InterNational Committee for Information Technology Standards
INCITS Secretariat, Information Technology Industry Council (ITI)
1250 Eye St. NW, Washington, DC 20005
Telephone 202-737-8888; Fax 202-638-4922
Email: incits@itic.org

DOC: M1/06-0073
Date: January 27, 2006
Ref: Project 1703-D

Draft 4

Information technology - Conformance Testing Methodology for ANSI INCITS 358-2002, BioAPI Specification

Revision	Date	M1 Document #	Comments
1.0	Nov. 2, 2004	M1/04-0697	First Draft
2.0	May 3, 2005	M1/05-0283	Second Draft
3.0	June 22, 2005	M1/05-0403	Third Draft
4.0	January 27, 2006	M1/06-0073	Fourth Draft

Foreward 13
Introduction 14
1 Scope 15
2 Conformance 16
3 Normative References 17
4 Terminology 18
4.1 Terms and Definitions 18
4.2 Abbreviations 18
5 Conformance Testing Methodology 20
5.1 General 20
5.1.1 Implementation Under Test 20
5.1.2 Test Method 20
5.1.3 Standard Components and Interfaces of the BioAPI Architecture 22
5.1.4 Physical Architectures 23
5.2 Conformance Testing Models 24
5.3 Abstract Test Engine 26
6 General Properties of the Assertion Language 28
6.1 General 28
6.2 Variables 30
6.3 Built-in Variables 31
6.4 Representation of Integers 32
6.5 Representation of Booleans 32
6.6 Representation of Universally Unique Identifiers 32
6.7 Representation of Binary Data Blocks 33
6.8 XML Documents 33
7 Elements of the Assertion Language 34
7.1 Element <package> 34
7.1.1 Syntax 34
7.1.2 Semantics 34
7.1.3 Example (non-normative) 34
7.2 Element `<assertion>` (child of `<package>`) 35
7.2.1 Syntax 35
7.2.2 Semantics 35
7.2.3 Example (non-normative) 36
7.3 Element `<input>` (child of `<assertion>`) 36
7.3.1 Syntax 36
7.3.2 Semantics 36
7.3.3 Example (non-normative) 37
7.4 Element `<invoke>` (child of `<assertion>`) 37
7.4.1 Syntax 37
7.4.2 Semantics 37
7.4.3 Example (non-normative) 38
7.5 Element `<input>` (child of `<invoke>`) 38
7.5.1 Syntax 38
7.5.2 Semantics 38
7.5.3 Example (non-normative) 39
7.6 Element `<output>` (child of `<invoke>`) 39
7.6.1 Syntax 39
7.6.2 Semantics 39

7.7 Element <return> (child of <invoke>) 40
7.7.1 Syntax 40
7.7.2 Semantics 40
7.8 Element <bind> (child of <assertion>) 41
7.8.1 Syntax 41
7.8.2 Semantics 41
7.9 Element <activity> (child of <package>) 42
7.9.1 Syntax 42
7.9.2 Semantics 42
7.9.3 Examples (non-normative) 45
7.10 Element <input> (child of <activity>) 48
7.10.1 Syntax 48
7.10.2 Semantics 48
7.11 8.11 Element <output> (child of <activity>) 49
7.11.1 Syntax 49
7.11.2 Semantics 49
7.12 Element <set> 49
7.12.1 Syntax 49
7.12.2 Semantics 49
7.13 Element <add> 50
7.13.1 Syntax 50
7.13.2 Semantics 50
7.14 Element <subtract> 51
7.14.1 Syntax 51
7.14.2 Semantics 51
7.15 8.15 Element <invoke> (child of <activity>) 52
7.15.1 Syntax 52
7.15.2 Semantics 53
7.15.3 Examples (non-normative) 55
7.16 8.16 Element <only_if> 55
7.16.1 Syntax 55
7.16.2 Semantics 56
7.17 Element <wait_until> 57
7.17.1 Syntax 57
7.17.2 Semantics 57
7.18 Element <assert_condition> 58
7.18.1 Syntax 58
7.18.2 Semantics 59
7.19 Element <and> 60
7.19.1 Syntax 60
7.19.2 Semantics 60
7.20 Element <or> 61
7.20.1 Syntax 61
7.20.2 Semantics 61
7.21 Element <xor> 61
7.21.1 Syntax 61
7.21.2 Semantics 62
7.22 Element <not> 62
7.22.1 Syntax 62
7.22.2 Semantics 62

7.23 Element <equal_to> 63
7.23.1 Syntax 63
7.23.2 Semantics 63
7.24 Element <not_equal_to> 64
7.24.1 Syntax 64
7.24.2 Semantics 64
7.25 Element <greater_than> 64
7.25.1 Syntax 64
7.25.2 Semantics 64
7.26 Element <greater_than_or_equal_to> 64
7.26.1 Syntax 64
7.26.2 Semantics 65
7.27 Element <less_than> 65
7.27.1 Syntax 65
7.27.2 Semantics 65
7.28 Element <less_than_or_equal_to> 65
7.28.1 Syntax 65
7.28.2 Semantics 65
7.29 Element <same_as> 65
7.29.1 Syntax 65
7.29.2 Semantics 66
7.30 Element <not_same_as> 66
7.30.1 Syntax 66
7.30.2 Semantics 66
7.31 Element <existing> 67
7.31.1 Syntax 67
7.31.2 Semantics 67
7.32 Element <not_existing> 67
7.32.1 Syntax 67
7.32.2 Semantics 67
8 Standard Interface Functions 68
8.1 General 68
8.2 Parameter Groups 70
8.2.1 Parameter Group "Factors" 70
8.2.2 Parameter Group "Events" 71
8.2.3 Parameter Group "Biometric type" 72
8.2.4 Parameter Group "BIR header" 73
8.2.5 Parameter Group "BIR" 74
8.2.6 Parameter Group "Input BIR" 75
8.2.8 Parameter Group "Candidate" 78
8.2.9 Parameter Group "GUI state" 79
8.3 BioSPI_ModuleLoad 80
8.3.1 General 80
8.3.2 Function Invocation Scheme 80
8.3.3 Function Invocation Input 80
8.3.4 Function Invocation Output 81
8.4 BioSPI_ModuleUnload 81
8.4.1 General 81
8.4.2 Function Invocation Scheme 81
8.4.3 Function Invocation Input 81

