONLY"/>
      <input name="Subtype" value="0"/>
      <input name="Timeout" var="capturetimeouttime"/>
      <output name="NewTemplate" setvar="template_handle2"/>
      <return setvar="return"/>
</invoke>
```

```
            <!-- Issue a conformity response.
                        If the condition specified in the <description> below is false, an UNDECIDED
                              conformity response is issued, otherwise a PASS conformity response is
                              issued.-->
            <assert_condition response_if_false="undecided" break_if_false="true">
                  <description>
                        The function BioSPI_Enroll has returned BioAPI_OK
                  </description>
                  <equal_to var1="return" var2="__BioAPI_OK"/>
            </assert_condition>

            <!-- Invoke the function BioSPI_VerifyMatch with the two templates to verify if the BSP
                              returns BioAPIERR_BSP_INCONSISTENT_PURPOSE -->
            <invoke function="BioSPI_VerifyMatch" >
                  <input name="BSPHandle" var="_bsphandle" />
                  <input name="MaxFMRRequested" var="maxFMRRequested" />
                  <input name="ProcessedBIR_Form"
                              var="__BioAPI_BIR_HANDLE_INPUT" />
                  <input name="ProcessedBIR_BIRHandle"
                              var="template_handle2" />
                  <input name="ReferenceTemplate_Form"
                              var="__BioAPI_BIR_HANDLE_INPUT" />
                  <input name="ReferenceTemplate_BIRHandle"
                              var="template_handle1" />
                  <input name="no_AdaptedBIR" value="true" />
                  <input name="no_Payload" value="true" />
                  <output name="Result" setvar="result" />
                  <output name="FMRAchieved" setvar="fmrAchieved" />
                  <return setvar="return"/>
            </invoke>

            <!-- Issue a conformity response.
                        If the condition specified in the <description> below is false, a FAIL
                              conformity response is issued, otherwise a PASS conformity response is
                              issued.-->
            <assert_condition>
                  <description>
                        The function BioSPI_VerifyMatch has returned BioAPIERR_INCONSISTENT_PURPOSE
                  </description>
                  <equal_to var1="return"
                              var2="__BioAPIERR_BSP_INCONSISTENT_PURPOSE"/>
            </assert_condition>

            <!-- Invoke the functions BioSPI_BSPDetach and BioSPI_BSPUnload -->
            <invoke activity="DetachAndUnload"
                        package="02c59458-0c46-1085-95d7-0002a5d5fd2e" >
                  <input name="bspUuid" var="bspUuid" />
                  <input name="BSPHandle" var="_bsphandle" />
            </invoke>
      </activity>

</package>
```

8.46 Assertion 13a - *BioSPI_Enroll_ValidParam*

Description: This assertion checks if calling BioSPI_Enroll with valid input parameters returns BioAPI_OK.

Excerpts

Subclause 9.3.4.7

BioAPI_RETURN BioAPI BioSPI_Enroll

(BioAPI_HANDLE BSPHandle,

BioAPI_BIR_PURPOSE Purpose,

BioAPI_BIR_SUBTYPE Subtype,

*const BioAPI_BIR_BIOMETRIC_DATA_FORMAT *OutputFormat,* *const BioAPI_INPUT_BIR *ReferenceTemplate,*

*BioAPI_BIR_HANDLE *NewTemplate,*

*const BioAPI_DATA *Payload,*

int32_t Timeout,

*BioAPI_BIR_HANDLE *AuditData,*

*BioAPI_UUID *TemplateUUID);*

Subclause 8.4.7.1

This function captures biometric data from the attached device (sensor unit) for the purpose of creating a ProcessedBIR for the purpose of enrollment.

Subclause A.4

This function is supported by both Verification and Identification BSPs.

References: 9.3.4.7, 8.4.7.1, and A.4

Scenario:

1) Load the BSP under test.

2) Attach the BSP under test.

3) Call BioSPI_Enroll to capture and enroll a BIR.

4) Check the return code, which is expected to be BioAPI_OK.

5) Call BioSPI_GetHeaderFromHandle to check the processed level of the new template BIR. It is expected to be PROCESSED.

If any of the intermediate operations fails, an UNDECIDED conformity response is issued.

Expected results: The call to BioSPI_Enroll returns BioAPI_OK

Assertion language package

```
<package name="0b5ebb60-eefb-11d9-990c-0002a5d5c51b">
      <author>
            ISO/IEC JTC1 SC37
      </author>

      <description>
            This package contains the assertion "BioSPI_Enroll_ValidParam" (see the "description"
                              element of the assertion below).
      </description>

      <assertion name="BioSPI_Enroll_ValidParam" model="BSPTesting">
            <description>
                  This assertion checks if calling BioSPI_Enroll with valid input parameters returns
                              BioAPI_OK.
                  The relevant text in BioAPI 2.0 is quoted below from subclauses A.4 and 8.4.7.1.
                  ________________________________________________________
                  BioAPI_RETURN BioAPI BioSPI_Enroll
                        (BioAPI_HANDLE  BSPHandle,
                        BioAPI_BIR_PURPOSE  Purpose,
                        BioAPI_BIR_SUBTYPE  Subtype,
                        const BioAPI_BIR_BIOMETRIC_DATA_FORMAT  *OutputFormat,
                        const BioAPI_INPUT_BIR  *ReferenceTemplate,
                BioAPI_BIR_HANDLE  *NewTemplate,
                const BioAPI_DATA  *Payload,
                int32_t  Timeout,
                BioAPI_BIR_HANDLE  *AuditData,
                BioAPI_UUID  *TemplateUUID);

          This function captures biometric data from the attached device (sensor unit) for
                      the purpose of creating a ProcessedBIR for the purpose of enrollment.
          _________________________________________________________
          Subclause A.4:
          This function is supported by both Verification and Identification BSPs.
          _________________________________________________________

          In order to determine conformance with respect to the text above, the following
                      steps are performed:

                1)    Load the BSP under test.
                2)    Attach the BSP under test.
                3)    Call BioSPI_Enroll to capture and enroll a BIR.
                4)    Check the return code, which is expected to be BioAPI_OK.
                5)    Call BioSPI_GetHeaderFromHandle to check the processed level of the new
                      template BIR.  It is expected to be PROCESSED.

          If any of the intermediate operations fails, an UNDECIDED conformity response is
                      issued.
    </description>

    <!-- UUID of the BSP under test -->
    <input name="_bspUuid"/>

    <!-- Timeout for the BioAPI_NOTIFY_INSERT event -->
    <input name="_inserttimeout"/>

    <!-- Indicates whether the BSP under test does not claim support for the
                      BioAPI_NOTIFY_SOURCE_PRESENT event notification -->
    <input name="_noSourcePresentSupported" />

    <!-- Timeout for the BioAPI_NOTIFY_SOURCE_PRESENT event -->
    <input name="_sourcepresenttimeout"/>

    <!-- Timeout for BioSPI_Enroll -->
    <input name="_capturetimeout"/>

    <!-- Invocation of the primary activity of this assertion with input parameter values
                      assigned from the assertion's parameters. -->
    <invoke activity="BioSPI_Enroll">
          <input name="bspUuid" var="_bspUuid"/>
          <input name="inserttimeouttime" var="_inserttimeout"/>
          <input name="nosourcepresentsupported"
                      var="_noSourcePresentSupported" />
          <input name="sourcepresenttimeouttime"
                      var="_sourcepresenttimeout"/>
          <input name="capturetimeouttime" var="_capturetimeout"/>
    </invoke>

    <!-- Activity bound to a function of the framework callback interface exposed by the
                      testing component.  This activity will be automatically invoked on each
                      incoming call to the function to which it is bound. -->
    <bind activity="EventHandler"
                package="02c59458-0c46-1085-95d7-0002a5d5fd2e"
                function="BioSPI_EventHandler"/>
</assertion>

<activity name="BioSPI_Enroll">
    <input name="bspUuid"/>
    <input name="inserttimeouttime"/>
    <input name="nosourcepresentsupported" />
    <input name="sourcepresenttimeouttime"/>
    <input name="capturetimeouttime"/>

    <!-- This assertion will use BSPHandle "1" for all BioSPI calls that require it -->
    <set name="_bsphandle" value="1"/>

    <!-- Invoke the functions BioSPI_BSPLoad and BioSPI_BSPAttach exposed by the BSP under
                      test.
```

```
            The input value for the parameter "unitIDOrNull" is "0", therefore the
                  assertion will test a sensor unit chosen by the BSP. -->
<invoke activity="LoadAndAttach"
            package="02c59458-0c46-1085-95d7-0002a5d5fd2e"
            break_on_break="true">
      <input name="bspUuid" var="bspUuid"/>
      <input name="bspVersion" value="32"/>
      <input name="unitIDOrNull" value="0"/>
      <input name="bspHandle" var="_bsphandle"/>
      <input name="eventtimeouttime" var="inserttimeouttime"/>
</invoke>

<set name="eventtimeoutflag" value="false"/>

<!-- If the BSP under test claims support for the BioAPI_NOTIFY_SOURCE_PRESENT event
                  notification, wait until that notification has been received, but no
                  longer than the specified maximum duration.-->
<wait_until timeout_var="sourcepresenttimeouttime"
            setvar="eventtimeoutflag">
      <or var1="nosourcepresentsupported" var2="_sourcePresent" />
</wait_until>

<!-- Issue a conformity response.
            If the condition specified in the <description> below is false, an UNDECIDED
                  conformity response is issued and the execution of the activity is
                  interrupted, otherwise a PASS conformity response is issued.-->
<assert_condition response_if_false="undecided"
            break_if_false="true">
      <description>
            Either the BSP under test does not claim support for the
                  BioAPI_NOTIFY_SOURCE_PRESENT event notification, or the event
                  notification has been received within the specified maximum duration.
      </description>
      <not var="eventtimeoutflag"/>
</assert_condition>

<!-- The BSP is ready to capture. Invoke the function BioSPI_Enroll for the purpose of
                  enrollment. -->
<invoke function="BioSPI_Enroll">
      <input name="BSPHandle" var="_bsphandle"/>
      <input name="Purpose"
                  var="__BioAPI_PURPOSE_ENROLL_FOR_VERIFICATION_ONLY"/>
      <input name="Subtype" value="0"/>
      <input name="Timeout" var="capturetimeouttime"/>
      <output name="NewTemplate" setvar="newtemplate_handle"/>
      <return setvar="return"/>
</invoke>

<!-- Issue a conformity response.
            If the condition specified in the <description> below is false, a FAIL
                  conformity response is issued, otherwise a PASS conformity response is
                  issued.-->
<assert_condition>
      <description>
            The function BioSPI_Enroll has returned BioAPI_OK
      </description>
      <equal_to var1="return" var2="__BioAPI_OK"/>
</assert_condition>

<!-- Invoke the function BioSPI_GetHeaderFromHandle. -->
<invoke function="BioSPI_GetHeaderFromHandle">
      <input name="BSPHandle" var="_bsphandle"/>
      <input name="Handle" var="newtemplate_handle"/>
      <output name="ProcessedLevel" setvar="processedLevel"/>
      <return setvar="return"/>
</invoke>

<!-- Issue a conformity response.
            If the condition specified in the <description> below is false, an UNDECIDED
                  conformity response is issued and the execution of the activity is
                  interrupted, otherwise a PASS conformity response is issued.-->
<assert_condition response_if_false="undecided"
            break_if_false="true">
      <description>
                  The function BioSPI_GetHeaderFromHandle has returned BioAPI_OK.
      </description>
```

```
            <equal_to var1="return" var2="__BioAPI_OK"/>
        </assert_condition>

        <!-- Issue a conformity response.
                If the condition specified in the <description> below is false, a FAIL
                    conformity response is issued, otherwise a PASS conformity response is
                    issued.-->
        <assert_condition>
            <description>
                The processed level of the enrolled BIR is processed.
            </description>
            <equal_to var1="processedLevel" var2="__BioAPI_BIR_DATA_TYPE_PROCESSED"/>
        </assert_condition>

        <!-- Invoke the functions BioSPI_BSPDetach and BioSPI_BSPUnload -->
        <invoke activity="DetachAndUnload"
                package="02c59458-0c46-1085-95d7-0002a5d5fd2e" >
            <input name="bspUuid" var="bspUuid" />
            <input name="BSPHandle" var="_bsphandle" />
        </invoke>
    </activity>

</package>
```

8.47 Assertion 13b - *BioSPI_Enroll_Payload*

Description: This assertion checks if calling BioSPI_Enroll with a payload returns BioAPI_OK.

Excerpts

Subclause 9.3.4.7

BioAPI_RETURN BioAPI BioSPI_Enroll

(BioAPI_HANDLE BSPHandle,

BioAPI_BIR_PURPOSE Purpose,

BioAPI_BIR_SUBTYPE Subtype,

*const BioAPI_BIR_BIOMETRIC_DATA_FORMAT *OutputFormat,*

*const BioAPI_INPUT_BIR *ReferenceTemplate,*

*BioAPI_BIR_HANDLE *NewTemplate,*

*const BioAPI_DATA *Payload,*

int32_t Timeout,

*BioAPI_BIR_HANDLE *AuditData,*

*BioAPI_UUID *TemplateUUID);*

Subclause 8.4.7.1

This function captures biometric data from the attached device (sensor unit) for the purpose of creating a ProcessedBIR for the purpose of enrollment.

Parameters: Payload (input/optional) - A pointer to data that will be stored by the BSP. This parameter is ignored if NULL.

Subclause A.4.6.2.6

If supported, BSPs shall post to the component registry the maximum payload size it can accommodate. A maximum size of "0" indicates payload carry is not supported. If input payloads exceed this size, an error shall be generated.

Subclause 7.47

#define BioAPI_PAYLOAD (0x00000080)

If set, the BSP supports payload carry (accepts payload during Enroll/CreateTemplate and returns payroll upon successful Verify/VerifyMatch).

References: 9.3.4.7, 8.4.7.1, A.4.6.2.6, and 7.47

Scenario:

1) Load the BSP under test

2) Attach the BSP under test

3) Call BioSPI_Enroll to capture and enroll a BIR with payload set to a non-NULL value.

4) Check the return code, which is expected to be BioAPI_OK.

If any of the intermediate operations fails, an UNDECIDED conformity response is issued.

Expected results: The call to BioSPI_Enroll returns BioAPI_OK (the payload is accepted).

Assertion language package

```
<package name="e8969d40-ef05-11d9-9098-0002a5d5c51b">
      <author>
            ISO/IEC JTC1 SC37
      </author>

      <description>
            This package contains the assertion "BioSPI_Enroll_Payload" (see the "description"
                  element of the assertion below).
      </description>

      <assertion name="BioSPI_Enroll_Payload" model="BSPTesting">
            <description>
                  This assertion checks if calling BioSPI_Enroll with a payload returns BioAPI_OK.
                  The relevant text in BioAPI 2.0 is quoted below from subclauses 7.47, A.4.6.2.6 and
                              8.4.7.1.
                  _____________________________________________________
                  BioAPI_RETURN BioAPI BioSPI_Enroll
                        (BioAPI_HANDLE  BSPHandle,
                        BioAPI_BIR_PURPOSE  Purpose,
                        BioAPI_BIR_SUBTYPE  Subtype,
                        const BioAPI_BIR_BIOMETRIC_DATA_FORMAT  *OutputFormat,
                        const BioAPI_INPUT_BIR  *ReferenceTemplate,
                        BioAPI_BIR_HANDLE  *NewTemplate,
                        const BioAPI_DATA  *Payload,
                        int32_t  Timeout,
                        BioAPI_BIR_HANDLE  *AuditData,
                        BioAPI_UUID  *TemplateUUID);

                  This function captures biometric data from the attached device (sensor unit) for
                              the purpose of creating a ProcessedBIR for the purpose of enrollment.
                  _____________________________________________________
                  Subclause 8.4.7.1:
                  Parameters: Payload (input/optional) - A pointer to data that will be stored by the
                              BSP. This parameter is ignored if NULL.
            ___________________________________________________________
            Subclause A.4.6.2.6
            If supported, BSPs shall post to the component registry the maximum payload size it
                        can accommodate.  A maximum size of "0" indicates payload carry is not
                        supported.  If input payloads exceed this size, an error shall be
                        generated.
            ___________________________________________________________
            Subclause 7.47:
            #define BioAPI_PAYLOAD (0x00000080)
            If set, the BSP supports payload carry (accepts payload during
                        Enroll/CreateTemplate and returns payroll upon successful
                        Verify/VerifyMatch).
            ______________________________________________________

            In order to determine conformance with respect to the text above, the following
                        steps are performed:

                  1)    Load the BSP under test
                  2)    Attach the BSP under test
                  3)    Call BioSPI_Enroll to capture and enroll a BIR with payload set to a
                        non-NULL value.
                  4)    Check the return code, which is expected to be BioAPI_OK.