8.4.4 Function Invocation Output 82
8.5 BioSPI_ModuleAttach 82
8.5.1 General 82
8.5.2 Function Invocation Scheme 82
8.5.3 Function Invocation Input 83
8.5.4 Function Invocation Output 83
8.6 BioSPI_ModuleDetach 83
8.6.1 General 83
8.6.2 Function Invocation Scheme 83
8.6.3 Function Invocation Input 83
8.6.4 Function Invocation Output 84
8.7 BioSPI_FreeBIRHandle 84
8.7.1 General 84
8.7.2 Function Invocation Scheme 84
8.7.3 Function Invocation Input 84
8.7.4 Function Invocation Output 84
8.8 BioSPI_GetBIRFromHandle 84
8.8.1 General 84
8.8.2 Function Invocation Scheme 84
8.8.3 Function Invocation Input 85
8.8.4 Function Invocation Output 85
8.9 BioSPI_GetHeaderFromHandle 85
8.9.1 General 85
8.9.2 Function Invocation Scheme 85
8.9.3 Function Invocation Input 86
8.9.4 Function Invocation Output 86
8.10 BioSPI_EnableEvents 86
8.10.1 General 86
8.10.2 Function Invocation Scheme 86
8.10.3 Function Invocation Input 86
8.10.4 Function Invocation Output 87
8.11 BioSPI_SetGUICallbacks 87
8.11.1 General 87
8.11.2 Function Invocation Scheme 87
8.11.3 Function Invocation Input 87
8.11.4 Function Invocation Output 88
8.12 BioSPI_SetStreamCallback 88
8.12.1 General 88
8.12.2 Function Invocation Scheme 88
8.12.3 Function Invocation Input 88
8.12.4 Function Invocation Output 89
8.13 BioSPI_StreamInputOutput 89
8.13.1 General 89
8.13.2 Function Invocation Scheme 89
8.13.3 Function Invocation Input 89
8.13.4 Function Invocation Output 90
8.14 BioSPI_Capture 90
8.14.1 General 90
8.14.2 Function Invocation Scheme 90
8.14.3 Function Invocation Input 91
8.14.4 Function Invocation Output 91

8.15 BioSPI_CreateTemplate 91
8.15.1 General 91
8.15.2 Function Invocation Scheme 92
8.15.3 Function Invocation Input 92
8.15.4 Function Invocation Output 93
8.16 BioSPI_Process 93
8.16.1 General 93
8.16.2 Function Invocation Scheme 93
8.16.3 Function Invocation Input 93
8.16.4 Function Invocation Output 94
8.17 BioSPI_VerifyMatch 94
8.17.1 General 94
8.17.2 Function Invocation Scheme 94
8.17.3 Function Invocation Input 95
8.17.4 Function Invocation Output 96
8.18 BioSPI_IdentifyMatch 97
8.18.1 General 97
8.18.2 Function Invocation Scheme 97
8.18.3 Function Invocation Input 98
8.18.4 Function Invocation Output 99
8.19 BioSPI_Enroll 99
8.19.1 General 99
8.19.2 Function Invocation Scheme 100
8.19.3 Function Invocation Input 100
8.19.4 Function Invocation Output 101
8.20 BioSPI_Verify 101
8.20.1 General 101
8.20.2 Function Invocation Scheme 101
8.20.3 Function Invocation Input 102
8.20.4 Function Invocation Output 103
8.21 BioSPI_Identify 104
8.21.1 General 104
8.21.2 Function Invocation Scheme 104
8.21.3 Function Invocation Input 105
8.21.4 Function Invocation Output 106
8.22 BioSPI_Import 107
8.22.1 General 107
8.22.2 Function Invocation Scheme 107
8.22.3 Function Invocation Input 107
8.22.4 Function Invocation Output 108
8.23 BioSPI_SetPowerMode 108
8.23.1 General 108
8.23.2 Function Invocation Scheme 108
8.23.3 Function Invocation Input 109
8.23.4 Function Invocation Output 109
8.24 BioSPI_DbOpen 109
8.24.1 General 109
8.24.2 Function Invocation Scheme 109
8.24.3 Function Invocation Input 109
8.24.4 Function Invocation Output 110
8.25 BioSPI_DbClose 110

8.25.1 General 110
8.25.2 Function Invocation Scheme 111
8.25.3 Function Invocation Input 111
8.25.4 Function Invocation Output 111
8.26 BioSPI_DbCreate 111
8.26.1 General 111
8.26.2 Function Invocation Scheme 111
8.26.3 Function Invocation Input 112
8.26.4 Function Invocation Output 112
8.27 BioSPI_DbDelete 112
8.27.1 General 112
8.27.2 Function Invocation Scheme 113
8.27.3 Function Invocation Input 113
8.27.4 Function Invocation Output 113
8.28 BioSPI_DbSetCursor 113
8.28.1 General 113
8.28.2 Function Invocation Scheme 113
8.28.3 Function Invocation Input 114
8.28.4 Function Invocation Output 114
8.29 BioSPI_DbFreeCursor 114
8.29.1 General 114
8.29.2 Function Invocation Scheme 114
8.29.3 Function Invocation Input 115
8.29.4 Function Invocation Output 115
8.30 BioSPI_DbStoreBIR 115
8.30.1 General 115
8.30.2 Function Invocation Scheme 115
8.30.3 Function Invocation Input 115
8.30.4 Function Invocation Output 116
8.31 BioSPI_DbGetBIR 116
8.31.1 General 116
8.31.2 Function Invocation Scheme 116
8.31.3 Function Invocation Input 116
8.31.4 Function Invocation Output 117
8.32 BioSPI_DbGetNextBIR 117
8.32.1 General 117
8.32.2 Function Invocation Scheme 117
8.32.3 Function Invocation Input 118
8.32.4 Function Invocation Output 118
8.33 BioSPI_DbQueryBIR 119
8.33.1 General 119
8.33.2 Function Invocation Scheme 119
8.33.3 Function Invocation Input 119
8.33.4 Function Invocation Output 120
8.34 BioSPI_DbDeleteBIR 120
8.34.1 General 120
8.34.2 Function Invocation Scheme 120
8.34.3 Function Invocation Input 120
8.34.4 Function Invocation Output 120
8.35 BioSPI_ModuleEventHandler 121
8.35.1 General 121

8.35.2 Bound Activity Parameters 121
8.35.3 Bound Activity Invocation Input 121
8.35.4 Bound Activity Invocation Output 121
8.35.5 Default Output 121
8.36 BioSPI_GUI_STATE_CALLBACK 121
8.36.1 General 121
8.36.2 Bound Activity Parameters 122
8.36.3 Bound Activity Invocation Input 122
8.36.4 Bound Activity Invocation Output 122
8.36.5 Default Output 123
8.37 BioSPI_GUI_STREAMING_CALLBACK 123
8.37.1 General 123
8.37.2 Bound Activity Parameters 123
8.37.3 Bound Activity Invocation Inputf 123
8.37.4 Bound Activity Invocation Output 123
8.37.5 Default Output 124
8.38 BioSPI_STREAM_CALLBACK 124
8.38.1 General 124
8.38.2 Bound Activity Parameters 124
8.38.3 Bound Activity Invocation Input 124
8.38.4 Bound Activity Invocation Output 124
8.38.5 Default Output 125
9 Predefined Variables 126
9.1 Variables Whose Value Never Changes 126
9.2 Variables Whose Value May Change 128
9.2.1 `__exit` 129
9.2.2 `__native_functions_called` 129
9.2.3 `__native_functions_returned` 129
10 Test Log 130
11 Test Report 133
12 Conformance Test Suite 135
12.1 General Concepts 135
12.2 Conformance Test Suite Structure 135
13 BioAPI Functions 138
14 BioAPI Conformance Features And Test Assertions 141
15 Correspondence Between Implementation Options and Test Assertions to be Executed 166
15.6 Testing BSPs of Category “Verification BSP” 166
15.7 Testing BSPs of Category “Identification BSP” 170
16 Test Assertions and Test Cases 176
16.1 Common activities 176
16.2 Feature 1. BioAPI Registry 187
16.2.3 **Assertion 1.1 BioAPI_Registry_Installation** 187
16.2.4 **Assertion 1.2 BioAPI_Registry_OpsSupported** 187
16.3 Feature 2. BioSPI Module Load 188
16.3.2 **Assertion 2.1 BioSPI_ModuleLoad_ValidParam** 188
16.3.3 **Assertion 2.2 BioSPI_ModuleLoad_InvalidUUID** 190
16.4 Feature 3. BioSPI Module Unload 191
16.4.2 **Assertion 3.1 BioSPI_ModuleUnload_ValidParam** 192
16.4.3 **Assertion 3.2 BioSPI_ModuleUnload_Unmatch** 193
16.4.4 **Assertion 3.3 BioSPI_ModuleUnload_InvalidUUID** 196
16.4.5 **Assertion 3.4 BioSPI_ModuleUnload_Confirm** 198

16.5 Feature 4. BioSPI Module Attach 200
16.5.2 **Assertion 4.1 BioSPI_ModuleAttach_ValidParam** 200
16.5.3 **Assertion 4.2 BioSPI_ModuleAttach_InvalidUUID** 204
16.5.4 **Assertion 4.3 BioSPI_ModuleAttach_InvalidVersion** 207
16.5.5 **Assertion 4.4 BioSPI_ModuleAttach_InvalidModuleHandle** 210
16.6 Feature 5. BioSPI Module Detach 213
16.6.2 **Assertion 5.1 BioSPI_ModuleDetach_ValidParam** 214
16.6.3 **Assertion 5.2 BioSPI_ModuleDetach_InvalidModuleHandle** 217
16.6.4 **Assertion 5.3 BioSPI_ModuleDetach_Confirm** 220
16.7 Feature 6. BioSPI Free BIR Handle 224
16.7.2 **Assertion 6.1 BioSPI_FreeBIRHandle_InvalidModuleHandle** 224
16.7.3 **Assertion 6.2 BioSPI_FreeBIRHandle_InvalidBIRHandle** 227
16.7.4 **Assertion 6.3 BioSPI_FreeBIRHandle_ValidParam** 230
16.8 Feature 7. BioSPI Get BIR From Handle 233
16.8.2 **Assertion 7.1 BioSPI_GetBIRFromHandle_ValidParam** 233
16.8.3 **Assertion 7.2 BioSPI_GetBIRFromHandle_InvalidModuleHandle** 236
16.8.4 **Assertion 7.3 BioSPI_GetBIRFromHandle_InvalidBIRHandle** 239
16.9 Feature 8. BioSPI Get Header from Handle 242
16.9.2 **Assertion 8.1 BioSPI_GetHeaderFromHandle_ValidParam** 242
16.9.3 **Assertion 8.2 BioSPI_GetHeaderfromHandle_InvalidModuleHandle** 246
16.9.4 **Assertion 8.3 BioSPI_GetHeaderfromHandle_InvalidBIRHandle** 249
16.9.5 **Assertion 8.4 BioSPI_GetHeaderFromHandle_ BIRHandleNotFreed** 253
16.10 Feature 9. BioSPI Enable Events 257
16.10.2 **Assertion 9.1 BioSPI_EnableEvents_ValidParam** 257
16.10.3 **Assertion 9.2 BioSPI_EnableEvents_InvalidModuleHandle** 261
16.11 Feature 13. BioSPI Capture 264
16.11.2 **Assertion 13.1 BioSPI_Capture_InvalidModuleHandle** 264
16.11.3 **Assertion 13.2 BioSPI_Capture_IntermediateProcessedBIR** 267
16.11.4 **Assertion 13.3 BioSPI_Capture_PurposeInHeader** 271
16.11.5 **Assertion 13.4 BioSPI_Capture_Serialization** 271
16.11.6 **Assertion 13.5 BioSPI_Capture_SerializationTimeout** 272
16.11.7 **Assertion 13.6 BioSPI_Capture_Timeout** 272
16.12 Feature 13a. Return of raw/audit data 272
16.12.2 **Assertion 13a.1 BioSPI_Capture_AuditData** 272
16.13 Feature 13b. Return of quality in the captured BIR header 277
16.13.2 **Assertion 13b.1 BioSPI_Capture_ReturnQuality** 277
16.14 Feature 13c. BIR signing (by BSP) 283
16.14.2 **Assertion 13c.1 BioSPI_Capture_BIRSigned** 283
16.15 Feature 13d. BIR encryption (by BSP) 283
16.15.2 **Assertion 13d.1 BioSPI_Capture_BIREncrypted** 283
16.16 Feature 14. BioSPI Create Template 284
16.16.2 **Assertion 14.1 BioSPI_CreateTemplate_OutputBIRPurpose** 284
16.16.3 **Assertion 14.2 BioSPI_CreateTemplate_OutputBIRDataType** 288
16.16.4 **Assertion 14.3 BioSPI_CreateTemplate_InputBIRDataType** 293
16.16.5 **Assertion 14.4 BioSPI_CreateTemplate_Purpose** 298
16.17 Feature 14a. Accept input of stored template to return update/adapted template 303
16.17.2 **Assertion 14a.1 BioSPI_CreateTemplate_StoredTemplateUnchanged** 303
16.18 Feature 14b. Acceptance of payload for inclusion of enrollment BIR 309
16.18.2 **Assertion 14b.1 BioSPI_CreateTemplate_PayloadSupported** 309
16.18.3 **Assertion 14b.2 BioSPI_CreateTemplate_PayloadNotCopied** 313
16.19 Feature 14c. Return of quality in the processed BIR header 313