            If any of the intermediate operations fails, an UNDECIDED conformity response is
                        issued.
      </description>

      <!-- UUID of the BSP under test -->
      <input name="_bspUuid"/>

      <!-- Timeout for the BioAPI_NOTIFY_INSERT event -->
      <input name="_inserttimeout"/>

      <!-- Indicates whether the BSP under test does not claim support for the
                        BioAPI_NOTIFY_SOURCE_PRESENT event notification -->
      <input name="_noSourcePresentSupported" />

      <!-- Timeout for the BioAPI_NOTIFY_SOURCE_PRESENT event -->
      <input name="_sourcepresenttimeout"/>

      <!-- Timeout for BioSPI_Enroll -->
      <input name="_capturetimeout"/>

      <!-- Indicates whether the BSP under test claims support for payload -->
      <input name="_supportPayload" />

      <!-- Payload must satisfy the constraint on the payload size posted by the BSP in the
                        component registry -->
      <input name="_payload" />

      <!-- Invocation of the primary activity of this assertion with input parameter values
                        assigned from the assertion's parameters. -->
      <invoke activity="BioSPI_Enroll">
            <input name="bspUuid" var="_bspUuid"/>
            <input name="inserttimeouttime" var="_inserttimeout"/>
            <input name="nosourcepresentsupported"
                        var="_noSourcePresentSupported" />
            <input name="sourcepresenttimeouttime"
                        var="_sourcepresenttimeout"/>
            <input name="capturetimeouttime" var="_capturetimeout"/>
            <input name="supportpayload"  var="_supportPayload" />
            <input name="payload" var="_payload" />
      </invoke>

      <!-- Activity bound to a function of the framework callback interface exposed by the
                        testing component.  This activity will be automatically invoked on each
                        incoming call to the function to which it is bound. -->
      <bind activity="EventHandler"
                  package="02c59458-0c46-1085-95d7-0002a5d5fd2e"
                  function="BioSPI_EventHandler"/>
</assertion>
```

```
<activity name="BioSPI_Enroll">
      <input name="bspUuid"/>
      <input name="inserttimeouttime"/>
      <input name="nosourcepresentsupported" />
      <input name="sourcepresenttimeouttime"/>
      <input name="capturetimeouttime"/>
      <input name="supportpayload" />
      <input name="payload" />

      <!-- This assertion will use BSPHandle "1" for all BioSPI calls that require it -->
      <set name="_bsphandle" value="1"/>

      <!-- Invoke the functions BioSPI_BSPLoad and BioSPI_BSPAttach exposed by the BSP under
                  test.
            The input value for the parameter "unitIDOrNull" is "0", therefore the
                  assertion will test a sensor unit chosen by the BSP. -->
      <invoke activity="LoadAndAttach"
            package="02c59458-0c46-1085-95d7-0002a5d5fd2e"
            break_on_break="true">
            <input name="bspUuid" var="bspUuid"/>
            <input name="bspVersion" value="32"/>
            <input name="unitIDOrNull" value="0"/>
            <input name="bspHandle" var="_bsphandle"/>
            <input name="eventtimeouttime" var="inserttimeouttime"/>
      </invoke>

      <set name="eventtimeoutflag" value="false"/>

      <!-- If the BSP under test claims support for the BioAPI_NOTIFY_SOURCE_PRESENT event
                  notification, wait until that notification has been received, but no
                  longer than the specified maximum duration.-->
      <wait_until timeout_var="sourcepresenttimeouttime"
            setvar="eventtimeoutflag">
            <or var1="nosourcepresentsupported" var2="_sourcePresent" />
      </wait_until>

      <!-- Issue a conformity response.
            If the condition specified in the <description> below is false, an UNDECIDED
                  conformity response is issued and the execution of the activity is
                  interrupted, otherwise a PASS conformity response is issued.-->
      <assert_condition response_if_false="undecided"
            break_if_false="true">
            <description>
                  Either the BSP under test does not claim support for the
                        BioAPI_NOTIFY_SOURCE_PRESENT event notification, or the event
                        notification has been received within the specified maximum duration.
            </description>
            <not var="eventtimeoutflag"/>
      </assert_condition>

      <!-- The BSP is ready to capture. Invoke the function BioSPI_Enroll for the purpose of
                  enrollment. -->
      <invoke function="BioSPI_Enroll">
            <input name="BSPHandle" var="_bsphandle"/>
            <input name="Purpose"
                  var="__BioAPI_PURPOSE_ENROLL_FOR_VERIFICATION_ONLY"/>
            <input name="Subtype" value="0"/>
            <input name="Timeout" var="capturetimeouttime"/>
            <input name="Payload" var="payload" />
            <output name="NewTemplate" setvar="newtemplate_handle"/>
            <return setvar="return"/>
      </invoke>

      <!-- Issue a conformity response.
            If the condition specified in the <description> below is false, a FAIL
                  conformity response is issued and the execution of the activity is
                  interrupted, otherwise a PASS conformity response is issued.-->
      <assert_condition break_if_false="true">
            <description>
                  If payload is supported by the BSP, the function BioSPI_Enroll returns
                        BioAPI_OK, otherwise BioAPIERR_BSP_UNABLE_TO_STORE_PAYLOAD is returned.
            </description>
            <or>
                  <and>
                        <equal_to var1="return" var2="__BioAPI_OK"/>
                        <same_as var1="supportpayload"  value2="true" />
                        </and>
                        <and>
                              <equal_to var1="return"
                                          var2="__BioAPIERR_BSP_UNABLE_TO_STORE_PAYLOAD" />
                              <same_as var1="supportpayload" value2="false" />
                        </and>
                  </or>
            </assert_condition>

            <!-- Invoke the functions BioSPI_BSPDetach and BioSPI_BSPUnload -->
            <invoke activity="DetachAndUnload"
                        package="02c59458-0c46-1085-95d7-0002a5d5fd2e" >
                  <input name="bspUuid" var="bspUuid" />
                  <input name="BSPHandle" var="_bsphandle" />
            </invoke>
      </activity>

</package>
```

8.48 Assertion 13c - *BioSPI_Enroll_AuditData*

Description: This assertion calls the function BioSPI_Enroll with a non-NULL AuditData parameter. The function is expected to return audit data if the BSP supports audit data.

Excerpts

Subclause 9.3.4.7

BioAPI_RETURN BioAPI BioSPI_Enroll

(BioAPI_HANDLE BSPHandle,

BioAPI_BIR_PURPOSE Purpose,

BioAPI_BIR_SUBTYPE Subtype,

*const BioAPI_BIR_BIOMETRIC_DATA_FORMAT *OutputFormat,*

*const BioAPI_INPUT_BIR *ReferenceTemplate,*

*BioAPI_BIR_HANDLE *NewTemplate,*

*const BioAPI_DATA *Payload,*

int32_t Timeout,

*BioAPI_BIR_HANDLE *AuditData,*

*BioAPI_UUID *TemplateUUID);*

Subclause 8.4.7.1

This function captures biometric data from the attached device (sensor unit) for the purpose of creating a ProcessedBIR for the purpose of enrollment.

Parameters: AuditData (output/optional) - A handle to a BIR containing raw biometric data. This data may be used to provide a human-identifiable data of the person at the sensor unit. If the pointer is NULL on input, no audit data is collected. Not all BSPs support the collection of audit data. A BSP may return a BIR handle value of BioAPI_UNSUPPORTED_BIR_HANDLE to indicate AuditData is not supported, or a value of BioAPI_INVALID_BIR_HANDLE to indicate that no audit data is available.

Subclause 7.47

#define BioAPI_RAW (0x00000001)

If set, indicates that the BSP supports the return of raw/audit data.

Subclause A.4.6.2.1

Return of raw data. Functions involving the capture of biometric data from a sensor may optionally support the return of this raw data for purposes of display or audit. If supported, the output parameter AuditData will contain a pointer to this data. If not supported, the BSP will return a value of -1.

References: 9.3.4.7, 8.4.7.1, 7.47, and A.4.6.2.1

Scenario:

1) Load the BSP under test

2) Attach the BSP under test

3) Call BioSPI_Enroll to capture and enroll a BIR with AuditData set to a non-NULL value.

4) Check the return code. If audit data is not supported, the returned BIR handle is expected to be BioAPI_UNSUPPORTED_BIR_HANDLE. If audit data is not available, the returned BIR handle is expected to be BioAPI_INVALID_BIR_HANDLE.

5) Invoke BioSPI_GetHeaderFromHandle to check the type of the audit data, if supported; it is expected to be RAW.

If any of the intermediate operations fails, an UNDECIDED conformity response is issued.

Expected results: If audit data is supported, then the call to BioSPI_Enroll returns an audit data BIR and that BIR has a processed level of RAW.

Assertion language package

```
<package name="b40a5260-ef14-11d9-a4fe-0002a5d5c51b">
        <author>
                ISO/IEC JTC1 SC37
        </author>

        <description>
                This package contains the assertion "BioSPI_Enroll_AuditData" (see the "description"
                                element of the assertion below).
        </description>

        <assertion name="BioSPI_Enroll_AuditData" model="BSPTesting">
                <description>
                        This assertion calls the function BioSPI_Enroll with the 'AuditData' parameter. The
                                        function is expected to return audit data if the BSP supports audit
                                        data.
                        The relevant text in BioAPI 2.0 is quoted below from subclauses 8.4.7.1
                        _______________________________________________________________
                        BioAPI_RETURN BioAPI BioSPI_Enroll
                                (BioAPI_HANDLE  BSPHandle,
                                BioAPI_BIR_PURPOSE  Purpose,
                                BioAPI_BIR_SUBTYPE  Subtype,
                                const BioAPI_BIR_BIOMETRIC_DATA_FORMAT  *OutputFormat,
                                const BioAPI_INPUT_BIR  *ReferenceTemplate,
                                BioAPI_BIR_HANDLE  *NewTemplate,
                                const BioAPI_DATA  *Payload,
                                int32_t  Timeout,
                                BioAPI_BIR_HANDLE  *AuditData,
                                BioAPI_UUID  *TemplateUUID);
```

```
      This function captures biometric data from the attached device (sensor unit) for
                  the purpose of creating a ProcessedBIR for the purpose of enrollment.

      ____________________________________________________________________________
      Subclause 8.4.7.1:
      Parameters: AuditData (output/optional) - A handle to a BIR containing raw
                  biometric data. This data may be used to provide a human-identifiable
                  data of the person at the sensor unit. If the pointer is NULL on input,
                  no audit data is collected. Not all BSPs support the collection of audit
                  data. A BSP may return a BIR handle value of
                  BioAPI_UNSUPPORTED_BIR_HANDLE to indicate AuditData is not supported, or
                  a value of BioAPI_INVALID_BIR_HANDLE to indicate that no audit data is
                  available.

      Subclause 7.47: BioAPI_OPTIONS_MASK
      #define BioAPI_RAW (0x00000001)
      If set, indicates that the BSP supports the return of raw/audit data.

      Subclause A.4.6.2.1
      Return of raw data. Functions involving the capture of biometric data from a sensor
                  may optionally support the return of this raw data for purposes of
                  display or audit. If supported, the output parameter AuditData will
                  contain a pointer to this data. If not supported, the BSP will return a
                  value of -1.
      ____________________________________________________________________________

      In order to determine conformance with respect to the text above, the following
                  steps are performed:

            1)    Load the BSP under test
            2)    Attach the BSP under test
            3)    Call BioSPI_Enroll to capture and enroll a BIR with AuditData set to a
                  non-NULL value.
            4)    Check the return code.  If audit data is not supported, the returned BIR
                  handle is expected to be BioAPI_UNSUPPORTED_BIR_HANDLE.  If audit data
                  is not available, the returned BIR handle is expected to be
                  BioAPI_INVALID_BIR_HANDLE.
            5)    Invoke BioSPI_GetHeaderFromHandle to check the type of the audit data,
                  if supported; it is expected to be RAW.

      If any of the intermediate operations fails, an UNDECIDED conformity response is
                  issued.
</description>

<!-- UUID of the BSP under test -->
<input name="_bspUuid"/>

<!-- Timeout for the BioAPI_NOTIFY_INSERT event -->
<input name="_inserttimeout"/>

<!-- Indicates whether the BSP under test does not claim support for the
                  BioAPI_NOTIFY_SOURCE_PRESENT event notification -->
<input name="_noSourcePresentSupported" />

<!-- Timeout for the BioAPI_NOTIFY_SOURCE_PRESENT event -->
<input name="_sourcepresenttimeout"/>

<!-- Timeout for BioSPI_Enroll -->
<input name="_capturetimeout"/>

<!-- Indicates whether the BSP under test claims supports for audit data -->
<input name="_supportAuditData" />

<!-- Invocation of the primary activity of this assertion with input parameter values
                  assigned from the assertion's parameters. -->
<invoke activity="BioSPI_Enroll">
      <input name="bspUuid" var="_bspUuid"/>
      <input name="inserttimeouttime" var="_inserttimeout"/>
      <input name="nosourcepresentsupported"
                  var="_noSourcePresentSupported" />
      <input name="sourcepresenttimeouttime"
                  var="_sourcepresenttimeout"/>
      <input name="capturetimeouttime" var="_capturetimeout"/>
      <input name="supportAuditData" var="_supportAuditData" />
</invoke>
```

```
        <!-- Activity bound to a function of the framework callback interface exposed by the
                          testing component.  This activity will be automatically invoked on each
                          incoming call to the function to which it is bound. -->
        <bind activity="EventHandler"
                    package="02c59458-0c46-1085-95d7-0002a5d5fd2e"
                    function="BioSPI_EventHandler"/>
  </assertion>

  <activity name="BioSPI_Enroll">
        <input name="bspUuid"/>
        <input name="inserttimeouttime"/>
        <input name="nosourcepresentsupported" />
        <input name="sourcepresenttimeouttime"/>
        <input name="capturetimeouttime"/>
        <input name="supportAuditData" />

        <!-- This assertion will use BSPHandle "1" for all BioSPI calls that require it -->
        <set name="_bsphandle" value="1"/>

        <!-- Invoke the functions BioSPI_BSPLoad and BioSPI_BSPAttach exposed by the BSP under
                          test.
                    The input value for the parameter "unitIDOrNull" is "0", therefore the
                          assertion will test a sensor unit chosen by the BSP. -->
        <invoke activity="LoadAndAttach"
                    package="02c59458-0c46-1085-95d7-0002a5d5fd2e"
                    break_on_break="true">
              <input name="bspUuid" var="bspUuid"/>
              <input name="bspVersion" value="32"/>
              <input name="unitIDOrNull" value="0"/>
              <input name="bspHandle" var="_bsphandle"/>
              <input name="eventtimeouttime" var="inserttimeouttime"/>
        </invoke>

        <set name="eventtimeoutflag" value="false"/>

        <!-- If the BSP under test claims support for the BioAPI_NOTIFY_SOURCE_PRESENT event
                          notification, wait until that notification has been received, but no
                          longer than the specified maximum duration.-->
        <wait_until timeout_var="sourcepresenttimeouttime"
                    setvar="eventtimeoutflag">
              <or var1="nosourcepresentsupported" var2="_sourcePresent" />
        </wait_until>

        <!-- Issue a conformity response.
                    If the condition specified in the <description> below is false, an UNDECIDED
                          conformity response is issued and the execution of the activity is
                          interrupted, otherwise a PASS conformity response is issued.-->
        <assert_condition response_if_false="undecided"
                    break_if_false="true">
              <description>
                    Either the BSP under test does not claim support for the
                          BioAPI_NOTIFY_SOURCE_PRESENT event notification, or the event
                          notification has been received within the specified maximum duration.
              </description>
              <not var="eventtimeoutflag"/>
        </assert_condition>

        <!-- The BSP is ready to capture. Invoke the function BioSPI_Enroll for the purpose of
                          enrollment. -->
        <invoke function="BioSPI_Enroll">
              <input name="BSPHandle" var="_bsphandle"/>
              <input name="Purpose"
                          var="__BioAPI_PURPOSE_ENROLL_FOR_VERIFICATION_ONLY"/>
              <input name="Subtype" value="0"/>
              <input name="Timeout" var="capturetimeouttime"/>
              <output name="AuditData" setvar="auditbir_handle" />
              <output name="NewTemplate" setvar="newtemplate_handle"/>
              <return setvar="return"/>
        </invoke>

        <!-- Issue a conformity response.
                    If the condition specified in the <description> below is false, an UNDECIDED
                          conformity response is issued and the execution of the activity is
                          interrupted, otherwise a PASS conformity response is issued.-->
        <assert_condition response_if_false="undecided"
                    break_if_false="true">
```

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

      <!-- Issue a conformity response.
                  If the condition specified in the <description> below is false, a FAIL
                        conformity response is issued and the execution of the activity is
                        interrupted, otherwise a PASS conformity response is issued.-->
      <assert_condition break_if_false="true">
            <description>
                  If audit data is supported, the output AuditData BIR handle is either a valid
                        BIR handle or BioAPI_INVALID_BIR_HANDLE; If audit data is not supported,
                        the output AuditData BIR handle is BioAPI_UNSUPPORTED_BIR_HANDLE.
            </description>
            <or>
                  <and>
                        <same_as var1="supportAuditData" value2="true" />
                        <or>
                              <equal_to var1="auditbir_handle"
                                          var2="__BioAPI_INVALID_BIR_HANDLE" />
                              <greater_than_or_equal_to var1="auditbir_handle" value2="0" />
                        </or>
                  </and>
                  <and>
                        <same_as var1="supportAuditData" value2="false" />
                        <equal_to var1="auditbir_handle"
                                    var2="__BioAPI_UNSUPPORTED_BIR_HANDLE" />
                  </and>
            </or>
      </assert_condition>

      <!-- Check the processed level of the audit data BIR -->
      <invoke activity="check_audit_data_type" >
            <only_if>
                  <same_as var1="supportAuditData" value2="true" />
                  <greater_than_or_equal_to var1="auditbir_handle" value2="0" />
            </only_if>
            <input name="BSPHandle" var="_bsphandle"/>
            <input name="Handle" var="auditbir_handle"/>
      </invoke>

      <!-- Invoke the functions BioSPI_BSPDetach and BioSPI_BSPUnload -->
      <invoke activity="DetachAndUnload"
                  package="02c59458-0c46-1085-95d7-0002a5d5fd2e" >
            <input name="bspUuid" var="bspUuid" />
            <input name="BSPHandle" var="_bsphandle" />
      </invoke>
</activity>

<activity name="check_audit_data_type" >
      <input name="BSPHandle" />
      <input name="Handle" />

      <!-- Invoke the function BioSPI_GetHeaderFromHandle. -->
      <invoke function="BioSPI_GetHeaderFromHandle">
            <input name="BSPHandle" var="_bsphandle"/>
            <input name="Handle" var="Handle"/>
            <output name="ProcessedLevel" setvar="processedLevel"/>
            <return setvar="return"/>
      </invoke>

      <!-- Issue a conformity response.
                  If the condition specified in the <description> below is false, an UNDECIDED
                        conformity response is issued and the execution of the activity is
                        interrupted, otherwise a PASS conformity response is issued.-->
      <assert_condition response_if_false="undecided"
                  break_if_false="true">
            <description>
                        The function BioSPI_GetHeaderFromHandle has returned BioAPI_OK.
            </description>
            <equal_to var1="return" var2="__BioAPI_OK"/>
      </assert_condition>
```

```
            <!-- Issue a conformity response.
                        If the condition specified in the <description> below is false, a FAIL
                              conformity response is issued, otherwise a PASS conformity response is
                              issued.-->
            <assert_condition>
                  <description>
                        The processed level of the audit data BIR is RAW.
                  </description>
                  <equal_to var1="processedLevel" var2="__BioAPI_BIR_DATA_TYPE_RAW"/>
            </assert_condition>
      </activity>

</package>
```

8.49 Assertion 13d - *BioSPI_Enroll_BIRHeaderQuality*

Description: This assertion checks if calling the function BioSPI_Enroll returns a valid quality value in the new template BIR's header.

Excerpts

Subclause 9.3.4.7

BioAPI_RETURN BioAPI BioSPI_Enroll

(BioAPI_HANDLE BSPHandle,

BioAPI_BIR_PURPOSE Purpose,

BioAPI_BIR_SUBTYPE Subtype,

*const BioAPI_BIR_BIOMETRIC_DATA_FORMAT *OutputFormat,*

*const BioAPI_INPUT_BIR *ReferenceTemplate,*

*BioAPI_BIR_HANDLE *NewTemplate,*

*const BioAPI_DATA *Payload,*

int32_t Timeout,

*BioAPI_BIR_HANDLE *AuditData,*

*BioAPI_UUID *TemplateUUID);*

Subclause 8.4.7.1

This function captures biometric data from the attached device (sensor unit) for the purpose of creating a ProcessedBIR for the purpose of enrollment.

Subclause 7.49.3

Quality measurements are reported as an integral value in the range 0-100 except as follows:

Value of -1: BioAPI_QUALITY has not been set by the BSP (reference BSP vendor's documentation for explanation).