16.19.2 **Assertion 14c.1 BioSPI_CreateTemplate_BIRHeaderQuality** 314
16.20 Feature 14d. BIR signing (by BSP) 318
16.20.2 **Assertion 14d.1 BioSPI_CreateTemplate_BIRSigned** 318
16.21 Feature 14e. BIR encryption (by BSP) 319
16.21.2 **Assertion 14e.1 BioSPI_CreateTemplate_BIREncrypted** 319
16.22 Feature 15. BioSPI Process 319
16.22.2 **Assertion 15.1 BioSPI_Process_ValidParam** 319
16.22.3 **Assertion 15.2 BioSPI_Process_CannotProcess** 323
16.22.4 **Assertion 15.3 BioSPI_Process_BuildsProcessedBIR** 323
16.22.5 **Assertion 15.4 BioSPI_Process_InputBIRDataType** 327
16.22.6 **Assertion 15.5 BioSPI_Process_OutputBIRPurpose** 333
16.23 Feature 15a. Return of quality in the processed BIR header 337
16.23.2 **Assertion 15a.1 BioSPI_Process_BIRHeaderQuality** 337
16.24 Feature 15b. BIR signing (by BSP) 342
16.24.2 **Assertion 15b.1 BioSPI_Process_BIRSigned** 342
16.25 Feature 15c. BIR encryption (by BSP) 342
16.25.2 **Assertion 15c.1 BioSPI_Process_BIREncrypted** 342
16.26 Feature 16. BioSPI Verify Match 342
16.26.2 **Assertion 16.1 BioSPI_VerifyMatch_ValidParam** 342
16.26.3 **Assertion 16.2 BioSPI_VerifyMatch_UnspecifiedFAR** 347
16.26.4 **Assertion 16.3 BioSPI_VerifyMatch_Inconsistent_Purpose** 352
16.27 Feature 16b. Model/template adaptation 356
16.27.2 **Assertion 16b.1 BioSPI_VerifyMatch_Adaptation** 356
16.28 Feature 16d. Return of achieved FRR score 362
16.28.2 **Assertion 16d.1 BioSPI_VerifyMatch_AchievedFRR** 362
16.29 Feature 16e. Return of payload 367
16.29.2 **Assertion 16e.1 BioSPI_VerifyMatch_Payload** 367
16.30 Feature 18. BioSPI Enroll 373
16.30.2 **Assertion 18.1 BioSPI_Enroll_ValidParam** 373
16.30.3 **Assertion 18.2 BioSPI_Enroll_InvalidPurpose** 376
16.30.4 **Assertion 18.3 BioSPI_Enroll_Timeout** 377
16.30.5 **Assertion 18.4 BioSPI_Enroll_InvalidTimeout** 377
16.30.6 **Assertion 18.5 BioSPI_Enroll_Payload** 377
16.31 Feature 18a. Template update 380
16.31.2 **Assertion 18a.1 BioSPI_Enroll_TemplateAdaptation** 380
16.32 Feature 18b. Acceptance of payload for inclusion of enrollment BIR 381
16.32.2 **Assertion 18b.1 BioSPI_Enroll_PayloadTooLarge** 381
16.33 Feature 18c. Return of raw/audit data 381
16.33.2 **Assertion 18c.1 BioSPI_Enroll_AuditData** 381
16.34 Feature 18d. Return of quality in the enrollment BIR header 386
16.34.2 **Assertion 18d.1 BioSPI_Enroll_BIRHeaderQuality** 386
16.35 Feature 18f. BIR signing (by BSP) 390
16.35.2 **Assertion 18f.1 BioSPI_Enroll_BIRSigned** 390
16.36 Feature 18g. BIR encryption (by BSP) 390
16.36.2 **Assertion 18g.1 BioSPI_Enroll_BIREncrypted** 391
16.37 Feature 19. BioSPI Verify 391
16.37.2 **Assertion 19.1 BioSPI_Verify_ValidParam** 391
16.37.3 **Assertion 19.2 BioSPI_Verify_FARUnspecified** 395
16.37.4 **Assertion 19.3 BioSPI_Verify_FRR** 399
16.38 Feature 19b. Model/template adaptation 404
16.38.2 **Assertion 19b.1 BioSPI_Verify_TemplateAdaptation** 404

16.39 Feature 19e. Return of payload 408
16.39.2 **Assertion 19e.1 BioSPI_Verify_Payload** 408
16.40 Feature 19f. Return of raw/audit data 412
16.40.2 **Assertion 19f.1 BioSPI_Verify_AuditData** 412
16.41 Feature 19h. BIR signing (by BSP) 418
16.41.2 **Assertion 19h.1 BioSPI_Verify_BIRSignedTemplateAdaptation** 418
16.42 Feature 19i. BIR encryption (by BSP) 419
16.42.2 **Assertion 19i.1 BioSPI_Verify_BIREncryptedTemplateAdaptation** 419
16.43 Feature 21. BioSPI Import 419
16.43.2 **Assertion 21.1 BioSPI_Import_ValidParam** 419
16.43.3 **Assertion 21.2 BioSPI_Import_InvalidModuleHandle** 423
16.43.4 **Assertion 21.3 BioSPI_Import_InvalidInputData** 426
16.43.5 **Assertion 21.4 BioSPI_Import_InvalidPurpose** 426
16.44 Feature 23. BioSPI Db Open 427
16.44.2 **Assertion 23.1 BioSPI_DbOpen_ValidParam** 427
16.44.3 **Assertion 23.2 BioSPI_DbOpen_InvalidModuleHandle** 430
16.44.4 **Assertion 23.3 BioSPI_DbOpen_InvalidDBName** 432
16.44.5 **Assertion 23.4 BioSPI_DbOpen_InvalidRequest** 433
16.45 Feature 24. BioSPI Db Close 433
16.45.2 **Assertion 24.1 BioSPI_DbClose_ValidParam** 433
16.45.3 **Assertion 24.2 BioSPI_DbClose_InvalidModuleHandle** 437
16.45.4 **Assertion 24.3 BioSPI_DbClose_InvalidDBHandle** 440
16.46 Feature 25. BioSPI Db Create 440
16.46.2 **Assertion 25.1 BioSPI_DbCreate_ValidParam** 440
16.46.3 **Assertion 25.2 BioSPI_DbCreate_InvalidModuleHandle** 443
16.46.4 **Assertion 25.3 BioSPI_DbCreate_DbProtected** 445
16.46.5 **Assertion 25.4 BioSPI_DbCreate_InvalidDBName** 448
16.46.6 **Assertion 25.5 BioSPI_DbCreate_InvalidRequest** 449
16.47 Feature 26. BioSPI Db Delete 449
16.47.2 **Assertion 26.1 BioSPI_DbDelete_ValidParam** 449
16.47.3 **Assertion 26.2 BioSPI_DbDelete_OpenDbProtected** 451
16.47.4 **Assertion 26.3 BioSPI_DbDelete_InvalidModuleHandle** 454
16.48 Feature 27. BioSPI Db Set Cursor 457
16.48.2 **Assertion 27.1 BioSPI_DbSetCursor_ValidParam** 457
16.48.3 **Assertion 27.2 BioSPI_DbSetCursor_RecordNotFound** 460
16.48.4 **Assertion 27.3 BioSPI_DbSetCursor_InvalidModuleHandle** 463
16.48.5 **Assertion 27.4 BioSPI_DbSetCursor_InvalidDBHandle** 467
16.49 Feature 28. BioSPI Db Free Cursor 467
16.49.2 **Assertion 28.1 BioSPI_DbFreeCursor_ValidParam** 467
16.49.3 **Assertion 28.2 BioSPI_DbFreeCursor_InvalidModuleHandle** 470
16.49.4 **Assertion 28.3 BioSPI_DbFreeCursor_InvalidCursor** 474
16.50 Feature 29. BioSPI Db Store BIR 477
16.50.2 **Assertion 29.1 BioSPI_DbStoreBIR_ValidParam** 477
16.50.3 **Assertion 29.2 BioSPI_DbStoreBIR_InvalidModuleHandle** 481
16.50.4 **Assertion 29.3 BioSPI_DbStoreBIR_InvalidDBHandle** 485
16.51 Feature 30. BioSPI Db Get BIR 485
16.51.2 **Assertion 30.1 BioSPI_DbGetBIR_ValidParam** 486
16.51.3 **Assertion 30.2 BioSPI_DbGetBIR_RecordNotFound** 489
16.51.4 **Assertion 30.3 BioSPI_DbGetBIR_InvalidModuleHandle** 493
16.51.5 **Assertion 30.4 BioSPI_DbGetBIR_InvalidDBHandle** 497
16.51.6 **Assertion 30.5 BioSPI_DbGetBIR_InvalidKeyValue** 498