Value of -2: BioAPI_QUALITY is not supported by the BSP.

Subclause 7.47

#define BioAPI_QUALITY_INTERMEDIATE (0x00000004)

If set, BSP supports the return of a quality value (in the BIR header) for intermediate biometric data.

#define BioAPI_QUALITY_PROCESSED (0x00000008)

If set, BSP supports the return of quality value (in the BIR header) for processed biometric data.

Subclause A.4.6.2.2

Return of Quality.

Upon the new capture of biometric data from a sensor, the BSP may calculate a relative quality value associated with this data, which it will include in the header of the returned CapturedBIR (and the optional AuditData). If supported, this header field will be filled with a positive value between '1' and '100'. If not supported, this field will be set to '-2'. This would occur during BioSPI_Capture and BioSPI_Enroll.

Similarly, when a BIR is processed, another quality calculation may be performed and the quality value included in the header of the ProcessedBIR (and the optional AdaptedBIR). This would occur during BioSPI_CreateTemplate, BioSPI_Process, BioSPI_ProcessWithAuxBIR, BioSPI_Verify, BioSPI_VerifyMatch, BioSPI_Enroll, and BioSPI_Import (ConstructedBIR) operations.

The BSP shall post to the component registry whether or not it supports the calculation of quality measurements for each type of BIR - raw, intermediate, and processed.

References: 9.3.4.7, 8.4.7.1, 7.49.3, 7.47, and A.4.6.2.2

Scenario:

1) Load the BSP under test.

2) Attach the BSP under test.

3) Call BioSPI_Enroll.

4) Call BioSPI_GetHeaderFromHandle for the new template BIR. Check the Quality field, which is expected to be in the range 0-100.

If any of the intermediate operations fails, an UNDECIDED conformity response is issued.

Expected results: The BIR generated by BioSPI_Enroll has a valid quality field.

Assertion language package

```
<package name="6f727320-ef1a-11d9-9143-0002a5d5c51b">
      <author>
            ISO/IEC JTC1 SC37
      </author>

      <description>
            This package contains the assertion "BioSPI_Enroll_BIRHeaderQuality" (see the
                              "description" element of the assertion below).
      </description>

      <assertion name="BioSPI_Enroll_BIRHeaderQuality" model="BSPTesting">
            <description>
                  This assertion checks if calling the function BioSPI_Enroll returns a valid quality
                              value in the new template BIR's header.
                  The relevant text in BioAPI 2.0 is quoted below from subclauses 7.49.3 and 8.4.7.1.
            _____________________________________________________________
            BioAPI_RETURN BioAPI BioSPI_Enroll
                  (BioAPI_HANDLE  BSPHandle,
                  BioAPI_BIR_PURPOSE  Purpose,
                  BioAPI_BIR_SUBTYPE  Subtype,
                  const BioAPI_BIR_BIOMETRIC_DATA_FORMAT  *OutputFormat,
                  const BioAPI_INPUT_BIR  *ReferenceTemplate,
                  BioAPI_BIR_HANDLE  *NewTemplate,
                  const BioAPI_DATA  *Payload,
                  int32_t  Timeout,
                  BioAPI_BIR_HANDLE  *AuditData,
                  BioAPI_UUID  *TemplateUUID);

            This function captures biometric data from the attached device (sensor unit) for
                        the purpose of creating a ProcessedBIR for the purpose of enrollment.

            Subclause 7.49.3:
            Quality measurements are reported as an integral value in the range 0-100 except as
                        follows:
            Value of -1: BioAPI_QUALITY has not been set by the BSP (reference BSP vendor's
                        documentation for explanation).
            Value of -2: BioAPI_QUALITY is not supported by the BSP.
            _____________________________________________________________
            Subclause 7.47:
            #define BioAPI_QUALITY_INTERMEDIATE (0x00000004)
            If set, BSP supports the return of a quality value (in the BIR header) for
                        intermediate biometric data.
            #define BioAPI_QUALITY_PROCESSED (0x00000008)
            If set, BSP supports the return of quality value (in the BIR header) for processed
                        biometric data.
            Subclause A.4.6.2.2:
            Return of Quality.
            Upon the new capture of biometric data from a sensor, the BSP may calculate a
                        relative quality value associated with this data, which it will include
                        in the header of the returned CapturedBIR (and the optional AuditData).
                        If supported, this header field will be filled with a positive value
                        between '1' and '100'. If not supported, this field will be set to '-2'.
                        This would occur during BioSPI_Capture and BioSPI_Enroll.
            Similarly, when a BIR is processed, another quality calculation may be performed
                        and the quality value included in the header of the ProcessedBIR (and
                        the optional AdaptedBIR). This would occur during BioSPI_CreateTemplate,
                        BioSPI_Process, BioSPI_ProcessWithAuxBIR, BioSPI_Verify,
                        BioSPI_VerifyMatch, BioSPI_Enroll, and BioSPI_Import (ConstructedBIR)
                        operations.
            The BSP shall post to the component registry whether or not it supports the
                        calculation of quality measurements for each type of BIR - raw,
                        intermediate, and processed.
            _____________________________________________________________

            In order to determine conformance with respect to the text above, the following
                        steps are performed:

                  1)    Load the BSP under test.
                  2)    Attach the BSP under test.
                  3)    Call BioSPI_Enroll.
                  4)    Call BioSPI_GetHeaderFromHandle for the new template BIR.  Check the
                        Quality field, which is expected to be in the range 0-100.

            If any of the intermediate operations fails, an UNDECIDED conformity response is
                        issued.
      </description>

      <!-- UUID of the BSP under test -->
      <input name="_bspUuid"/>

      <!-- Timeout for the BioAPI_NOTIFY_INSERT event -->
      <input name="_inserttimeout"/>

      <!-- Indicates whether the BSP under test does not claim support for the
                        BioAPI_NOTIFY_SOURCE_PRESENT event notification -->
      <input name="_noSourcePresentSupported"/>

      <!-- Timeout for the BioAPI_NOTIFY_SOURCE_PRESENT event -->
      <input name="_sourcepresenttimeout"/>
```

```
        <!-- Timeout for BioSPI_Enroll -->
        <input name="_capturetimeout"/>

        <!-- Indicates whether the BSP under test claims support for return of quality in a
                          processed BIR -->
        <input name="_processedQualitySupported"/>

        <!-- Invocation of the primary activity of this assertion with input parameter values
                          assigned from the assertion's parameters. -->
        <invoke activity="BioSPI_Enroll">
              <input name="bspUuid" var="_bspUuid"/>
              <input name="inserttimeouttime" var="_inserttimeout"/>
              <input name="nosourcepresentsupported"
                          var="_noSourcePresentSupported"/>
              <input name="sourcepresenttimeouttime"
                          var="_sourcepresenttimeout"/>
              <input name="capturetimeouttime" var="_capturetimeout"/>
              <input name="processedQualitySupported"
                          var="_processedQualitySupported"/>
        </invoke>

        <!-- Activity bound to a function of the framework callback interface exposed by the
                          testing component.  This activity will be automatically invoked on each
                          incoming call to the function to which it is bound. -->
        <bind activity="EventHandler"
                    package="02c59458-0c46-1085-95d7-0002a5d5fd2e"
                    function="BioSPI_EventHandler"/>
  </assertion>

  <activity name="BioSPI_Enroll">
        <input name="bspUuid"/>
        <input name="inserttimeouttime"/>
        <input name="nosourcepresentsupported"/>
        <input name="sourcepresenttimeouttime"/>
        <input name="capturetimeouttime"/>
        <input name="processedQualitySupported"/>

        <!-- This assertion will use BSPHandle "1" for all BioSPI calls that require it -->
        <set name="_bsphandle" value="1"/>

        <!-- Invoke the functions BioSPI_BSPLoad and BioSPI_BSPAttach exposed by the BSP under
                          test.
                    The input value for the parameter "unitIDOrNull" is "0", therefore the
                          assertion will test a sensor unit chosen by the BSP. -->
        <invoke activity="LoadAndAttach"
                    package="02c59458-0c46-1085-95d7-0002a5d5fd2e"
                    break_on_break="true">
              <input name="bspUuid" var="bspUuid"/>
              <input name="bspVersion" value="32"/>
              <input name="unitIDOrNull" value="0"/>
              <input name="bspHandle" var="_bsphandle"/>
              <input name="eventtimeouttime" var="inserttimeouttime"/>
        </invoke>

        <set name="eventtimeoutflag" value="false"/>

        <!-- If the BSP under test claims support for the BioAPI_NOTIFY_SOURCE_PRESENT event
                          notification, wait until that notification has been received, but no
                          longer than the specified maximum duration.-->
        <wait_until timeout_var="sourcepresenttimeouttime"
                    setvar="eventtimeoutflag">
              <or var1="nosourcepresentsupported" var2="_sourcePresent"/>
        </wait_until>
```

```
<!-- Issue a conformity response.
            If the condition specified in the <description> below is false, an UNDECIDED
                  conformity response is issued and the execution of the activity is
                  interrupted, otherwise a PASS conformity response is issued.-->
<assert_condition response_if_false="undecided"
            break_if_false="true">
      <description>
            Either the BSP under test does not claim support for the
                  BioAPI_NOTIFY_SOURCE_PRESENT event notification, or the event
                  notification has been received within the specified maximum duration.
      </description>
      <not var="eventtimeoutflag"/>
</assert_condition>

<!-- The BSP is ready to capture. Invoke the function BioSPI_Enroll for the purpose of
                  enrollment. -->
<invoke function="BioSPI_Enroll">
      <input name="BSPHandle" var="_bsphandle"/>
      <input name="Purpose"
                  var="__BioAPI_PURPOSE_ENROLL_FOR_VERIFICATION_ONLY"/>
      <input name="Subtype" value="0"/>
      <input name="Timeout" var="capturetimeouttime"/>
      <output name="NewTemplate" setvar="newtemplate_handle"/>
      <return setvar="return"/>
</invoke>

<!-- Issue a conformity response.
            If the condition specified in the <description> below is false, an UNDECIDED
                  conformity response is issued and the execution of the activity is
                  interrupted, otherwise a PASS conformity response is issued.-->
<assert_condition response_if_false="undecided"
            break_if_false="true">
      <description>
            The function BioSPI_Enroll has returned BioAPI_OK
      </description>
      <equal_to var1="return" var2="__BioAPI_OK"/>
</assert_condition>

<!-- Invoke the function BioSPI_GetHeaderFromHandle to check the processed level of the
                  new template BIR -->
<invoke function="BioSPI_GetHeaderFromHandle">
      <input name="BSPHandle" var="_bsphandle"/>
      <input name="Handle" var="newtemplate_handle"/>
      <output name="Quality" setvar="quality"/>
      <return setvar="return"/>
</invoke>

<!-- Issue a conformity response.
            If the condition specified in the <description> below is false, an UNDECIDED
                  conformity response is issued and the execution of the activity is
                  interrupted, otherwise a PASS conformity response is issued.-->
<assert_condition response_if_false="undecided"
            break_if_false="true">
      <description>
            The function BioSPI_GetHeaderFromHandle has returned BioAPI_OK.
      </description>
      <equal_to var1="return" var2="__BioAPI_OK"/>
</assert_condition>

<invoke activity="check_quality_supported"
            package="02c59458-0c46-1085-95d7-0002a5d5fd2e"
            break_on_break="true">
      <only_if>
            <same_as var1="processedQualitySupported" value2="true" />
      </only_if>
      <input name="quality" var="quality" />
</invoke>
```

```
            <invoke activity="check_quality_not_supported"
                        package="02c59458-0c46-1085-95d7-0002a5d5fd2e"
                        break_on_break="true" >
                  <only_if>
                        <same_as var1="processedQualitySupported"
                                    value2="false" />
                  </only_if>
                  <input name="quality" var="quality" />
            </invoke>

            <!-- Invoke the functions BioSPI_BSPDetach and BioSPI_BSPUnload -->
            <invoke activity="DetachAndUnload"
                        package="02c59458-0c46-1085-95d7-0002a5d5fd2e" >
                  <input name="bspUuid" var="bspUuid"/>
                  <input name="BSPHandle" var="_bsphandle"/>
            </invoke>
      </activity>

</package>
```

8.50 Assertion 14a - *BioSPI_Verify_ValidParam*

Description: This assertion checks if calling the function BioSPI_Verify with valid input parameters returns BioAPI_OK.

Excerpts

Subclause 9.3.4.8

BioAPI_RETURN BioAPI BioSPI_Verify

(BioAPI_HANDLE BSPHandle,

BioAPI_FMR MaxFMRRequested,

*const BioAPI_INPUT_BIR *ReferenceTemplate,*

BioAPI_BIR_SUBTYPE Subtype,

*BioAPI_BIR_HANDLE *AdaptedBIR,*

*BioAPI_BOOL *Result,*

*BioAPI_FMR *FMRAchieved,*

*BioAPI_DATA *Payload,*

int32_t Timeout,

*BioAPI_BIR_HANDLE *AuditData);*

Subclause 8.4.8.1

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 for a successful match. The Boolean Result indicates whether verification was successful or not, and the FMRAchieved is a FMR value indicating how closely the BIRs actually matched

Subclause A.4

This function should be supported by both Verification and Identification BSPs.

References: 9.3.4.8, 8.4.8.1, and A.4

Scenario:

1) Load the BSP under test.

2) Attach the BSP under test.

3) Call BioSPI_Enroll to create a template.

4) Call BioSPI_Verify without adaptation and audit data.

5) Check the return code.

If any of the intermediate operations fails, an UNDECIDED conformity response is issued.

Expected results: The call to BioSPI_Verify returns BioAPI_OK.

Assertion language package

```
<package name="b78e5be0-efcb-11d9-b2c7-0002a5d5c51b">
      <author>
            ISO/IEC JTC1 SC37
      </author>

      <description>
            This package contains the assertion "BioSPI_Verify_ValidParam" (see the "description"
                        element of the assertion below)
      </description>

      <assertion name="BioSPI_Verify_ValidParam" model="BSPTesting">
            <description>
                  This assertion checks if calling the function BioSPI_Verify with valid input
                        parameters returns BioAPI_OK.
                  The relevant text in BioAPI 2.0 is quoted below from subclauses A.4 and 8.4.8.1
                  ________________________________________________________
                  BioAPI_RETURN BioAPI BioSPI_Verify
                        (BioAPI_HANDLE  BSPHandle,
                        BioAPI_FMR  MaxFMRRequested,
                        const BioAPI_INPUT_BIR  *ReferenceTemplate,
                        BioAPI_BIR_SUBTYPE  Subtype,
                        BioAPI_BIR_HANDLE  *AdaptedBIR,
                        BioAPI_BOOL  *Result,
                        BioAPI_FMR  *FMRAchieved,
                        BioAPI_DATA  *Payload,
                        int32_t  Timeout,
                        BioAPI_BIR_HANDLE  *AuditData);

                  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 for a successful match. The Boolean Result
                  indicates whether verification was successful or not, and the FMRAchieved is a FMR
                        value indicating how closely the BIRs actually matched
                  _______________________________________________________
                  Subclause A.4:
                  This function should be supported by both Verification and Identification BSPs.
                  _______________________________________________________

                  In order to determine conformance with respect to the text above, the following
                        steps are performed:

                        1)    Load the BSP under test.
                        2)    Attach the BSP under test.
                        3)    Call BioSPI_Enroll to create a template.
                        4)    Call BioSPI_Verify without adaptation and audit data.
                        5)    Check the return code.

                  If any of the intermediate operations fails, an UNDECIDED conformity response is
                        issued.
            </description>

            <!-- UUID of the BSP under test -->
            <input name="_bspUuid"/>
```

```
        <!-- Timeout for the BioAPI_NOTIFY_INSERT event -->
        <input name="_inserttimeout"/>

        <!-- Indicates whether the BSP under test does not claim support for the
                          BioAPI_NOTIFY_SOURCE_PRESENT event notification -->
        <input name="_noSourcePresentSupported"/>

        <!-- Timeout for the BioAPI_NOTIFY_SOURCE_PRESENT event -->
        <input name="_sourcepresenttimeout"/>

        <!-- Timeout for BioSPI_Enroll -->
        <input name="_capturetimeout"/>

        <!-- MaxFMRRequested for BioSPI_Verify -->
        <input name="_maxFMRRequested"/>

        <!-- Timeout for BioSPI_Verify -->
        <input name="_verifytimeout"/>

        <!-- Invocation of the primary activity of this assertion with input parameter values
                          assigned from the assertion's parameters. -->
        <invoke activity="BioSPI_Verify">
              <input name="bspUuid" var="_bspUuid"/>
              <input name="inserttimeouttime" var="_inserttimeout"/>
              <input name="nosourcepresentsupported"
                          var="_noSourcePresentSupported"/>
              <input name="sourcepresenttimeouttime"
                          var="_sourcepresenttimeout"/>
              <input name="capturetimeouttime" var="_capturetimeout"/>
              <input name="maxFMRRequested" var="_maxFMRRequested"/>
              <input name="verifytimeout" var="_verifytimeout"/>
        </invoke>

        <!-- Activity bound to a function of the framework callback interface exposed by the
                          testing component.  This activity will be automatically invoked on each
                          incoming call to the function to which it is bound. -->
        <bind activity="EventHandler"
                    package="02c59458-0c46-1085-95d7-0002a5d5fd2e"
                    function="BioSPI_EventHandler"/>
  </assertion>

  <activity name="BioSPI_Verify">
        <input name="bspUuid"/>
        <input name="inserttimeouttime"/>
        <input name="nosourcepresentsupported"/>
        <input name="sourcepresenttimeouttime"/>
        <input name="capturetimeouttime"/>
        <input name="maxFMRRequested"/>
        <input name="verifytimeout"/>

        <!-- This assertion will use BSPHandle "1" for all BioSPI calls that require it -->
        <set name="_bsphandle" value="1"/>

        <!-- Invoke the functions BioSPI_BSPLoad and BioSPI_BSPAttach exposed by the BSP under
                          test.
                    The input value for the parameter "unitIDOrNull" is "0", therefore the
                          assertion will test a sensor unit chosen by the BSP. -->
        <invoke activity="LoadAndAttach"
                    package="02c59458-0c46-1085-95d7-0002a5d5fd2e"
                    break_on_break="true">
              <input name="bspUuid" var="bspUuid"/>
              <input name="bspVersion" value="32"/>
              <input name="unitIDOrNull" value="0"/>
              <input name="bspHandle" var="_bsphandle"/>
              <input name="eventtimeouttime" var="inserttimeouttime"/>
        </invoke>

        <set name="eventtimeoutflag" value="false"/>

        <!-- If the BSP under test claims support for the BioAPI_NOTIFY_SOURCE_PRESENT event
                          notification, wait until that notification has been received, but no
                          longer than the specified maximum duration.-->
        <wait_until timeout_var="sourcepresenttimeouttime"
                    setvar="eventtimeoutflag">
              <or var1="nosourcepresentsupported" var2="_sourcePresent"/>
        </wait_until>
```

```
            <!-- Issue a conformity response.
                        If the condition specified in the <description> below is false, an UNDECIDED
                              conformity response is issued and the execution of the activity is
                              interrupted, otherwise a PASS conformity response is issued.-->
            <assert_condition response_if_false="undecided"
                        break_if_false="true">
                  <description>
                        Either the BSP under test does not claim support for the
                              BioAPI_NOTIFY_SOURCE_PRESENT event notification, or the event
                              notification has been received within the specified maximum duration.
                  </description>
                  <not var="eventtimeoutflag"/>
            </assert_condition>

            <!-- Invoke the function BioSPI_Enroll to create a template.-->
            <invoke function="BioSPI_Enroll">
                  <input name="BSPHandle" var="_bsphandle"/>
                  <input name="Purpose"
                              var="__BioAPI_PURPOSE_ENROLL_FOR_VERIFICATION_ONLY"/>
                  <input name="Subtype" value="0"/>
                  <input name="Timeout" var="capturetimeouttime"/>
                  <output name="NewTemplate" setvar="template_handle"/>
                  <return setvar="return"/>
            </invoke>

            <!-- Issue a conformity response.
                        If the condition specified in the <description> below is false, a FAIL
                              conformity response is issued, otherwise a PASS conformity response is
                              issued.-->
            <assert_condition response_if_false="undecided" >
                  <description>
                        The function BioSPI_Enroll has returned BioAPI_OK
                  </description>
                  <equal_to var1="return" var2="__BioAPI_OK"/>
            </assert_condition>

            <!-- Invoke the function BioSPI_Verify to verify against the newly created template -->
            <invoke function="BioSPI_Verify" >
                  <input name="BSPHandle" var="_bsphandle"/>
                  <input name="MaxFMRRequested" var="maxFMRRequested"/>
                  <input name="ReferenceTemplate_Form"
                              var="__BioAPI_BIR_HANDLE_INPUT"/>
                  <input name="ReferenceTemplate_BIRHandle"
                              var="template_handle"/>
                  <input name="Subtype" value="0"/>
                  <input name="no_AdaptedBIR" value="true"/>
                  <input name="Timeout" var="verifytimeout"/>
                  <input name="no_AuditData" value="true"/>
                  <input name="no_Result" value="false"/>
                  <input name="no_Payload" value="false"/>
                  <output name="Result" setvar="result"/>
                  <output name="FMRAchieved" setvar="fmrAchieved"/>
                  <output name="Payload" setvar="payload"/>
                  <return setvar="return"/>
            </invoke>

            <!-- Issue a conformity response.
                        If the condition specified in the <description> below is false, a FAIL
                              conformity response is issued, otherwise a PASS conformity response is
                              issued.-->
            <assert_condition>
                  <description>
                        The function BioSPI_Verify has returned BioAPI_OK.
                  </description>
                  <equal_to var1="return" var2="__BioAPI_OK"/>
            </assert_condition>

            <!-- Invoke the functions BioSPI_BSPDetach and BioSPI_BSPUnload -->
            <invoke activity="DetachAndUnload"
                        package="02c59458-0c46-1085-95d7-0002a5d5fd2e" >
                  <input name="bspUuid" var="bspUuid"/>
                  <input name="BSPHandle" var="_bsphandle"/>
            </invoke>
      </activity>

</package>
```

8.51 Assertion 14b - *BioSPI_Verify_Payload*

Description: This assertion checks if calling the function BioSPI_Verify with a payload returns BioAPI_OK.

Excerpts

Subclause 9.3.4.8

BioAPI_RETURN BioAPI BioSPI_Verify

(BioAPI_HANDLE BSPHandle,

BioAPI_FMR MaxFMRRequested,

*const BioAPI_INPUT_BIR *ReferenceTemplate,*

BioAPI_BIR_SUBTYPE Subtype,

*BioAPI_BIR_HANDLE *AdaptedBIR,*

*BioAPI_BOOL *Result,*

*BioAPI_FMR *FMRAchieved,*

*BioAPI_DATA *Payload,*

int32_t Timeout,

*BioAPI_BIR_HANDLE *AuditData);*

Subclause 8.4.8.1

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 for a successful match. The Boolean Result indicates whether verification was successful or not, and the FMRAchieved is a FMR value indicating how closely the BIRs actually matched.

Parameters: Payload (output/optional) - If a payload is associated with the ReferenceTemplate, it is returned in an allocated BioAPI_DATA structure if the FMRAchieved satisfies the policy of the BSP.

Subclause 7.47

#define BioAPI_PAYLOAD (0x00000080)

If set, the BSP supports payload carry (accepts payload during Enroll/CreateTemplate and returns payload upon successful Verify/VerifyMatch).

Subclause A.4.6.2.6

If supported, BSPs shall post to the component registry the maximum payload size it can accommodate. A maximum size of 0 indicates payload carry is not supported. If input payloads exceed this size, an error shall be generated.

References: 9.3.4.8, 8.4.8.1, 7.47, and A.4.6.2.6

Scenario:

1) Load the BSP under test.

2) Attach the BSP under test.

3) Call the function BioSPI_Enroll to create a template using the specified payload.

4) Call the function BioSPI_Verify to verify the captured BIR against the stored template BIR.

5) Check the output parameter 'Payload'. It is expected to be the same as the input parameter 'payload' to BioSPI_CreateTemplate.

6) Detach and unload the BSP under test.

If any of the intermediate operations fails, an UNDECIDED conformity response is issued.

Expected results: The call to BioSPI_Verify returns BioAPI_OK and the payload returned is the same as the original payload.

Assertion language package

```
<package name="32969ec0-eff8-11d9-9831-0002a5d5c51b">
      <author>
            ISO/IEC JTC1 SC37
      </author>

      <description>
            This package contains the assertion "BioSPI_Verify_Payload" (see the "description"
                        element of the assertion below).
      </description>

      <assertion name="BioSPI_Verify_Payload" model="BSPTesting">
            <description>
                  This assertion checks if calling the function BioSPI_Verify with a payload returns
                              BioAPI_OK.
                  The relevant text in BioAPI 2.0 is quoted below from subclauses 7.47, A.4.6.2.6 and
                              8.4.8.1.
                  _______________________________________________________
                  BioAPI_RETURN BioAPI BioSPI_Verify
                        (BioAPI_HANDLE  BSPHandle,
                        BioAPI_FMR  MaxFMRRequested,
                        const BioAPI_INPUT_BIR  *ReferenceTemplate,
                        BioAPI_BIR_SUBTYPE  Subtype,
                        BioAPI_BIR_HANDLE  *AdaptedBIR,
                        BioAPI_BOOL  *Result,
                        BioAPI_FMR  *FMRAchieved,
                        BioAPI_DATA  *Payload,
                        int32_t  Timeout,
                        BioAPI_BIR_HANDLE  *AuditData);

                  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 for a successful match. The Boolean Result
                  indicates whether verification was successful or not, and the FMRAchieved is a FMR
                              value indicating how closely the BIRs actually matched
                  ______________________________________________________
                  Subclause 8.4.8.1:
                  Parameters: Payload (output/optional) - If a payload is associated with the
                              ReferenceTemplate, it is returned in an allocated BioAPI_DATA structure
                              if the FMRAchieved satisfies the policy of the BSP.
                  Subclause 7.47:
                  #define BioAPI_PAYLOAD (0x00000080)
                  If set, the BSP supports payload carry (accepts payload during
                              Enroll/CreateTemplate and returns payload upon successful
                              Verify/VerifyMatch).
```

```
          ___________________________________________________________
          Subclause A.4.6.2.6
          If supported, BSPs shall post to the component registry the maximum payload size it
                      can accommodate.  A maximum size of 0 indicates payload carry is not
                      supported.  If input payloads exceed this size, an error shall be
                      generated.
          ______________________________________________________

          In order to determine conformance with respect to the text above, the following
                      steps are performed:

                1)    Load the BSP under test.
                2)    Attach the BSP under test.
                3)    Call the function BioSPI_Enroll to create a template using the specified
                      payload.
                4)    Call the function BioSPI_Verify to verify the captured BIR against the
                      stored template BIR.
                5)    Check the output parameter 'Payload'.  It is expected to be the same as
                      the input parameter 'payload' to BioSPI_CreateTemplate.
                6)    Detach and unload the BSP under test.

          If any of the intermediate operations fails, an UNDECIDED conformity response is
                      issued.
    </description>

    <!-- UUID of the BSP under test -->
    <input name="_bspUuid"/>

    <!-- Timeout for the BioAPI_NOTIFY_INSERT event -->
    <input name="_inserttimeout"/>

    <!-- Indicates whether the BSP under test does not claim support for the
                      BioAPI_NOTIFY_SOURCE_PRESENT event notification -->
    <input name="_noSourcePresentSupported"/>

    <!-- Timeout for the BioAPI_NOTIFY_SOURCE_PRESENT event -->
    <input name="_sourcepresenttimeout"/>

    <!-- Timeout for BioSPI_Enroll -->
    <input name="_capturetimeout"/>

    <!-- MaxFMRRequested for BioSPI_VerifyMatch -->
    <input name="_maxFMRRequested"/>

    <!-- Timeout for BioSPI_Verify -->
    <input name="_verifytimeout"/>

    <!-- Indicates whether the BSP claims support for the payload.-->
    <input name="_payloadSupported"/>

    <!-- PayloadPolicy -->
    <input name="_payloadPolicy"/>

    <!-- Payload should satisfy the constraint on the payload size posted by the BSP in the
                      component registry -->
    <input name="_payload"/>

    <!-- Invocation of the primary activity of this assertion with input parameter values
                      assigned from the assertion's parameters. -->
    <invoke activity="BioSPI_Verify">
          <input name="bspUuid" var="_bspUuid"/>
          <input name="inserttimeouttime" var="_inserttimeout"/>
          <input name="nosourcepresentsupported"
                      var="_noSourcePresentSupported"/>
          <input name="sourcepresenttimeouttime"
                      var="_sourcepresenttimeout"/>
          <input name="capturetimeouttime" var="_capturetimeout"/>
          <input name="maxFMRRequested" var="_maxFMRRequested"/>
          <input name="verifytimeout" var="_verifytimeout"/>
          <input name="payloadSupported" var="_payloadSupported"/>
          <input name="payloadPolicy" var="_payloadPolicy"/>
          <input name="payload" var="_payload"/>
    </invoke>
```

```
      <!-- Activity bound to a function of the framework callback interface exposed by the
                  testing component.  This activity will be automatically invoked on each
                  incoming call to the function to which it is bound. -->
      <bind activity="EventHandler"
                  package="02c59458-0c46-1085-95d7-0002a5d5fd2e"
                  function="BioSPI_EventHandler"/>
</assertion>

<activity name="BioSPI_Verify">
      <input name="bspUuid"/>
      <input name="inserttimeouttime"/>
      <input name="nosourcepresentsupported"/>
      <input name="sourcepresenttimeouttime"/>
      <input name="capturetimeouttime"/>
      <input name="maxFMRRequested"/>
      <input name="verifytimeout"/>
      <input name="payloadSupported"/>
      <input name="payloadPolicy"/>
      <input name="payload"/>

      <!-- This assertion will use BSPHandle "1" for all BioSPI calls that require it -->
      <set name="_bsphandle" value="1"/>

      <!-- Invoke the functions BioSPI_BSPLoad and BioSPI_BSPAttach exposed by the BSP under
                  test.
            The input value for the parameter "unitIDOrNull" is "0", therefore the
                  assertion will test a sensor unit chosen by the BSP. -->
      <invoke activity="LoadAndAttach"
                  package="02c59458-0c46-1085-95d7-0002a5d5fd2e"
                  break_on_break="true">
            <input name="bspUuid" var="bspUuid"/>
            <input name="bspVersion" value="32"/>
            <input name="unitIDOrNull" value="0"/>
            <input name="bspHandle" var="_bsphandle"/>
            <input name="eventtimeouttime" var="inserttimeouttime"/>
      </invoke>

      <set name="eventtimeoutflag" value="false"/>
      <!-- If the BSP under test claims support for the BioAPI_NOTIFY_SOURCE_PRESENT event
                  notification, wait until that notification has been received, but no
                  longer than the specified maximum duration.-->
      <wait_until timeout_var="sourcepresenttimeouttime"
                  setvar="eventtimeoutflag">
            <or var1="nosourcepresentsupported" var2="_sourcePresent"/>
      </wait_until>

      <!-- Issue a conformity response.
            If the condition specified in the <description> below is false, an UNDECIDED
                  conformity response is issued and the execution of the activity is
                  interrupted, otherwise a PASS conformity response is issued.-->
      <assert_condition response_if_false="undecided"
                  break_if_false="true">
            <description>
                  Either the BSP under test does not claim support for the
                        BioAPI_NOTIFY_SOURCE_PRESENT event notification, or the event
                        notification has been received within the specified maximum duration.
            </description>
            <not var="eventtimeoutflag"/>
      </assert_condition>

      <!-- Invoke the function BioSPI_Enroll to create a template.-->
      <invoke function="BioSPI_Enroll">
            <input name="BSPHandle" var="_bsphandle"/>
            <input name="Purpose"
                  var="__BioAPI_PURPOSE_ENROLL_FOR_VERIFICATION_ONLY"/>
            <input name="Subtype" value="0"/>
            <input name="Timeout" var="capturetimeouttime"/>
            <input name="Payload"  var="payload"/>
            <output name="NewTemplate" setvar="template_handle"/>
            <return setvar="return"/>
      </invoke>
```

```
<!-- Issue a conformity response.
            If the condition specified in the <description> below is false, an UNDECIDED
                  conformity response is issued, otherwise a PASS conformity response is
                  issued.-->
<assert_condition response_if_false="undecided" >
      <description>
            The function BioSPI_Enroll has returned BioAPI_OK
      </description>
      <equal_to var1="return" var2="__BioAPI_OK"/>
</assert_condition>

<!-- Invoke the function BioSPI_Verify to capture and verifies against the stored
                  template -->
<invoke function="BioSPI_Verify" >
      <input name="BSPHandle" var="_bsphandle"/>
      <input name="MaxFMRRequested" var="maxFMRRequested"/>
      <input name="ReferenceTemplate_Form"
                  var="__BioAPI_BIR_HANDLE_INPUT"/>
      <input name="ReferenceTemplate_BIRHandle"
                  var="template_handle"/>
      <input name="Subtype" value="0"/>
      <input name="no_AdaptedBIR" value="true"/>
      <input name="Timeout" var="verifytimeout"/>
      <input name="no_AuditData" value="true"/>
      <input name="no_Result" value="false"/>
      <input name="no_Payload"  value="false"/>
      <output name="Result" setvar="result"/>
      <output name="AdaptedBIR" setvar="adaptedbir_handle"/>
      <output name="FMRAchieved" setvar="fmrAchieved"/>
      <output name="Payload" setvar="output_payload"/>
      <return setvar="return"/>
</invoke>

<!-- Issue a conformity response.
            If the condition specified in the <description> below is false, an UNDECIDED
                  conformity response is issued and the execution of the activity is
                  interrupted, otherwise a PASS conformity response is issued.-->
<assert_condition response_if_false="undecided"
            break_if_false="true" >
      <description>
            The function BioSPI_Verify has returned BioAPI_OK.
      </description>
      <equal_to var1="return" var2="__BioAPI_OK"/>
</assert_condition>

<!-- Check the value of the output parameter "payload" (BSP claiming to support the
                  payload) -->
<invoke activity="payloadSupport_checkPayload" package="02c59458-0c46-1085-95d7-
                  0002a5d5fd2e" break_on_break="true">
      <only_if>
            <same_as var1="payloadSupported" value2="true"/>
      </only_if>
      <input name="inputPayload" var="payload" />
      <input name="outputPayload" var="output_payload"/>
      <input name="result" var="result"/>
      <input name="payloadPolicy" var="payloadPolicy"/>
      <input name="fmrAchieved" var="fmrAchieved"/>
</invoke>

<!-- Check the value of the output parameter "payload" (BSP not claiming to support the
                  payload) -->
<invoke activity="payloadNotSupport_checkPayload" package="02c59458-0c46-1085-95d7-
                  0002a5d5fd2e" break_on_break="true">
      <only_if>
            <same_as var1="payloadSupported" value2="false" />
      </only_if>
      <input name="outputPayload"  var="output_payload" />
</invoke>
```

```
            <!-- Invoke the functions BioSPI_BSPDetach and BioSPI_BSPUnload -->
            <invoke activity="DetachAndUnload"
                    package="02c59458-0c46-1085-95d7-0002a5d5fd2e" >
                <input name="bspUuid" var="bspUuid"/>
                <input name="BSPHandle" var="_bsphandle"/>
            </invoke>
        </activity>

</package>
```

8.52 Assertion 14c - *BioSPI_Verify_AuditData*

Description: This assertion checks if calling BioSPI_Verify with audit data returns BioAPI_OK.

Excerpts

Subclause 9.3.4.8

BioAPI_RETURN BioAPI BioSPI_Verify

(BioAPI_HANDLE BSPHandle,

BioAPI_FMR MaxFMRRequested,

*const BioAPI_INPUT_BIR *ReferenceTemplate,*

BioAPI_BIR_SUBTYPE Subtype,

*BioAPI_BIR_HANDLE *AdaptedBIR,*

*BioAPI_BOOL *Result,*

*BioAPI_FMR *FMRAchieved,*

*BioAPI_DATA *Payload,*

int32_t Timeout,

*BioAPI_BIR_HANDLE *AuditData);*

Subclause 8.4.8.1

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 for a successful match. The Boolean Result indicates whether verification was successful or not, and the FMRAchieved is a FMR value indicating how closely the BIRs actually matched.

Parameters: AuditData (output/optional) - A handle to a BIR containing raw biometric data.

This data may be used to provide human-identifiable data of the person at the sensor unit. If the pointer is NULL on input, no audit data is collected. Not all BSPs support the collection of audit data. A BSP may return a BIR handle value of BioAPI_UNSUPPORTED_BIR_HANDLE to indicate AuditData is not supported, or a value of BioAPI_INVALID_BIR_HANDLE to indicate that no audit data is available.

A BSP may return a handle value of BioAPI_UNSUPPORTED_BIR_HANDLE to indicate AuditData is not supported, or a value of BioAPI_INVALID_BIR_HANDLE to indicate that no audit data is available.

Subclause 7.47

#define BioAPI_RAW (0x00000001)

If set, indicates that the BSP supports the return of raw/audit data.

Subclause A.4.6.2.1

Return of raw data. Functions involving the capture of biometric data from a sensor may optionally support the return of this raw data for purposes of display or audit. If supported, the output parameter AuditData will contain a pointer to this data. If not supported, the BSP will return a value of '-1'.