16.52 Feature 31. BioSPI Db Get Next BIR 498
16.52.2 **Assertion 31.1 BioSPI_DbGetNextBIR_ValidParam** 498
16.52.3 **Assertion 31.2 BioSPI_DbGetNextBIR_InvalidModuleHandle** 503
16.52.4 **Assertion 31.3 BioSPI_DbGetNextBIR_InvalidCursor** 507
16.52.5 **Assertion 31.4 BioSPI_DbGetNextBIR_EndOfDatabase** 507
16.53 Feature 32. BioSPI Db Query BIR 507
16.53.2 **Assertion 32.1 BioSPI_DbQueryBIR_ValidParam** 508
16.53.3 **Assertion 32.2 BioSPI_DbQueryBIR_InvalidModuleHandle.** 512
16.53.4 **Assertion 32.3 BioSPI_DbQueryBIR_InvalidDBHandle** 517
16.54 Feature 33. BioSPI Db Delete BIR 517
16.54.2 **Assertion 33.1 BioSPI_DbDeleteBIR_ValidParam** 517
16.54.3 **Assertion 33.2 BioSPI_DbDeleteBIR_InvalidModuleHandle** 521
16.54.4 **Assertion 33.3 BioSPI_DbDeleteBIR_InvalidDBHandle** 524
16.55 Feature 101. BSP must implement all mandatory functions IAW SPI 525
16.55.2 **Assertion 101.1 BSP_MandatoryFunctionsImplemented** 525
16.56 Feature 102. BSP accepts all valid input parameters and returns valid outputs 525
16.56.2 **Assertion 102.1 BSP_ValidInOut** 525
16.57 Feature 103. Options implemented must be in accordance to spec (as shown in column 2 & 3 of Table) 526
16.57.2 **Assertion 103.1 BSP_Options_InSpec** 526
16.58 Feature 104. BSP provides all registry entries 526
16.58.3 **Assertion 104.1 BSP_Registry** 526
16.59 Feature 105. BSP has UUID according to data definition 527
16.59.2 **Assertion 105.1 BSP_UUID** 527
16.60 Feature 106. Conformant data structures -- (Biometric data according to 2.1 & 3.2 including the BIR) 527
16.60.3 **Assertion 106.1 DataConformant** 527
16.61 Feature 107. Registered valid Format Owner and Format Type 528
16.61.2 **Assertion 107.1 RegisteredFormatOwnerType** 528
16.62 Feature 108. Error handling according to 2.3 528
16.62.3 **Assertion 108.1 ErrorHandlingIAW_2.3** 528
16.63 Feature 109. If GUI BSP must provide it 529
16.63.2 **Assertion 109.1 BSP_GUI_Provided** 529
16.63.3 **Assertion 109.2 BSP_GUI_Provided_Cancel** 529
Annex B XML Schema of the test log 531
Annex C. Recommendations for Implementers 533
Bibliography 534

Foreward

Introduction

The BioAPI specification, describes a general application-programming interface that can work with any type of biometric application. The sponsoring organization for the BioAPI specification is the BioAPI Consortium, which was founded in 1998 and has more than 128 members.

The objective of the BioAPI specification cannot be completely achieved until biometric products can be tested to determine whether they conform to the specification. Conforming implementations are a necessary prerequisite for achieving interoperability among implementations; therefore there is a need for a standardized, generally accepted BioAPI conformance testing methodology, a set of test tools implementing this methodology, and a process for recognition and publishing of the conformity test results.

The standardization of conformance testing requires definition and acceptance of a common testing methodology, together with appropriate testing tools, methods and procedures. This would help to achieve comparability and wide acceptance of test results produced by different test laboratories, and minimize the need for repeated conformance testing of the same products.

The BioAPI specification describes an architecture composed of several components for use with any type of biometric application. This Standard applies to only critical functionalities of the Biometric Service Provider (BSP) component of the BioAPI specification and provides the framework, concepts, methodology for testing and the criteria to be achieved to claim conformity to BioAPI specification. Currently there are no existing standards specifying conformance testing methodologies for ANSI INCITS 358-2002. A project is underway in JTC 1 SC37 to develop a similar testing methodology for ISO/IEC FCD 19784-1 (the international version of the BioAPI specification). The proposed standard would limit its scope of work to the national version of BioAPI and to only document test procedures for critical characteristics of the ANSI version of the BioAPI standard.

ISO, IEC, NIST, IEEE, and other organizations have published a number of normative documents and white papers related to conformance testing methodologies. Rather than make normative references to these documents, this Standard incorporates appropriate excerpts of their text, in some cases paraphrasing the text, or adapting the provisions to the specific circumstances. Therefore, these documents are listed in the bibliography of this Standard, or are referred to in the body text of this Standard explicitly as appropriate.

Objective

It is proposed that this Standard:

a) Establish a framework for Conformance Testing Methodology for BioAPI-conformant Biometric Service Providers components that can be adapted for validation of these products, and would provide implementations of the BioAPI specification the ability to verify conformance with the specification.

b) Define requirements and guidelines for specifying conformance test suites and related test methods for measuring conformity of Biometric Service Provider components to BioAPI specification, define procedures to be followed before, during, and after conformance testing.

1 Scope

1.1 It is proposed that this Standard specify the concepts, framework, test methods, and criteria to be achieved to claim conformity of Biometric Service Providers to the BioAPI specification ANSI INCITS 358-2002.

1.2 It is proposed that the Standard provide guidelines for specifying conformance test suites, writing test assertions, and for defining the procedures to be followed during the conformance testing.

1.3 This Standard should concern only to the testing of conformity of the Biometric Service Providers to the BioAPI specification. It should not concern to the testing of any other characteristics of the Biometric products or any other types of testing of Biometric products (i.e., acceptance, performance, robustness, security, etc.)

1.4 It is proposed that the Standard focus only on the critical set of functions/features of the BioAPI specification and other requirements detailed in the conformance clause of the BioAPI specification. Section 15 of this contribution includes all the BioAPI functions/features and lists **the critical functions** we propose should be addressed by the Standard.

1.5 This standard includes conformance requirements related to error handling to the extent that the errors are a) generated by the BSP (not the framework), b) explicitly listed as an error return in the function being tested, and c) are feasible to be induced by a test application. Testing of all possible error conditions is beyond the scope of this standard.

2 Conformance

2.1 Implementations, such as conformance test suites, claiming conformance to this standard shall support conformance testing models described in this standard (see clause 6) and shall be able to execute any valid test assertion for the testing models that it supports, and that is written in the assertion language specified in clauses 7 through 10. A conformance test suite shall be able to process all the test assertions specified in clause 17 according to the methodology and the general principles and provisions specified in clause 6.