References: 9.3.4.8, 8.4.8.1, 7.47, and A.4.6.2.1

Scenario:

1) Load the BSP under test

2) Attach the BSP under test

3) Call BioSPI_Enroll to create a template

4) Call BioSPI_Verify to verify the captured BIR against the storedTemplate.

5) Check the return code. If audit data is not supported, the audit data BIR handle is expected to be BioAPI_UNSUPPORTED_BIR_HANDLE. If audit data is not available, the audit data BIR handle is expected to be BioAPI_INVALID_BIR_HANDLE.

6) Detach and unload the BSP under test.

If any of the intermediate operations fails, an UNDECIDED conformity response is issued.

Expected results: The call to BioSPI_Verify with audit data returns either BioAPI_UNSUPPORTED_BIR_HANDLE (if audit data is not supported) or BioAPI_OK or BioAPI_INVALID_BIR_HANDLE (if audit data is supported).

Assertion language package

```
<package name="89719700-f218-11d9-b028-0002a5d5c51b">
      <author>
            ISO/IEC JTC1 SC37
      </author>

      <description>
            This package contains the assertion "BioSPI_Verify_AuditData" (see the "description"
                              element of the assertion below).
      </description>

      <assertion name="BioSPI_Verify_AuditData" model="BSPTesting">
            <description>
                  This assertion checks if calling BioSPI_Verify with audit data returns BioAPI_OK.
                  The relevant text in BioAPI 2.0 is quoted below from subclauses 8.4.8.1, A.4.6.2.1
                              and 7.47.
                  ________________________________________________
                  BioAPI_RETURN BioAPI BioSPI_Verify
                        (BioAPI_HANDLE  BSPHandle,
                        BioAPI_FMR  MaxFMRRequested,
                        const BioAPI_INPUT_BIR  *ReferenceTemplate,
                        BioAPI_BIR_SUBTYPE  Subtype,
                        BioAPI_BIR_HANDLE  *AdaptedBIR,
                        BioAPI_BOOL  *Result,
                        BioAPI_FMR  *FMRAchieved,
                        BioAPI_DATA  *Payload,
                        int32_t  Timeout,
                        BioAPI_BIR_HANDLE  *AuditData);

                  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 for a successful match. The Boolean Result
                              indicates whether verification was successful or not, and the
                              FMRAchieved is a FMR value indicating how closely the BIRs actually
                              matched.
```

```
          ________________________________________________
          Subclause 8.4.8.1:
          Parameters: AuditData (output/optional) - A handle to a BIR containing raw
                      biometric data. This data may be used to provide human-identifiable data
                      of the person at the sensor unit. If the pointer is NULL on input, no
                      audit data is collected. Not all BSPs support the collection of audit
                      data. A BSP may return a BIR handle value of
                      BioAPI_UNSUPPORTED_BIR_HANDLE to indicate AuditData is not supported, or
                      a value of BioAPI_INVALID_BIR_HANDLE to indicate that no audit data is
                      available.
          A BSP may return a handle value of BioAPI_UNSUPPORTED_BIR_HANDLE to indicate
                      AuditData is not supported, or a value of BioAPI_INVALID_BIR_HANDLE to
                      indicate that no audit data is available.

          Subclause 7.47: BioAPI_OPTIONS_MASK
          #define BioAPI_RAW (0x00000001)
          If set, indicates that the BSP supports the return of raw/audit data.

          Subclause A.4.6.2.1:
          A.4.6.2.1 Return of raw data. Functions involving the capture of biometric data
                      from a sensor may optionally support the return of this raw data for
                      purposes of display or audit. If supported, the output parameter
                      AuditData will contain a pointer to this data. If not supported, the BSP
                      will return a value of '-1'.

          ________________________________________________

          In order to determine conformance with respect to the text above, the following
                      steps are performed:

                1)    Load the BSP under test
                2)    Attach the BSP under test
                3)    Call BioSPI_Enroll to create a template
                4)    Call BioSPI_Verify to verify the captured BIR against the
                      storedTemplate.
                5)    Check the return code. If audit data is not supported, the audit data
                      BIR handle is expected to be BioAPI_UNSUPPORTED_BIR_HANDLE.  If audit
                      data is not available, the audit data BIR handle is expected to be
                      BioAPI_INVALID_BIR_HANDLE.
                6)    Detach and unload the BSP under test.

          If any of the intermediate operations fails, an UNDECIDED conformity response is
                      issued.
    </description>

    <!-- UUID of the BSP under test -->
    <input name="_bspUuid"/>

    <!-- Timeout for the BioAPI_NOTIFY_INSERT event -->
    <input name="_inserttimeout"/>

    <!-- Indicates whether the BSP under test does not claim support for the
                      BioAPI_NOTIFY_SOURCE_PRESENT event notification -->
    <input name="_noSourcePresentSupported" />

    <!-- Timeout for the BioAPI_NOTIFY_SOURCE_PRESENT event -->
    <input name="_sourcepresenttimeout"/>

    <!-- Timeout for BioSPI_Enroll -->
    <input name="_capturetimeout"/>

    <!-- MaxFMRRequested for BioSPI_VerifyMatch -->
    <input name="_maxFMRRequested" />

    <!-- Timeout for BioSPI_Verify -->
    <input name="_verifytimeout" />

    <!-- Indicates whether the BSP under test claims support for audit data -->
    <input name="_supportAuditData" />

    <!-- Invocation of the primary activity of this assertion with input parameter values
                      assigned from the assertion's parameters. -->
    <invoke activity="BioSPI_Verify">
          <input name="bspUuid" var="_bspUuid"/>
          <input name="inserttimeouttime" var="_inserttimeout"/>
          <input name="nosourcepresentsupported"
```

```
                    var="_noSourcePresentSupported" />
            <input name="sourcepresenttimeouttime"
                    var="_sourcepresenttimeout"/>
            <input name="capturetimeouttime" var="_capturetimeout"/>
            <input name="maxFMRRequested" var="_maxFMRRequested"/>
            <input name="verifytimeout" var="_verifytimeout" />
            <input name="supportAuditData" var="_supportAuditData" />
        </invoke>

        <!-- Activity bound to a function of the framework callback interface exposed by the
                    testing component.  This activity will be automatically invoked on each
                    incoming call to the function to which it is bound. -->
        <bind activity="EventHandler"
                package="02c59458-0c46-1085-95d7-0002a5d5fd2e"
                function="BioSPI_EventHandler"/>
    </assertion>

    <activity name="BioSPI_Verify">
        <input name="bspUuid"/>
        <input name="inserttimeouttime"/>
        <input name="nosourcepresentsupported" />
        <input name="sourcepresenttimeouttime"/>
        <input name="capturetimeouttime"/>
        <input name="maxFMRRequested" />
        <input name="verifytimeout" />
        <input name="supportAuditData" />

        <!-- This assertion will use BSPHandle "1" for all BioSPI calls that require it -->
        <set name="_bsphandle" value="1"/>

        <!-- Invoke the functions BioSPI_BSPLoad and BioSPI_BSPAttach exposed by the BSP under
                    test.
                The input value for the parameter "unitIDOrNull" is "0", therefore the
                    assertion will test a sensor unit chosen by the BSP. -->
        <invoke activity="LoadAndAttach"
                package="02c59458-0c46-1085-95d7-0002a5d5fd2e"
                break_on_break="true">
            <input name="bspUuid" var="bspUuid"/>
            <input name="bspVersion" value="32"/>
            <input name="unitIDOrNull" value="0"/>
            <input name="bspHandle" var="_bsphandle"/>
            <input name="eventtimeouttime" var="inserttimeouttime"/>
        </invoke>

        <set name="eventtimeoutflag" value="false"/>

        <!-- If the BSP under test claims support for the BioAPI_NOTIFY_SOURCE_PRESENT event
                    notification, wait until that notification has been received, but no
                    longer than the specified maximum duration.-->
        <wait_until timeout_var="sourcepresenttimeouttime"
                setvar="eventtimeoutflag">
            <or var1="nosourcepresentsupported" var2="_sourcePresent" />
        </wait_until>

        <!-- Issue a conformity response.
                If the condition specified in the <description> below is false, an UNDECIDED
                    conformity response is issued and the execution of the activity is
                    interrupted, otherwise a PASS conformity response is issued.-->
        <assert_condition response_if_false="undecided"
                break_if_false="true">
            <description>
                Either the BSP under test does not claim support for the
                    BioAPI_NOTIFY_SOURCE_PRESENT event notification, or the event
                    notification has been received within the specified maximum duration.
            </description>
            <not var="eventtimeoutflag"/>
        </assert_condition>
```

```
<!-- Invoke the function BioSPI_Enroll to create a template.-->
<invoke function="BioSPI_Enroll">
      <input name="BSPHandle" var="_bsphandle"/>
      <input name="Purpose"
                  var="__BioAPI_PURPOSE_ENROLL_FOR_VERIFICATION_ONLY"/>
      <input name="Subtype" value="0"/>
      <input name="Timeout" var="capturetimeouttime"/>
      <output name="NewTemplate" setvar="template_handle"/>
      <return setvar="return"/>
</invoke>

<!-- Issue a conformity response.
            If the condition specified in the <description> below is false, an UNDECIDED
                  conformity response is issued, otherwise a PASS conformity response is
                  issued.-->
<assert_condition response_if_false="undecided" >
      <description>
            The function BioSPI_Enroll has returned BioAPI_OK
      </description>
      <equal_to var1="return" var2="__BioAPI_OK"/>
</assert_condition>

<!-- Invoke the function BioSPI_Verify to capture and verifies against the stored
                  template -->
<invoke function="BioSPI_Verify" >
      <input name="BSPHandle" var="_bsphandle"/>
      <input name="MaxFMRRequested" var="maxFMRRequested" />
      <input name="ReferenceTemplate_Form"
                  var="__BioAPI_BIR_HANDLE_INPUT" />
      <input name="ReferenceTemplate_BIRHandle"
                  var="template_handle" />
      <input name="Subtype" value="0"/>
      <input name="no_AdaptedBIR" value="true" />
      <input name="Timeout" var="verifytimeout" />
      <input name="no_AuditData" value="false" />
      <input name="no_Result" value="false" />
      <input name="no_Payload"  value="true" />
      <output name="Result" setvar="result" />
      <output name="AuditData" setvar="auditbir_handle" />
      <output name="FMRAchieved" setvar="fmrAchieved" />
      <return setvar="return"/>
</invoke>

<!-- Issue a conformity response.
            If the condition specified in the <description> below is false, an UNDECIDED
                  conformity response is issued and the execution of the activity is
                  interrupted, otherwise a PASS conformity response is issued.-->
<assert_condition  response_if_false="undecided"
            break_if_false="true" >
      <description>
            The function BioSPI_Verify has returned BioAPI_OK.
      </description>
      <equal_to var1="return" var2="__BioAPI_OK"/>
</assert_condition>

<!-- Issue a conformity response.
            If the condition specified in the <description> below is false, a FAIL
                  conformity response is issued and the execution of the activity is
                  interrupted, otherwise a PASS conformity response is issued.-->
<assert_condition break_if_false="true">
      <description>
            The output AuditData BIR handle is either a valid value or
                  BioAPI_INVALID_BIR_HANDLE (if audit data is supported), or
                  BioAPI_UNSUPPORTED_BIR_HANDLE (if audit data is not supported).
      </description>
      <or>
            <and>
                  <same_as var1="supportAuditData" value2="true" />
                  <or>
                        <equal_to var1="auditbir_handle"
                                    var2="__BioAPI_INVALID_BIR_HANDLE" />
                        <greater_than_or_equal_to var1="auditbir_handle" value2="0" />
                  </or>
            </and>
            <and>
                  <same_as var1="supportAuditData" value2="false" />
```

```
                        <equal_to var1="auditbir_handle"
                                    var2="__BioAPI_UNSUPPORTED_BIR_HANDLE" />
                  </and>
            </or>
      </assert_condition>

      <!-- Check the processed level of the audit data BIR -->
      <invoke activity="check_audit_data_type" >
            <only_if>
                  <same_as var1="supportAuditData" value2="true" />
                  <greater_than_or_equal_to var1="auditbir_handle" value2="0" />
            </only_if>
            <input name="BSPHandle" var="_bsphandle"/>
            <input name="Handle" var="auditbir_handle"/>
      </invoke>

      <!-- Invoke the functions BioSPI_BSPDetach and BioSPI_BSPUnload -->
      <invoke activity="DetachAndUnload"
                  package="02c59458-0c46-1085-95d7-0002a5d5fd2e" >
            <input name="bspUuid" var="bspUuid" />
            <input name="BSPHandle" var="_bsphandle" />
      </invoke>
</activity>

<activity name="check_audit_data_type" >
      <input name="BSPHandle" />
      <input name="Handle" />

      <!-- Invoke the function BioSPI_GetHeaderFromHandle. -->
      <invoke function="BioSPI_GetHeaderFromHandle">
            <input name="BSPHandle" var="_bsphandle"/>
            <input name="Handle" var="Handle"/>
            <output name="ProcessedLevel" setvar="processedLevel"/>
            <return setvar="return"/>
      </invoke>

      <!-- Issue a conformity response.
                  If the condition specified in the <description> below is false, an UNDECIDED
                        conformity response is issued and the execution of the activity is
                        interrupted, otherwise a PASS conformity response is issued.-->
      <assert_condition response_if_false="undecided"
                  break_if_false="true">
            <description>
                        The function BioSPI_GetHeaderFromHandle has returned BioAPI_OK.
            </description>
            <equal_to var1="return" var2="__BioAPI_OK"/>
      </assert_condition>

      <!-- Issue a conformity response.
                  If the condition specified in the <description> below is false, a FAIL
                        conformity response is issued, otherwise a PASS conformity response is
                        issued.-->
      <assert_condition>
            <description>
                        The processed level of the audit data BIR is RAW.
            </description>
            <equal_to var1="processedLevel"
                        var2="__BioAPI_BIR_DATA_TYPE_RAW"/>
      </assert_condition>
</activity>
</package>
```

8.53 Assertion 15a - *BioSPI_DbOpen_ValidParam*

Description: This assertion invokes BioSPI_DbOpen with valid input parameters and verifies if the return code is BioAPI_OK.

Excerpts

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);*

Subclause 8.5.1.1

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.

References: 9.3.5.1 and 8.5.1.1

Scenario:

1) Load the BSP under test.

2) Attach the BSP under test.

3) Invoke function BioSPI_DbOpen to open the specified database.

4) Verify the return code; it is expected to be BioAPI_OK.

5) Detach and unload the BSP.

If any of the intermediate operations fails, an UNDECIDED conformity response is issued.

Expected results: The call to BioSPI_DbOpen returns BioAPI_OK.

Assertion language package

```
<package name="e68ff9a0-e506-11d9-a6a1-0002a5d5c51b">
      <author>
            ISO/IEC JTC1 SC37
      </author>

      <description>
            This package contains the assertion "BioSPI_DbOpen_ValidParam"
      </description>

      <assertion name="BioSPI_DbOpen_ValidParam" model="BSPTesting">
            <description>
                  This assertion invokes BioSPI_DbOpen with valid input parameters and verifies if
                              the return code is BioAPI_OK.
                  The relevant text in BioAPI 2.0 is quoted below from subclauses 8.5.1.1.
                  _________________________________________________________
                  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);

          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.

          _________________________________________________________
          In order to determine conformance with respect to the text above, the following
                      steps are performed:

                1)    Load the BSP under test.
                2)    Attach the BSP under test.
                3)    Invoke function BioSPI_DbOpen to open the specified database.
                4)    Verify the return code; it is expected to be BioAPI_OK.
                5)    Detach and unload the BSP.

          If any of the intermediate operations fails, an UNDECIDED conformity response is
                      issued.
    </description>

    <!-- UUID of the BSP under test -->
    <input name="_bspUuid"/>

    <!-- Timeout for the BioAPI_NOTIFY_INSERT event -->
    <input name="_inserttimeout"/>

    <!-- Indicates whether the BSP under test does not claim support for the
                      BioAPI_NOTIFY_SOURCE_PRESENT event notification -->
    <input name="_noSourcePresentSupported" />

    <!-- Timeout for the BioAPI_NOTIFY_SOURCE_PRESENT event -->
    <input name="_sourcepresenttimeout"/>

    <!-- Database UUID to be opened -->
    <input name="_dbUuid"/>

    <!-- Read Access Request to the database -->
    <input name="_readAccessRequest"/>

    <!-- Write Access Request -->
    <input name="_writeAccessRequest"/>

    <!-- Invocation of the primary activity of this assertion with input parameter values
                      assigned from the assertion's parameters. -->
    <invoke activity="BioSPI_DbOpen">
          <input name="bspUuid" var="_bspUuid"/>
          <input name="inserttimeouttime" var="_inserttimeout"/>
          <input name="nosourcepresentsupported"
                      var="_noSourcePresentSupported" />
          <input name="sourcepresenttimeouttime"
                      var="_sourcepresenttimeout"/>
          <input name="dbUuid" var="_dbUuid"/>
          <input name="readAccessRequest" var="_readAccessRequest"/>
          <input name="writeAccessRequest"
                      var="_writeAccessRequest" />
    </invoke>

    <!-- Activity bound to a function of the framework callback interface exposed by the
                      testing component.  This activity will be automatically invoked on each
                      incoming call to the function to which it is bound. -->
    <bind activity="EventHandler"
                package="02c59458-0c46-1085-95d7-0002a5d5fd2e"
                function="BioSPI_EventHandler"/>
</assertion>

<activity name="BioSPI_DbOpen">
    <input name="bspUuid"/>
    <input name="inserttimeouttime"/>
    <input name="nosourcepresentsupported" />
    <input name="sourcepresenttimeouttime"/>
    <input name="dbUuid"/>
    <input name="readAccessRequest"/>
    <input name="writeAccessRequest"/>
```

```
<!-- This assertion will use BSPHandle "1" for all BioSPI calls that require it -->
<set name="_bsphandle" value="1"/>

<!-- Invoke the functions BioSPI_BSPLoad and BioSPI_BSPAttach exposed by the BSP under
                  test.
            The input value for the parameter "unitIDOrNull" is "0", therefore the
                  assertion will test a sensor unit chosen by the BSP. -->
<invoke activity="LoadAndAttach"
            package="02c59458-0c46-1085-95d7-0002a5d5fd2e"
            break_on_break="true">
      <input name="bspUuid" var="bspUuid"/>
      <input name="bspVersion" value="32"/>
      <input name="unitIDOrNull" value="0"/>
      <input name="bspHandle" var="_bsphandle"/>
      <input name="eventtimeouttime" var="inserttimeouttime"/>
</invoke>


<invoke activity="PrepareDBTesting"
            package="02c59458-0c46-1085-95d7-0002a5d5fd2e"
            break_on_break="true">
      <input name="bspHandle" var="_bsphandle" />
      <input name="dbUuid" var="dbUuid" />
      <input name="nosourcepresentsupported" var="nosourcepresentsupported"/>
      <input name="sourcepresenttimeouttime" var="sourcepresenttimeouttime"/>
</invoke>


<!-- Invoke the function BioSPI_DbOpen to open the specified database. -->
<invoke function="BioSPI_DbOpen">
      <input name="BSPHandle" var="_bsphandle"/>
      <input name="DbUuid" var="dbUuid"/>
      <input name="ReadAccessRequest" var="readAccessRequest"/>
      <input name="WriteAccessRequest" var="writeAccessRequest"/>
      <output name="DbHandle" setvar="dbHandle"/>
      <output name="MarkerHandle" setvar="markerHandle"/>
      <return setvar="return"/>
</invoke>

<!-- Issue a conformity response.
            If the condition specified in the <description> below is false, a FAIL
                  conformity response is issued, otherwise a PASS conformity response is
                  issued.-->
<assert_condition>
      <description>
            The function BioSPI_DbOpen has returned BioAPI_OK and the output dbHandle is
                  a valid DB handle.
      </description>
      <equal_to var1="return" var2="__BioAPI_OK"/>
      <not_equal_to var1="dbHandle" var2="__BioAPI_DB_INVALID_HANDLE"/>
</assert_condition>

<!-- Invoke the function BioSPI_DbClose with valid input parameters -->
<invoke function="BioSPI_DbClose">
      <input name="BSPHandle" var="_bsphandle"/>
      <input name="DbHandle" var="dbHandle"/>
      <return setvar="return"/>
</invoke>

<!-- Issue a conformity response.
            If the condition specified in the <description> below is false, a FAIL
                  conformity response is issued, otherwise a PASS conformity response is
                  issued.-->
<assert_condition response_if_false="undecided"
            break_if_false="true">
      <description>
            The function BioSPI_DbClose has returned BioAPI_OK
      </description>
      <equal_to var1="return" var2="__BioAPI_OK"/>
</assert_condition>
```

```
            <invoke activity="CleanUpDBTesting"
                        package="02c59458-0c46-1085-95d7-0002a5d5fd2e"
                        break_on_break="true">
                  <input name="BSPHandle" var="_bsphandle" />
                  <input name="dbUuid" var="dbUuid" />
            </invoke>


            <!-- Invoke the functions BioSPI_ModuleDetach and BioSPI_ModuleUnload -->
            <invoke activity="DetachAndUnload"
                        package="02c59458-0c46-1085-95d7-0002a5d5fd2e" >
                  <input name="bspUuid" var="bspUuid" />
                  <input name="BSPHandle" var="_bsphandle" />
            </invoke>
      </activity>

</package>
```

8.54 Assertion 15b - *BioSPI_DbOpen_InvalidBSPHandle*

Description: This assertion invokes BioSPI_DbOpen with an invalid BSP handle and verifies if the return code is BioAPIERR_INVALID_BSP_HANDLE.

Excerpts

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);*

Subclause 8.5.1.1

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.

References: 9.3.5.1 and 8.5.1.1

Scenario:

1) Load the BSP under test.

2) Attach the BSP under test.

3) Invoke function BioSPI_DbOpen with an invalid BSP handle.

4) Verify the return code; it is expected to be BioAPIERR_INVALID_BSP_HANDLE.

5) Detach and unload the BSP.

If any of the intermediate operations fails, an UNDECIDED conformity response is issued.

Expected results: The call to BioSPI_DbOpen returns BioAPIERR_INVALID_BSP_HANDLE.

Assertion language package

```
<package name="bfd44400-e5de-11d9-bdb9-0002a5d5c51b">
      <author>
            ISO/IEC JTC1 SC37
      </author>

      <description>
            This package contains the assertion "BioSPI_DbOpen_InvalidBSPHandle"
      </description>

      <assertion name="BioSPI_DbOpen_InvalidBSPHandle" model="BSPTesting">
            <description>
                  This assertion invokes BioSPI_DbOpen with invalid bsp handle and verifies if the
                              return code is BioAPIERR_INVALID_BSP_HANDLE.
                  The relevant text in BioAPI 2.0 is quoted below from subclauses 8.5.1.1.
                  ___________________________________________________________
                  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);

                  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.
                  ___________________________________________________________
                  Subclause 8.5.1.2:

                  BSPHandle (input) - The handle of an attached BSP invocation.
                  ___________________________________________________________

                  In order to determine conformance with respect to the text above, the following
                              steps are performed:

                        1)    Load the BSP under test.
                        2)    Attach the BSP under test.
                        3)    Invoke function BioSPI_DbOpen with invalid bsp handle.
                        4)    Verify the return code; it is expected to be
                              BioAPIERR_INVALID_BSP_HANDLE.
                        5)    Detach and unload the BSP.

                  If any of the intermediate operations fails, an UNDECIDED conformity response is
                              issued.
            </description>

            <!-- UUID of the BSP under test -->
            <input name="_bspUuid"/>

            <!-- Timeout for the BioAPI_NOTIFY_INSERT event -->
            <input name="_inserttimeout"/>

            <!-- Indicates whether the BSP under test does not claim support for the
                              BioAPI_NOTIFY_SOURCE_PRESENT event notification -->
            <input name="_noSourcePresentSupported" />

            <!-- Timeout for the BioAPI_NOTIFY_SOURCE_PRESENT event -->
            <input name="_sourcepresenttimeout"/>

            <!-- Database Name to be opened -->
            <input name="_dbUuid"/>

            <!-- Read Access Request to the database -->
            <input name="_readAccessRequest"/>

            <!-- Write Access Request -->
            <input name="_writeAccessRequest"/>

            <!-- Invocation of the primary activity of this assertion with input parameter values
                              assigned from the assertion's parameters. -->
            <invoke activity="BioSPI_DbOpen">
                  <input name="bspUuid" var="_bspUuid"/>
                  <input name="inserttimeouttime" var="_inserttimeout"/>
                  <input name="nosourcepresentsupported"
                              var="_noSourcePresentSupported" />
            <input name="sourcepresenttimeouttime"
                        var="_sourcepresenttimeout"/>
            <input name="dbUuid" var="_dbUuid"/>
            <input name="readAccessRequest" var="_readAccessRequest"/>
            <input name="writeAccessRequest"
                        var="_writeAccessRequest" />
      </invoke>

      <!-- Activity bound to a function of the framework callback interface exposed by the
                        testing component.  This activity will be automatically invoked on each
                        incoming call to the function to which it is bound. -->
      <bind activity="EventHandler"
                  package="02c59458-0c46-1085-95d7-0002a5d5fd2e"
                  function="BioSPI_EventHandler"/>
</assertion>

<activity name="BioSPI_DbOpen">
      <input name="bspUuid"/>
      <input name="inserttimeouttime"/>
      <input name="nosourcepresentsupported" />
      <input name="sourcepresenttimeouttime"/>
      <input name="dbUuid"/>
      <input name="readAccessRequest"/>
      <input name="writeAccessRequest"/>

      <!-- This assertion will use BSPHandle "1" for all BioSPI calls that require it -->
      <set name="_bsphandle" value="1"/>

      <!-- Invoke the functions BioSPI_BSPLoad and BioSPI_BSPAttach exposed by the BSP under
                        test.
                  The input value for the parameter "unitIDOrNull" is "0", therefore the
                        assertion will test a sensor unit chosen by the BSP. -->
      <invoke activity="LoadAndAttach"
                  package="02c59458-0c46-1085-95d7-0002a5d5fd2e"
                  break_on_break="true">
            <input name="bspUuid" var="bspUuid"/>
            <input name="bspVersion" value="32"/>
            <input name="unitIDOrNull" value="0"/>
            <input name="bspHandle" var="_bsphandle"/>
            <input name="eventtimeouttime" var="inserttimeouttime"/>
      </invoke>

      <invoke activity="PrepareDBTesting"
                  package="02c59458-0c46-1085-95d7-0002a5d5fd2e"
                  break_on_break="true">
            <input name="bspHandle" var="_bsphandle" />
            <input name="dbUuid" var="dbUuid" />
            <input name="nosourcepresentsupported" var="nosourcepresentsupported"/>
            <input name="sourcepresenttimeouttime" var="sourcepresenttimeouttime"/>
      </invoke>

      <!-- Invoke the function BioSPI_DbOpen with invalid module handle. -->
      <invoke function="BioSPI_DbOpen">
            <input name="BSPHandle" value="0"/>
            <input name="DbUuid" var="dbUuid"/>
            <input name="ReadAccessRequest" var="readAccessRequest"/>
            <input name="WriteAccessRequest" var="writeAccessRequest"/>
            <output name="DbHandle" setvar="dbHandle"/>
            <output name="MarkerHandle" setvar="markerHandle"/>
            <return setvar="return"/>
      </invoke>

      <!-- Issue a conformity response.
                  If the condition specified in the <description> below is false, a FAIL
                        conformity response is issued, otherwise a PASS conformity response is
                        issued.-->
      <assert_condition>
            <description>
                  The function BioSPI_DbOpen has returned BioAPIERR_INVALID_BSP_HANDLE.
            </description>
            <equal_to var1="return" var2="__BioAPIERR_BSP_INVALID_BSP_HANDLE"/>
            <equal_to var1="dbHandle" var2="__BioAPI_DB_INVALID_HANDLE" />
      </assert_condition>
```

```
            <invoke activity="CleanUpDBTesting"
                    package="02c59458-0c46-1085-95d7-0002a5d5fd2e"
                    break_on_break="true">
                <input name="BSPHandle" var="_bsphandle" />
                <input name="dbUuid" var="dbUuid" />
            </invoke>

            <!-- Invoke the functions BioSPI_BSPDetach and BioSPI_BSPUnload -->
            <invoke activity="DetachAndUnload"
                    package="02c59458-0c46-1085-95d7-0002a5d5fd2e" >
                <input name="bspUuid" var="bspUuid" />
                <input name="BSPHandle" var="_bsphandle" />
            </invoke>
        </activity>

</package>
```

8.55 Assertion 16a - *BioSPI_DbClose_ValidParam*

Description: This assertion invokes BioSPI_DbClose with valid input parameters and verifies if the return code is BioAPI_OK.

Excerpts

Subclause 9.3.5.2

BioAPI_RETURN BioAPI BioSPI_DbClose

(BioAPI_HANDLE BSPHandle,

BioAPI_DB_HANDLE DbHandle);

Subclause 8.5.2.1

This function closes an open BIR database.

References: 9.3.5.2 and 8.5.2.1

Scenario:

1) Load the BSP under test.

2) Attach the BSP under test.

3) Create temporary database.

4) Invoke function BioSPI_DbOpen to open the temporary database.

5) Invoke function BioSPI_DbClose with valid input parameters. Verify the return code.

6) Detach and unload the BSP.

If any of the intermediate operations fails, an UNDECIDED conformity response is issued.

Expected results: The call to BioSPI_DbClose returns BioAPI_OK.

Assertion language package

```
<package name="39aa9560-e5f1-11d9-89f3-0002a5d5c51b">
      <author>
            ISO/IEC JTC1 SC37
      </author>

      <description>
            This package contains the assertion "BioSPI_DbClose_ValidParam"
      </description>

      <assertion name="BioSPI_DbClose_ValidParam" model="BSPTesting">
            <description>
                  This assertion invokes BioSPI_DbClose with valid input parameters and verifies if
                              the return code is BioAPI_OK.
                  The relevant text in BioAPI 2.0 is quoted below from subclauses 8.5.2.1.
                  _________________________________________________________
                  BioAPI_RETURN BioAPI BioAPI_DbClose
                        (BioAPI_HANDLE BSPHandle,
                        BioAPI_DB_HANDLE DbHandle);

                  This function closes an open BIR database.
                  _________________________________________________________
                  In order to determine conformance with respect to the text above, the following
                              steps are performed:

                        1)    Load the BSP under test.
                        2)    Attach the BSP under test.
                        3)  Create temporary database.
                        4)    Invoke function BioSPI_DbOpen to open the temporary database.
                        5)    Invoke function BioSPI_DbClose with valid input parameters. Verify the
                              return code.
                        6)    Detach and unload the BSP.

                  If any of the intermediate operations fails, an UNDECIDED conformity response is
                              issued.
            </description>

            <!-- UUID of the BSP under test -->
            <input name="_bspUuid"/>

            <!-- Timeout for the BioAPI_NOTIFY_INSERT event -->
            <input name="_inserttimeout"/>

            <!-- Indicates whether the BSP under test does not claim support for the
                              BioAPI_NOTIFY_SOURCE_PRESENT event notification -->
            <input name="_noSourcePresentSupported" />

            <!-- Timeout for the BioAPI_NOTIFY_SOURCE_PRESENT event -->
            <input name="_sourcepresenttimeout"/>

            <!-- Database Name to be created -->
            <input name="_dbUuid"/>

            <!-- Invocation of the primary activity of this assertion with input parameter values
                              assigned from the assertion's parameters. -->
            <invoke activity="BioSPI_DbClose">
                  <input name="bspUuid" var="_bspUuid"/>
                  <input name="inserttimeouttime" var="_inserttimeout"/>
                  <input name="nosourcepresentsupported"
                              var="_noSourcePresentSupported" />
                  <input name="sourcepresenttimeouttime"
                              var="_sourcepresenttimeout"/>
                  <input name="dbUuid" var="_dbUuid"/>
            </invoke>

            <!-- Activity bound to a function of the framework callback interface exposed by the
                              testing component.  This activity will be automatically invoked on each
                              incoming call to the function to which it is bound. -->
            <bind activity="EventHandler"
                        package="02c59458-0c46-1085-95d7-0002a5d5fd2e"
                        function="BioSPI_EventHandler"/>
      </assertion>
```

```
<activity name="BioSPI_DbClose">
      <input name="bspUuid"/>
      <input name="inserttimeouttime"/>
      <input name="nosourcepresentsupported" />
      <input name="sourcepresenttimeouttime"/>
      <input name="dbUuid"/>

      <!-- This assertion will use BSPHandle "1" for all BioSPI calls that require it -->
      <set name="_bsphandle" value="1"/>

      <!-- Invoke the functions BioSPI_BSPLoad and BioSPI_BSPAttach exposed by the BSP under
                  test.
            The input value for the parameter "unitIDOrNull" is "0", therefore the
                  assertion will test a sensor unit chosen by the BSP. -->
      <invoke activity="LoadAndAttach"
            package="02c59458-0c46-1085-95d7-0002a5d5fd2e"
            break_on_break="true">
            <input name="bspUuid" var="bspUuid"/>
            <input name="bspVersion" value="32"/>
            <input name="unitIDOrNull" value="0"/>
            <input name="bspHandle" var="_bsphandle"/>
            <input name="eventtimeouttime" var="inserttimeouttime"/>
      </invoke>

      <invoke activity="PrepareDBTesting"
            package="02c59458-0c46-1085-95d7-0002a5d5fd2e"
            break_on_break="true">
            <input name="bspHandle" var="_bsphandle" />
            <input name="dbUuid" var="dbUuid" />
            <input name="nosourcepresentsupported" var="nosourcepresentsupported"/>
            <input name="sourcepresenttimeouttime" var="sourcepresenttimeouttime"/>
      </invoke>

      <!-- Invoke the function BioSPI_DbOpen to open the specified database. -->
      <invoke function="BioSPI_DbOpen">
            <input name="BSPHandle" var="_bsphandle"/>
            <input name="DbUuid" var="dbUuid"/>
            <input name="ReadAccessRequest" value="true"/>
            <input name="WriteAccessRequest" value="true"/>
            <output name="DbHandle" setvar="dbHandle"/>
            <output name="MarkerHandle" setvar="markerHandle"/>
            <return setvar="return"/>
      </invoke>

      <!-- Issue a conformity response.
            If the condition specified in the <description> below is false, a UNDECIDED
                  conformity response is issued, otherwise a PASS conformity response is
                  issued.-->
      <assert_condition response_if_false="undecided"
            break_if_false="true">
            <description>
                  The function BioSPI_DbOpen has returned BioAPI_OK and the output dbHandle is
                  a valid DB handle.
            </description>
            <equal_to var1="return" var2="__BioAPI_OK"/>
            <not_equal_to var1="dbHandle" var2="__BioAPI_DB_INVALID_HANDLE"/>
      </assert_condition>

      <!-- Invoke the function BioSPI_DbClose with valid input parameters -->
      <invoke function="BioSPI_DbClose">
            <input name="BSPHandle" var="_bsphandle"/>
            <input name="DbHandle" var="dbHandle"/>
            <return setvar="return"/>
      </invoke>

      <!-- Issue a conformity response.
            If the condition specified in the <description> below is false, an FAIL
                  conformity response is issued, otherwise a PASS conformity response is
                  issued.-->
      <assert_condition>
            <description>
                  The function BioSPI_DbClose has returned BioAPI_OK
            </description>
            <equal_to var1="return" var2="__BioAPI_OK"/>
      </assert_condition>
```

```
            <invoke activity="CleanUpDBTesting"
                        package="02c59458-0c46-1085-95d7-0002a5d5fd2e"
                        break_on_break="true">
                  <input name="BSPHandle" var="_bsphandle" />
                  <input name="dbUuid" var="dbUuid" />
            </invoke>

            <!-- Invoke the functions BioSPI_BSPDetach and BioSPI_BSPUnload -->
            <invoke activity="DetachAndUnload"
                        package="02c59458-0c46-1085-95d7-0002a5d5fd2e" >
                  <input name="bspUuid" var="bspUuid" />
                  <input name="BSPHandle" var="_bsphandle" />
            </invoke>
      </activity>

</package>
```

8.56 Assertion 16b - *BioSPI_DbClose_InvalidBSPHandle*

Description: This assertion invokes BioSPI_DbClose with an invalid BSP handle and verifies if the return code is BioAPIERR_INVALID_BSP_HANDLE.

Excerpts

Subclause 9.3.5.2

BioAPI_RETURN BioAPI BioSPI_DbClose

(BioAPI_HANDLE BSPHandle,

BioAPI_DB_HANDLE DbHandle);

Subclause 8.5.2.1

This function closes an open BIR database.

References: 9.3.5.2 and 8.5.2.1

Scenario:

1) Load the BSP under test.

2) Attach the BSP under test.

3) Create temporary database.

4) Invoke function BioSPI_DbOpen to open the temporary database.

5) Invoke function BioSPI_DbClose with an invalid BSP handle. Verify the return code.

6) Detach and unload the BSP.

If any of the intermediate operations fails, an UNDECIDED conformity response is issued.

Expected results: The call to BioSPI_DbClose returns BioAPIERR_INVALID_BSP_HANDLE.