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

2.2 A conformance test suite shall be able to verify the syntactic correctness of any package (see clause 8.1) containing assertions or activities (or both) for any conformance testing model, including the testing models that the implementation does not support (if any).

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

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

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

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

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

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

3 Normative References

a) ANSI INCITS 358-2002, BioAPI Specification

b) Common Biometric Exchange Formats Framework (CBEFF) – NISTIR 6529-A

4 Terminology

4.1 Terms and Definitions

4.1.1 Abstract Test Suite: Generic compliance testing concepts and procedures for a given requirement.

4.1.2 Assertion: The specification for testing a conformance requirement in an IUT in the forms defined in this Standard. [ISO/IEC 13210]

4.1.3 Base standard: The standard for which a test method specification is written and/or a test method implementation is developed.

4.1.4 Conformance: [ISO/IEC Guide 25] Fulfillment by a product, process, or service of all relevant specified conformance requirements.[JTC-1]

4.1.5 Conformance Test Suites (CTS): Test software used to ascertain conformance to a specification or standards.

4.1.6 Implementation under test (IUT): That which implements the standard(s) being tested.

4.1.7 System under test (SUT): The computer system of hardware and software on which the implementation under test operates.

4.1.8 Test case: [ISO/IEC 9646-1] A specification of the actions required to achieve a specific test purpose or combination of test purposes.

4.1.9 Test purpose: [ISO/IEC 9646-1] A prose description of a narrowly defined objective of testing, focusing on a single conformance requirement.

4.2 Abbreviations

For the purposes of this Standard, the following abbreviations apply:

API	Application Programming Interface
ATS	Abstract Test Suite
BCS	BioAPI Conformity Statement
BIR	Biometric Information Record
BSP	Biometric Service Provider
CB	Control Board
CBEFF	Common Biometric Exchange Formats Framework
CTL	Conformance Testing Layer
CTS	Conformance Test Suite
IUT	Implementation under test

SBH	Standard Biometric Header
SPI	Service Provider Interface
SUT	System under test
UUID	Universally Unique Identifier

5 Conformance Testing Methodology

5.1 General

5.1.1 Implementation Under Test

5.1.1.1 The implementation under test (IUT) is the object that is being tested for conformity. For BioAPI specifications it is the software that has ‘implemented’ the specification. The software and supporting hardware constitute the IUT and shall be listed in both the test report and certificate of conformity.

5.1.1.2 Biometric products claiming to conform to the BioAPI specification are expected to conform to all the applicable conformity requirements in the Conformity clause of the BioAPI specification These requirements can be:

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

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

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

5.1.1.3 To evaluate the conformity of a particular implementation, whether a Biometric product or a part of such a product, it is necessary to have a statement of the capabilities that have been implemented in conformance with the BioAPI specification, so that the implementation can be tested for conformity against relevant requirements only. Such a statement is called a BioAPI Conformity Statement (BCS), and shall be prepared by the IUT supplier prior to the beginning of the Conformance Testing.

At a minimum, the BCS shall contain an itemized list of all mandatory, optional, and conditional conformity requirements of the BioAPI specification included in the IUT

5.1.2 Test Method

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

5.1.2.2 Assessing conformity of an implementation to the BioAPI specification standard includes testing the capabilities and behavior of the IUT with respect to the conformity requirements of the standard. The process of testing the IUT for conformity applies test methods to the Biometric products.

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

a) Analyze the BioAPI specification and its conformity clause, and develop documented test cases in form of test assertions. The test assertions are documented in this Standard and form a normative part of this Standard.

b) The test assertions are documented in the form of executable test scripts, which, in combination with applicable data files, shall constitute a Conformance Test Suite (CTS).

c) An IUT shall be subjected to various combinations of legal and illegal inputs, and the resulting output shall be compared to a set of corresponding “expected results.”

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

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

5.1.2.5 A test method implementation shall document that it conforms to this Standard and shall document any other test method specifications to which it claims to conform. Each test method implementation shall include the following:

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

b) Documented test cases, which will sufficiently assure conformity to the standard. The test cases shall be formally represented in the form of assertions using the assertions language documented in this Standard. These assertions can be submitted to the Conformance Test Suite for subsequent execution.

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

5.1.2.6 A test method implementation shall use the required assertion definitions, types, syntax, and constructs specified in this Standard. It shall use test result codes specified in this Standard for test results defined by this Standard.

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

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

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

c) Test Report production, which includes recording of all events occurred during the execution of each test case, including all test outcomes and test verdicts.

5.1.3 Standard Components and Interfaces of the BioAPI Architecture

5.1.3.1 This Standard specifies a methodology for assessing conformance of implementations of ANSI/INCITS 358-2002 (BioAPI 1.1). Three types of implementations of BioAPI 1.1 are distinguished (see Figure 1):

a) BioAPI application;

b) BioAPI framework;

c) BioAPI biometric service provider (BSP).

5.1.3.2 A BioAPI application is a software component (or set of software components) that uses the BioAPI interface specified in BioAPI 1.1, making one or more function calls to the BioAPI interface in the course of its execution.

5.1.3.3 A BioAPI framework is a software component (or set of software components) that implements the BioAPI interface specified in BioAPI 1.1 and realizes its prescribed behavior, making one or more function calls to the BioSPI interface specified in BioAPI 1.1 in the course of its execution.

5.1.3.4 A BioAPI biometric service provider is a software component (or set of software components) that implements the BioSPI interface specified in BioAPI 1.1 and realizes its prescribed behavior.

5.1.3.5 Besides the two main interfaces BioAPI and BioSPI, two other interfaces are specified in BioAPI 1.1, as indicated in the four following subclauses.

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

a) notifications of events related to BioAPI devices;

b) streaming data (grayscale bitmaps) sent during an operation performed by a BSP;

c) GUI state information sent during an operation performed by a BSP.