Assertion language package

```
<package name="6e3f5c00-e5f3-11d9-b663-0002a5d5c51b">
      <author>
            ISO/IEC JTC1 SC37
      </author>

      <description>
            This package contains the assertion "BioSPI_DbClose_InvalidBSPHandle"
      </description>

      <assertion name="BioSPI_DbClose_InvalidBSPHandle" model="BSPTesting">
            <description>
                  This assertion invokes BioSPI_DbClose with invalid BSP handle and verifies if the
                              return code is BioAPIERR_INVALID_BSP_HANDLE.
                  The relevant text in BioAPI 2.0 is quoted below from
                  subclauses 8.5.2.1.
                  _________________________________________________________
                  BioAPI_RETURN BioAPI BioAPI_DbClose
                        (BioAPI_HANDLE BSPHandle,
                        BioAPI_DB_HANDLE DbHandle);

                  This function closes an open BIR database.
                  _________________________________________________________
                  8.5.2.2 Parameters
                        BSPHandle (input) - The handle of the attached BioAPI service provider.
                  _________________________________________________________


                  In order to determine conformance with respect to the text above, the following
                              steps are performed:

                        1)    Load the BSP under test.
                        2)    Attach the BSP under test.
                        3)  Create temporary database.
                        3)    Invoke function BioSPI_DbOpen to open the temporary database.
                        4)    Invoke function BioSPI_DbClose with invalid bsp handle. Verify the
                              return code.
                        5)    Detach and unload the BSP.

                  If any of the intermediate operations fails, an UNDECIDED conformity response is
                              issued.
            </description>

            <!-- UUID of the BSP under test -->
            <input name="_bspUuid"/>

            <!-- Timeout for the BioAPI_NOTIFY_INSERT event -->
            <input name="_inserttimeout"/>

            <!-- Indicates whether the BSP under test does not claim support for the
                              BioAPI_NOTIFY_SOURCE_PRESENT event notification -->
            <input name="_noSourcePresentSupported" />

            <!-- Timeout for the BioAPI_NOTIFY_SOURCE_PRESENT event -->
            <input name="_sourcepresenttimeout"/>

            <!-- Database Name to be created -->
            <input name="_dbUuid"/>

            <!-- Invocation of the primary activity of this assertion with input parameter values
                              assigned from the assertion's parameters. -->
            <invoke activity="BioSPI_DbClose">
                  <input name="bspUuid" var="_bspUuid"/>
                  <input name="inserttimeouttime" var="_inserttimeout"/>
                  <input name="nosourcepresentsupported"
                              var="_noSourcePresentSupported" />
                  <input name="sourcepresenttimeouttime"
                              var="_sourcepresenttimeout"/>
                  <input name="dbUuid" var="_dbUuid"/>
            </invoke>
```

```
        <!-- Activity bound to a function of the framework callback interface exposed by the
                     testing component.  This activity will be automatically invoked on each
                     incoming call to the function to which it is bound. -->
        <bind activity="EventHandler"
                  package="02c59458-0c46-1085-95d7-0002a5d5fd2e"
                  function="BioSPI_EventHandler"/>
  </assertion>

  <activity name="BioSPI_DbClose">
        <input name="bspUuid"/>
        <input name="inserttimeouttime"/>
        <input name="nosourcepresentsupported" />
        <input name="sourcepresenttimeouttime"/>
        <input name="dbUuid"/>

        <!-- This assertion will use BSPHandle "1" for all BioSPI calls that require it -->
        <set name="_bsphandle" value="1"/>

        <!-- Invoke the functions BioSPI_BSPLoad and BioSPI_BSPAttach exposed by the BSP under
                     test.
                  The input value for the parameter "unitIDOrNull" is "0", therefore the
                     assertion will test a sensor unit chosen by the BSP. -->
        <invoke activity="LoadAndAttach"
                  package="02c59458-0c46-1085-95d7-0002a5d5fd2e"
                  break_on_break="true">
              <input name="bspUuid" var="bspUuid"/>
              <input name="bspVersion" value="32"/>
              <input name="unitIDOrNull" value="0"/>
              <input name="bspHandle" var="_bsphandle"/>
              <input name="eventtimeouttime" var="inserttimeouttime"/>
        </invoke>

        <invoke activity="PrepareDBTesting"
                  package="02c59458-0c46-1085-95d7-0002a5d5fd2e"
                  break_on_break="true">
              <input name="bspHandle" var="_bsphandle" />
              <input name="dbUuid" var="dbUuid" />
              <input name="nosourcepresentsupported" var="nosourcepresentsupported"/>
              <input name="sourcepresenttimeouttime" var="sourcepresenttimeouttime"/>
        </invoke>

        <!-- Invoke the function BioSPI_DbOpen to open the specified database. -->
        <invoke function="BioSPI_DbOpen">
              <input name="BSPHandle" var="_bsphandle"/>
              <input name="DbUuid" var="dbUuid"/>
              <input name="ReadAccessRequest" value="true"/>
              <input name="WriteAccessRequest" value="true"/>
              <output name="DbHandle" setvar="dbHandle"/>
              <output name="MarkerHandle" setvar="markerHandle"/>
              <return setvar="return"/>
        </invoke>

        <!-- Issue a conformity response.
                  If the condition specified in the <description> below is false, a UNDECIDED
                     conformity response is issued, otherwise a PASS conformity response is
                     issued.-->
        <assert_condition response_if_false="undecided"
                  break_if_false="true">
              <description>
                  The function BioSPI_DbOpen has returned BioAPI_OK and the output dbHandle is
                     a valid DB handle.
              </description>
              <equal_to var1="return" var2="__BioAPI_OK"/>
              <not_equal_to var1="dbHandle" var2="__BioAPI_DB_INVALID_HANDLE"/>
        </assert_condition>

        <!-- Invoke the function BioSPI_DbClose with invalid BSP handle -->
        <invoke function="BioSPI_DbClose">
              <input name="BSPHandle" value="0"/>
              <input name="DbHandle" var="dbHandle"/>
              <return setvar="return"/>
        </invoke>

        <!-- Issue a conformity response.
```

```
                    If the condition specified in the <description> below is false, a FAIL
                          conformity response is issued, otherwise a PASS conformity response is
                          issued.-->
            <assert_condition>
                  <description>
                        The function BioSPI_DbClose has returned
                              BioAPIERR_H_FRAMEWORK_INVALID_BSP_HANDLE.
                  </description>
                  <equal_to var1="return" var2="__BioAPIERR_BSP_INVALID_BSP_HANDLE"/>
            </assert_condition>

            <!-- Invoke the function BioSPI_DbClose with valid input parameters -->
            <invoke function="BioSPI_DbClose">
                  <input name="BSPHandle" var="_bsphandle"/>
                  <input name="DbHandle" var="dbHandle"/>
                  <return setvar="return"/>
            </invoke>

            <!-- Issue a conformity response.
                        If the condition specified in the <description> below is false, a FAIL
                              conformity response is issued, otherwise a PASS conformity response is
                              issued.-->
            <assert_condition response_if_false="undecided"
                        break_if_false="true">
                  <description>
                        The function BioSPI_DbClose has returned BioAPI_OK
                  </description>
                  <equal_to var1="return" var2="__BioAPI_OK"/>
            </assert_condition>

            <invoke activity="CleanUpDBTesting"
                        package="02c59458-0c46-1085-95d7-0002a5d5fd2e"
                        break_on_break="true">
                  <input name="BSPHandle" var="_bsphandle" />
                  <input name="dbUuid" var="dbUuid" />
            </invoke>

            <!-- Invoke the functions BioSPI_BSPDetach and BioSPI_BSPUnload -->
            <invoke activity="DetachAndUnload"
                        package="02c59458-0c46-1085-95d7-0002a5d5fd2e" >
                  <input name="bspUuid" var="bspUuid" />
                  <input name="BSPHandle" var="_bsphandle" />
            </invoke>
      </activity>

</package>
```

8.57 Assertion 17a - *BioSPI_DbCreate_DbProtected*

Description: This assertion invokes BioSPI_DbCreate twice with valid input parameters and verifies if the second invocation results in returning error code BioAPIERR_DATABASE_ALREADY_EXISTS.

Excerpts

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);*

Subclause 8.5.3.1

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.

References: 9.3.5.3 and 8.5.3.1

Scenario:

1) Load the BSP under test.

2) Attach the BSP under test.

3) Invoke function BioSPI_DbCreate to create the specified database.

4) Invoke function BioSPI_DbCreate again.

5) Verify the return code; it is expected to be BioAPIERR_DATABASE_ALREADY_EXISTS.

6) Detach and unload the BSP.

If any of the intermediate operations fails, an UNDECIDED conformity response is issued.

Expected results: The second call to BioSPI_DbCreate returns BioAPIERR_DATABASE_ALREADY_EXISTS.

Assertion language package

```
<package name="7b6c2f40-e650-11d9-812f-0002a5d5c51b">
      <author>
            ISO/IEC JTC1 SC37
      </author>

      <description>
            This package contains the assertion "BioSPI_DbCreate_DbProtected"
      </description>

      <assertion name="BioSPI_DbCreate_DbProtected" model="BSPTesting">
            <description>
                  This assertion invokes BioSPI_DbCreate twice with valid input parameters and
                              verifies if the second invocation results returning error code
                              BioAPIERR_DATABASE_ALREADY_EXISTS.
                  The relevant text in BioAPI 2.0 is quoted below from subclauses 8.5.3.1
                  __________________________________________________________
                  BioAPI_RETURN BioAPI BioSPI_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.
```

```
        Errors
              BioAPIERR_DATABASE_ALREADY_EXISTS

        DbHandle (output) - The handle to the newly created and open data store. The value
                    will be set to BioAPI_DB_INVALID_HANDLE if the                function
                    fails.

        ________________________________________________________
        In order to determine conformance with respect to the text above, the following
                    steps are performed:

              1)    Load the BSP under test.
              2)    Attach the BSP under test.
              3)    Invoke function BioSPI_DbCreate to create the specified database.
              4)    Invoke function BioSPI_DbCreate again.
              4)    Verify the return code; it is expected to be
                    BioAPIERR_DATABASE_ALREADY_EXISTS.
              5)    Detach and unload the BSP.

        If any of the intermediate operations fails, an UNDECIDED conformity response is
                    issued.
  </description>

  <!-- UUID of the BSP under test -->
  <input name="_bspUuid"/>

  <!-- Timeout for the BioAPI_NOTIFY_INSERT event -->
  <input name="_inserttimeout"/>

  <!-- Database Uuid to be opened -->
  <input name="_dbUuid"/>

  <!-- Read Access Request to the database -->
  <input name="_readAccessRequest"/>

  <!-- Write Access Request -->
  <input name="_writeAccessRequest"/>

  <!-- Invocation of the primary activity of this assertion with input parameter values
              assigned from the assertion's parameters. -->
  <invoke activity="BioSPI_DbCreate">
        <input name="bspUuid" var="_bspUuid"/>
        <input name="inserttimeouttime" var="_inserttimeout"/>
        <input name="dbUuid" var="_dbUuid"/>
        <input name="readAccessRequest" var="_readAccessRequest"/>
        <input name="writeAccessRequest"
              var="_writeAccessRequest" />
  </invoke>

  <!-- Activity bound to a function of the framework callback interface exposed by the
              testing component.  This activity will be automatically invoked on each
              incoming call to the function to which it is bound. -->
  <bind activity="EventHandler"
        package="02c59458-0c46-1085-95d7-0002a5d5fd2e"
        function="BioSPI_EventHandler"/>
</assertion>

<activity name="BioSPI_DbCreate">
  <input name="bspUuid"/>
  <input name="inserttimeouttime"/>
  <input name="dbUuid"/>
  <input name="readAccessRequest"/>
  <input name="writeAccessRequest"/>

  <!-- This assertion will use BSPHandle "1" for all BioSPI calls that require it -->
  <set name="_bsphandle" value="1"/>

  <!-- Invoke the functions BioSPI_BSPLoad and BioSPI_BSPAttach exposed by the BSP under
              test.
        The input value for the parameter "unitIDOrNull" is "0", therefore the
              assertion will test a sensor unit chosen by the BSP. -->
  <invoke activity="LoadAndAttach"
        package="02c59458-0c46-1085-95d7-0002a5d5fd2e"
        break_on_break="true">
     <input name="bspUuid" var="bspUuid"/>
     <input name="bspVersion" value="32"/>
```

```
      <input name="unitIDOrNull" value="0"/>
      <input name="bspHandle" var="_bsphandle"/>
      <input name="eventtimeouttime" var="inserttimeouttime"/>
</invoke>

<!-- Remove the database if it already exists -->
<invoke function="BioSPI_DbDelete" >
      <input name="BSPHandle" var="_bsphandle" />
      <input name="DbUuid" var="dbUuid" />
      <return setvar="return"/>
</invoke>

<!-- Invoke the function BioSPI_DbCreate to create the specified database. -->
<invoke function="BioSPI_DbCreate">
      <input name="BSPHandle" var="_bsphandle"/>
      <input name="DbUuid" var="dbUuid"/>
      <input name="NumberOfRecords" value="1"/>
      <input name="ReadAccessRequest" var="readAccessRequest"/>
      <input name="WriteAccessRequest" var="writeAccessRequest"/>
      <output name="DbHandle" setvar="dbHandle"/>
      <return setvar="return"/>
</invoke>

<!-- Issue a conformity response.
            If the condition specified in the <description> below is false, an UNDECIDED
                  conformity response is issued, otherwise a PASS conformity response is
                  issued.-->
<assert_condition response_if_false="undecided"
            break_if_false="true">
      <description>
            The function BioSPI_DbCreate has returned BioAPI_OK
      </description>
      <equal_to var1="return" var2="__BioAPI_OK"/>
</assert_condition>

<!-- Invoke the function BioSPI_DbClose with valid input parameters -->
<invoke function="BioSPI_DbClose">
      <input name="BSPHandle" var="_bsphandle"/>
      <input name="DbHandle" var="dbHandle"/>
      <return setvar="return"/>
</invoke>

<!-- Issue a conformity response.
            If the condition specified in the <description> below is false, a FAIL
                  conformity response is issued, otherwise a PASS conformity response is
                  issued.-->
<assert_condition response_if_false="undecided"
            break_if_false="true">
      <description>
            The function BioSPI_DbClose has returned BioAPI_OK
      </description>
      <equal_to var1="return" var2="__BioAPI_OK"/>
</assert_condition>

<!-- Invoke the function BioSPI_DbCreate to create the specified database again. -->
<invoke function="BioSPI_DbCreate">
      <input name="BSPHandle" var="_bsphandle"/>
      <input name="DbUuid" var="dbUuid"/>
      <input name="NumberOfRecords" value="1"/>
      <input name="ReadAccessRequest" var="readAccessRequest"/>
      <input name="WriteAccessRequest" var="writeAccessRequest"/>
      <output name="DbHandle" setvar="dbHandle"/>
      <return setvar="return"/>
</invoke>
```

```
            <!-- Issue a conformity response.
                    If the condition specified in the <description> below is false, a FAIL
                        conformity response is issued, otherwise a PASS conformity response is
                        issued.-->
            <assert_condition>
                <description>
                    The function BioSPI_DbCreate has returned BioAPIERR_DATABASE_ALREADY_EXISTS,
                        and the output DbHandle is set to BioAPI_DB_INVALID_HANDLE.
                </description>
                <equal_to var1="return" var2="__BioAPIERR_BSP_DATABASE_ALREADY_EXISTS"/>
                <equal_to var1="dbHandle" var2="__BioAPI_DB_INVALID_HANDLE"/>
            </assert_condition>

            <!-- Invoke the function BioSPI_DbDelete with valid input parameters -->
            <invoke function="BioSPI_DbDelete">
                <input name="BSPHandle" var="_bsphandle"/>
                <input name="DbUuid" var="dbUuid"/>
                <return setvar="return"/>
            </invoke>

            <!-- Issue a conformity response.
                    If the condition specified in the <description> below is false, an UNDECIDED
                        conformity response is issued, otherwise a PASS conformity response is
                        issued.-->
            <assert_condition response_if_false="undecided"
                    break_if_false="true">
                <description>
                    The function BioSPI_DbDelete has returned BioAPI_OK
                </description>
                <equal_to var1="return" var2="__BioAPI_OK"/>
            </assert_condition>


            <!-- Invoke the functions BioSPI_BSPDetach and BioSPI_BSPUnload -->
            <invoke activity="DetachAndUnload"
                    package="02c59458-0c46-1085-95d7-0002a5d5fd2e" >
                <input name="bspUuid" var="bspUuid" />
                <input name="BSPHandle" var="_bsphandle" />
            </invoke>
        </activity>
    </package>
```

8.58 Assertion 17b - *BioSPI_DbCreate_ValidParam*

Description: This assertion invokes BioSPI_DbCreate with valid input parameters and verifies if the return code is BioAPI_OK.

Excerpts

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);*

Subclause 8.5.3.1

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.

References: 9.3.5.3 and 8.5.3.1

Scenario:

1) Load the BSP under test.

2) Attach the BSP under test.

3) Invoke function BioSPI_DbCreate to create the specified database.

4) Verify the return code; it is expected to be BioAPI_OK.

5) Detach and unload the BSP.

If any of the intermediate operations fails, an UNDECIDED conformity response is issued.

Expected results: The call to BioSPI_DbCreate returns BioAPI_OK.

Assertion language package

```
<package name="1421ec38-1db6-49d4-873d-03e2de17598b">
      <author>
            ISO/IEC JTC1 SC37
      </author>

      <description>
            This package contains the assertion "BioSPI_DbCreate_ValidParam"
      </description>

      <assertion name="BioSPI_DbCreate_ValidParam" model="BSPTesting">
            <description>
                  This assertion invokes BioSPI_DbCreate with valid input parameters and verifies if
                              the return code is BioAPI_OK.
                  The relevant text in BioAPI 2.0 is quoted below from subclauses 8.5.3.1.
                  _________________________________________________________
                  BioAPI_RETURN BioAPI BioSPI_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.

                  _________________________________________________________
                  In order to determine conformance with respect to the text above, the following
                              steps are performed:

                        1)    Load the BSP under test.
                        2)    Attach the BSP under test.
                        3)    Invoke function BioSPI_DbCreate to create the specified database.
                        4)    Verify the return code; it is expected to be BioAPI_OK.
                        5)    Detach and unload the BSP.

                  If any of the intermediate operations fails, an UNDECIDED conformity response is
                              issued.
            </description>

            <!-- UUID of the BSP under test -->
            <input name="_bspUuid"/>

            <!-- Timeout for the BioAPI_NOTIFY_INSERT event -->
            <input name="_inserttimeout"/>
```

```
        <!-- Database Name to be opened -->
        <input name="_dbUuid"/>

        <!-- Number of Records -->
        <input name="_nbrRecords"/>

        <!-- Read Access Request to the database -->
        <input name="_readAccessRequest"/>

        <!-- Write Access Request -->
        <input name="_writeAccessRequest"/>

        <!-- Invocation of the primary activity of this assertion with input parameter values
                      assigned from the assertion's parameters. -->
        <invoke activity="BioSPI_DbCreate">
              <input name="bspUuid" var="_bspUuid"/>
              <input name="inserttimeouttime" var="_inserttimeout"/>
              <input name="dbUuid" var="_dbUuid"/>
              <input name="nbrRecords" var="_nbrRecords"/>
              <input name="readAccessRequest" var="_readAccessRequest"/>
              <input name="writeAccessRequest"
                      var="_writeAccessRequest" />
        </invoke>

        <!-- Activity bound to a function of the framework callback interface exposed by the
                      testing component.  This activity will be automatically invoked on each
                      incoming call to the function to which it is bound. -->
        <bind activity="EventHandler"
                    package="02c59458-0c46-1085-95d7-0002a5d5fd2e"
                    function="BioSPI_EventHandler"/>
  </assertion>

  <activity name="BioSPI_DbCreate">
        <input name="bspUuid"/>
        <input name="inserttimeouttime"/>
        <input name="dbSupported"/>
        <input name="dbUuid"/>
        <input name="nbrRecords"/>
        <input name="readAccessRequest"/>
        <input name="writeAccessRequest"/>

        <!-- This assertion will use BSPHandle "1" for all BioSPI calls that require it -->
        <set name="_bsphandle" value="1"/>

        <!-- Invoke the functions BioSPI_BSPLoad and BioSPI_BSPAttach exposed by the BSP under
                      test.
                The input value for the parameter "unitIDOrNull" is "0", therefore the
                      assertion will test a sensor unit chosen by the BSP. -->
        <invoke activity="LoadAndAttach"
                    package="02c59458-0c46-1085-95d7-0002a5d5fd2e"
                    break_on_break="true">
              <input name="bspUuid" var="bspUuid"/>
              <input name="bspVersion" value="32"/>
              <input name="unitIDOrNull" value="0"/>
              <input name="bspHandle" var="_bsphandle"/>
              <input name="eventtimeouttime" var="inserttimeouttime"/>
        </invoke>

        <!-- Remove the database if it already exists -->
        <invoke function="BioSPI_DbDelete" >
              <input name="BSPHandle" var="_bsphandle" />
              <input name="DbUuid" var="dbUuid" />
              <return setvar="return"/>
        </invoke>
```

```
<!-- Issue a conformity response.
            If the condition specified in the <description> below is false, an UNDECIDED
                  conformity response is issued, otherwise a PASS conformity response is
                  issued.-->
<assert_condition response_if_false="undecided"
            break_if_false="true">
      <description>
            The function BioSPI_DbCreate has returned BioAPI_OK
      </description>
      <or>
            <equal_to var1="return" var2="__BioAPI_OK"/>
            <equal_to var1="return" var2="__BioAPIERR_BSP_DATABASE_DOES_NOT_EXIST"/>
      </or>
</assert_condition>

<!-- Invoke the function BioSPI_DbCreate to create the specified database. -->
<invoke function="BioSPI_DbCreate">
      <input name="BSPHandle" var="_bsphandle"/>
      <input name="DbUuid" var="dbUuid"/>
      <input name="NumberOfRecords" var="nbrRecords"/>
      <input name="ReadAccessRequest" var="readAccessRequest"/>
      <input name="WriteAccessRequest" var="writeAccessRequest"/>
      <output name="DbHandle" setvar="dbHandle"/>
      <return setvar="return"/>
</invoke>

<!-- Issue a conformity response.
            If the condition specified in the <description> below is false, a FAIL
                  conformity response is issued, otherwise a PASS conformity response is
                  issued.-->
<assert_condition>
      <description>
            The function BioSPI_DbCreate has returned BioAPI_OK
      </description>
      <equal_to var1="return" var2="__BioAPI_OK"/>
</assert_condition>

<!-- Invoke the function BioSPI_DbClose with valid input parameters -->
<invoke function="BioSPI_DbClose">
      <input name="BSPHandle" var="_bsphandle"/>
      <input name="DbHandle" var="dbHandle"/>
      <return setvar="return"/>
</invoke>

<!-- Issue a conformity response.
            If the condition specified in the <description> below is false, an UNDECIDED
                  conformity response is issued, otherwise a PASS conformity response is
                  issued.-->
<assert_condition response_if_false="undecided"
            break_if_false="true">
      <description>
            The function BioSPI_DbClose has returned BioAPI_OK
      </description>
      <equal_to var1="return" var2="__BioAPI_OK"/>
</assert_condition>

<!-- Invoke the function BioSPI_DbDelete with valid input parameters -->
<invoke function="BioSPI_DbDelete">
      <input name="BSPHandle" var="_bsphandle"/>
      <input name="DbUuid" var="dbUuid"/>
      <return setvar="return"/>
</invoke>

<!-- Issue a conformity response.
            If the condition specified in the <description> below is false, an UNDECIDED
                  conformity response is issued, otherwise a PASS conformity response is
                  issued.-->
<assert_condition response_if_false="undecided"
            break_if_false="true">
      <description>
            The function BioSPI_DbDelete has returned BioAPI_OK
      </description>
      <equal_to var1="return" var2="__BioAPI_OK"/>
</assert_condition>
```

```
            <!-- Invoke the functions BioSPI_ModuleDetach and BioSPI_ModuleUnload -->
            <invoke activity="DetachAndUnload"
                        package="02c59458-0c46-1085-95d7-0002a5d5fd2e" >
                  <input name="bspUuid" var="bspUuid" />
                  <input name="BSPHandle" var="_bsphandle" />
            </invoke>
      </activity>

</package>
```

8.59 Assertion 17c - *BioSPI_DbCreate_InvalidBSPHandle*

Description: This assertion invokes BioSPI_DbCreate with invalid BSP handle and verifies if the return code is BioAPIERR_INVALID_BSP_HANDLE.

Excerpts

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);*

Subclause 8.5.3.1

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.

References: 9.3.5.3 and 8.5.3.1

Scenario:

1) Load the BSP under test.

2) Attach the BSP under test.

3) Invoke function BioSPI_DbCreate to create the specified database with an invalid BSP handle.

4) Verify the return code; it is expected to be BioAPIERR_INVALID_BSP_HANDLE.

5) Detach and unload the BSP.

If any of the intermediate operations fails, an UNDECIDED conformity response is issued.

Expected results: The call to BioSPI_DbCreate returns BioAPIERR_INVALID_BSP_HANDLE.

Assertion language package

```
<package name="ef4bb862-79f6-4f01-8f5d-af5c3abf23c0">
      <author>
            ISO/IEC JTC1 SC37
      </author>

      <description>
            This package contains the assertion "BioSPI_DbCreate_InvalidBSPHandle"
      </description>

      <assertion name="BioSPI_DbCreate_InvalidBSPHandle" model="BSPTesting">
            <description>
                  This assertion invokes BioSPI_DbCreate with invalid BSP handle and verifies if the
                              return code is BioAPIERR_INVALID_BSP_HANDLE.
                  The relevant text in BioAPI 2.0 is quoted below from subclauses 8.5.3.1.
                  _________________________________________________________
                  BioAPI_RETURN BioAPI BioSPI_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.

                  _________________________________________________________
                  8.5.3.2 Parameters
                        BSPHandle (input) - The handle of the attached BioAPI service provider.
                  _________________________________________________________

                  In order to determine conformance with respect to the text above, the following
                              steps are performed:

                        1)    Load the BSP under test.
                        2)    Attach the BSP under test.
                        3)    Invoke function BioSPI_DbCreate to create the specified database with an
                              invalid module handle.
                        4)    Verify the return code; it is expected to be
                              BioAPIERR_INVALID_BSP_HANDLE.
                        5)    Detach and unload the BSP.

                  If any of the intermediate operations fails, an UNDECIDED conformity response is
                              issued.
            </description>

            <!-- UUID of the BSP under test -->
            <input name="_bspUuid"/>

            <!-- Timeout for the BioAPI_NOTIFY_INSERT event -->
            <input name="_inserttimeout"/>

            <!-- Database Uuid to be opened -->
            <input name="_dbUuid"/>

            <!-- Read Access Request to the database -->
            <input name="_readAccessRequest"/>

            <!-- Write Access Request -->
            <input name="_writeAccessRequest"/>
      <!-- Invocation of the primary activity of this assertion with input parameter values
                  assigned from the assertion's parameters. -->
      <invoke activity="BioSPI_DbCreate">
            <input name="bspUuid" var="_bspUuid"/>
            <input name="inserttimeouttime" var="_inserttimeout"/>
            <input name="dbUuid" var="_dbUuid"/>
            <input name="readAccessRequest" var="_readAccessRequest"/>
            <input name="writeAccessRequest"
                  var="_writeAccessRequest" />
      </invoke>

      <!-- Activity bound to a function of the framework callback interface exposed by the
                  testing component.  This activity will be automatically invoked on each
                  incoming call to the function to which it is bound. -->
      <bind activity="EventHandler"
            package="02c59458-0c46-1085-95d7-0002a5d5fd2e"
            function="BioSPI_EventHandler"/>
</assertion>

<activity name="BioSPI_DbCreate">
      <input name="bspUuid"/>
      <input name="inserttimeouttime"/>
      <input name="nosourcepresentsupported" />
      <input name="sourcepresenttimeouttime"/>
      <input name="dbUuid"/>
      <input name="readAccessRequest"/>
      <input name="writeAccessRequest"/>

      <!-- This assertion will use BSPHandle "1" for all BioSPI calls that require it -->
      <set name="_bsphandle" value="1"/>

      <!-- Invoke the functions BioSPI_BSPLoad and BioSPI_BSPAttach exposed by the BSP under
                  test.
            The input value for the parameter "unitIDOrNull" is "0", therefore the
                  assertion will test a sensor unit chosen by the BSP. -->
      <invoke activity="LoadAndAttach"
            package="02c59458-0c46-1085-95d7-0002a5d5fd2e"
            break_on_break="true">
            <input name="bspUuid" var="bspUuid"/>
            <input name="bspVersion" value="32"/>
            <input name="unitIDOrNull" value="0"/>
            <input name="bspHandle" var="_bsphandle"/>
            <input name="eventtimeouttime" var="inserttimeouttime"/>
      </invoke>

      <!-- Remove the database if it already exists -->
      <invoke function="BioSPI_DbDelete" >
            <input name="BSPHandle" var="_bsphandle" />
            <input name="DbUuid" var="dbUuid" />
            <return setvar="return"/>
      </invoke>

      <!-- Invoke the function BioSPI_DbCreate to create the specified database. -->
      <invoke function="BioSPI_DbCreate">
            <input name="BSPHandle" value="0"/>
            <input name="DbUuid" var="dbUuid"/>
            <input name="NumberOfRecords" value="1"/>
            <input name="ReadAccessRequest" var="readAccessRequest"/>
            <input name="WriteAccessRequest" var="writeAccessRequest"/>
            <output name="DbHandle" setvar="dbHandle"/>
            <return setvar="return"/>
      </invoke>

      <!-- Issue a conformity response.
            If the condition specified in the <description> below is false, a FAIL
                  conformity response is issued, otherwise a PASS conformity response is
                  issued.-->
      <assert_condition>
            <description>
                  The function BioSPI_DbCreate has returned BioAPIERR_INVALID_BSP_HANDLE and
                        the output DbHandle is set to BioAPI_DB_INVALID_HANDLE.
            </description>
            <equal_to var1="return" var2="__BioAPIERR_BSP_INVALID_BSP_HANDLE "/>
            <equal_to var1="dbHandle" var2="__BioAPI_DB_INVALID_HANDLE" />
      </assert_condition>
```

```
            <!-- Invoke the functions BioSPI_BSPDetach and BioSPI_BSPUnload -->
            <invoke activity="DetachAndUnload"
                        package="02c59458-0c46-1085-95d7-0002a5d5fd2e" >
                  <input name="bspUuid" var="bspUuid" />
                  <input name="BSPHandle" var="_bsphandle" />
            </invoke>
      </activity>

</package>
```

8.60 Assertion 18a - *BioSPI_DbDelete_InvalidBSPHandle*

Description: This assertion invokes BioSPI_DbDelete with an invalid BSP handle and verifies if the return code is BioAPIERR_INVALID_BSP_HANDLE.

Excerpts

Subclause 9.3.5.4

BioAPI_RETURN BioAPI BioSPI_DbDelete

(BioAPI_HANDLE BSPHandle,
*const BioAPI_UUID *DbUuid);*

Subclause 8.5.4.1

This function deletes all records from the specified BIR database and removes all state information associated with that database.

References: 9.3.5.4 and 8.5.4.1

Scenario:

1) Load the BSP under test.

2) Attach the BSP under test.

3) Invoke function BioSPI_DbCreate to create the specified database.

4) Invoke function BioSPI_DbClose to close the created database.

5) Invoke function BioSPI_DbDelete with an invalid module handle.

6) Verify the return code; it is expected to be BioAPIERR_INVALID_BSP_HANDLE.

7) Detach and unload the BSP.

If any of the intermediate operations fails, an UNDECIDED conformity response is issued.

Expected results: The call to BioSPI_DbDelete returns BioAPIERR_INVALID_BSP_HANDLE.

Assertion language package

```
<package name="678e5d12-3d51-41ec-a672-13f34ea24545">
      <author>
            ISO/IEC JTC1 SC37
      </author>

      <description>
            This package contains the assertion "BioSPI_DbDelete_InvalidBSPHandle"
      </description>

      <assertion name="BioSPI_DbDelete_InvalidBSPHandle" model="BSPTesting">
            <description>
                  This assertion invokes BioSPI_DbDelete with an invalid bsp handle and verifies if
                              the return code is BioAPIERR_INVALID_BSP_HANDLE.
                  The relevant text in BioAPI 2.0 is quoted below from subclause 8.5.4.1.
                  _________________________________________________________
                  BioAPI_RETURN BioAPI BioSPI_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.
                  _________________________________________________________
                  8.5.4.2 Parameters
                        BSPHandle (input) - The handle of the attached BioAPI service provider.
                  _________________________________________________________
                  In order to determine conformance with respect to the text above, the following
                              steps are performed:

                        1)    Load the BSP under test.
                        2)    Attach the BSP under test.
                        3)    Invoke function BioSPI_DbCreate to create the specified database.
                        4)    Invoke function BioSPI_DbClose to close the created database.
                        5)    Invoke function BioSPI_DbDelete with an invalid module handle.
                        6)    Verify the return code; it is expected to be
                              BioAPIERR_INVALID_BSP_HANDLE.
                        7)    Detach and unload the BSP.

                  If any of the intermediate operations fails, an UNDECIDED conformity response is
                              issued.
            </description>

            <!-- UUID of the BSP under test -->
            <input name="_bspUuid"/>

            <!-- Timeout for the BioAPI_NOTIFY_INSERT event -->
            <input name="_inserttimeout"/>

            <!-- Indicates whether the BSP under test does not claim support for the
                              BioAPI_NOTIFY_SOURCE_PRESENT event notification -->
            <input name="_noSourcePresentSupported" />

            <!-- Timeout for the BioAPI_NOTIFY_SOURCE_PRESENT event -->
            <input name="_sourcepresenttimeout"/>

            <!-- Database Uuid to be opened -->
            <input name="_dbUuid"/>

            <!-- Invocation of the primary activity of this assertion with input parameter values
                              assigned from the assertion's parameters. -->
            <invoke activity="BioSPI_DbDelete">
                  <input name="bspUuid" var="_bspUuid"/>
                  <input name="inserttimeouttime" var="_inserttimeout"/>
                  <input name="nosourcepresentsupported"
                              var="_noSourcePresentSupported" />
                  <input name="sourcepresenttimeouttime"
                              var="_sourcepresenttimeout"/>
                  <input name="dbUuid" var="_dbUuid"/>
            </invoke>
```

```
        <!-- Activity bound to a function of the framework callback interface exposed by the
                         testing component.  This activity will be automatically invoked on each
                         incoming call to the function to which it is bound. -->
        <bind activity="EventHandler"
                    package="02c59458-0c46-1085-95d7-0002a5d5fd2e"
                    function="BioSPI_EventHandler"/>
    </assertion>

    <activity name="BioSPI_DbDelete">
        <input name="bspUuid"/>
        <input name="inserttimeouttime"/>
        <input name="nosourcepresentsupported" />
        <input name="sourcepresenttimeouttime"/>
        <input name="dbUuid"/>

        <!-- This assertion will use BSPHandle "1" for all BioSPI calls that require it -->
        <set name="_bsphandle" value="1"/>

        <!-- Invoke the functions BioSPI_BSPLoad and BioSPI_BSPAttach exposed by the BSP under
                         test.
                    The input value for the parameter "unitIDOrNull" is "0", therefore the
                         assertion will test a sensor unit chosen by the BSP. -->
        <invoke activity="LoadAndAttach"
                    package="02c59458-0c46-1085-95d7-0002a5d5fd2e"
                    break_on_break="true">
            <input name="bspUuid" var="bspUuid"/>
            <input name="bspVersion" value="32"/>
            <input name="unitIDOrNull" value="0"/>
            <input name="bspHandle" var="_bsphandle"/>
            <input name="eventtimeouttime" var="inserttimeouttime"/>
        </invoke>

        <invoke activity="PrepareDBTesting"
                    package="02c59458-0c46-1085-95d7-0002a5d5fd2e"
                    break_on_break="true">
            <input name="bspHandle" var="_bsphandle" />
            <input name="dbUuid" var="dbUuid" />
            <input name="nosourcepresentsupported" var="nosourcepresentsupported"/>
            <input name="sourcepresenttimeouttime" var="sourcepresenttimeouttime"/>
        </invoke>

        <!-- Invoke the function BioSPI_DbDelete with an invalid BSP handle -->
        <invoke function="BioSPI_DbDelete">
            <input name="BSPHandle" value="0"/>
            <input name="DbUuid" var="dbUuid"/>
            <return setvar="return"/>
        </invoke>

        <!-- Issue a conformity response.
                    If the condition specified in the <description> below is false, a FAIL
                         conformity response is issued, otherwise a PASS conformity response is
                         issued.-->
        <assert_condition>
            <description>
                    The function BioSPI_DbDelete has returned BioAPIERR_INVALID_BSP_HANDLE
            </description>
            <equal_to var1="return" var2="__BioAPIERR_BSP_INVALID_BSP_HANDLE"/>
        </assert_condition>

        <invoke activity="CleanUpDBTesting"
                    package="02c59458-0c46-1085-95d7-0002a5d5fd2e"
                    break_on_break="true">
            <input name="BSPHandle" var="_bsphandle" />
            <input name="dbUuid" var="dbUuid" />
        </invoke>

        <!-- Invoke the functions BioSPI_BSPDetach and BioSPI_BSPUnload -->
        <invoke activity="DetachAndUnload"
                    package="02c59458-0c46-1085-95d7-0002a5d5fd2e" >
            <input name="bspUuid" var="bspUuid" />
            <input name="BSPHandle" var="_bsphandle" />
        </invoke>
    </activity>

</package>
```