5.1.3.5.2 In this Standard, the interface mentioned in the previous subclause, implemented by BioAPI frameworks, is called the "framework callback interface."

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

a) notifications of events related to devices, originating in a BSP and relayed by the framework to the application;

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

c) GUI state information sent during an operation performed by a BSP and relayed by the framework to the application.

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

5.1.3.5.4 In this Standard, the interface mentioned in the previous subclause, optionally implemented by BioAPI applications, is called the "application callback interface".

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

5.1.3.7 This Standard specifies a BioAPI conformance testing methodology in terms of one standard components (the BioAPI BSP) and two standard interfaces (the BioSPI interface and the framework callback interface).

5.1.4 Physical Architectures

5.1.4.1 BioAPI 1.1 does not specify the details of loading in memory any one of the standard components of the BioAPI architecture, nor does it specify a physical association between standard components and nodes, nor does it limit the number of instances of each standard component within a node. All this information is therefore platform-specific or implementation-specific.

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

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

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

5.1.4.5 Given the abstractness of the BioAPI architecture specified in BioAPI 1.1, the conformance testing methodology specified in this Standard does not depend on any particular physical BioAPI architecture. The conformance testing methodology treats the standard components of the BioAPI architecture as abstract components implementing the interfaces and the behavior specified in BioAPI 1.1, with no assumptions being made on their physical realization.

5.2 Conformance Testing Models

5.2.1 The conformance testing methodology specified in this Standard addresses only one standard components of the BioAPI architecture, the BioAPI BSP. A conformance testing model for BSPs is defined within the methodology.

5.2.2 Implementations of the conformance testing methodology contain procedures for testing the conformance of a given implementation of the BioAPI BSP standard component.

5.2.3 For the purposes of this clause, the basic BioAPI architecture is described as comprising a normal BioAPI application, a normal BioAPI framework, and one or more normal BioAPI BSPs.

5.2.4 In the conformance testing model, a special testing component (called the "BSP-testing application") shall replace the normal application and the normal framework (see Figure 2). This testing component shall act both as a BioAPI application and as a BioAPI framework, and shall implement the framework callback interface. As a result, it shall appear to the BSP under test as a framework. The testing component shall be able to make calls to the BioSPI interface of the BSP under test.

5.2.5 In contrast to the above paragraph, it is appropriate for a CTS implementation to use the BioAPI reference implementation framework, or an enhanced or modified version of the BioAPI reference framework. In such a CTS implementation the CTS may use the BioAPI registry or other means such as configuration files for obtaining the necessary information about the BSP to perform testing.

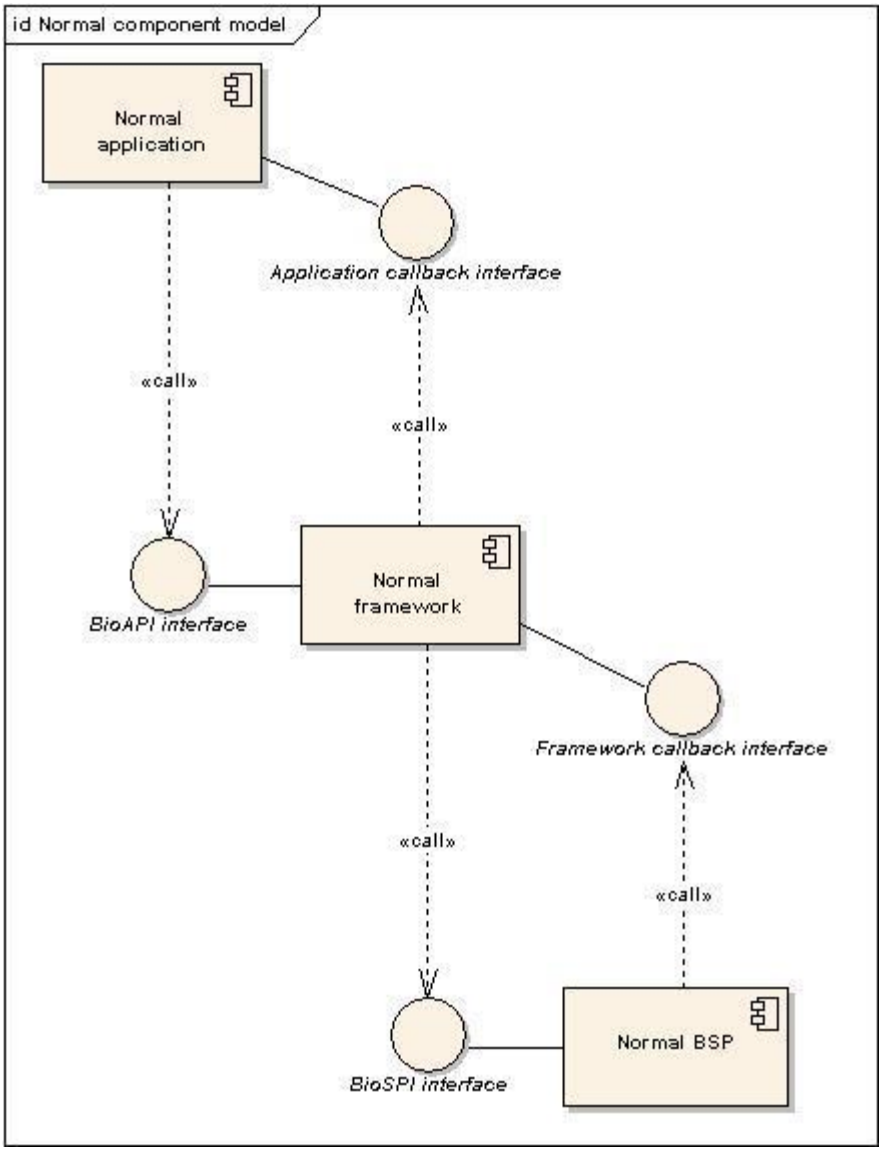

Figure 1 - Normal BioAPI component model

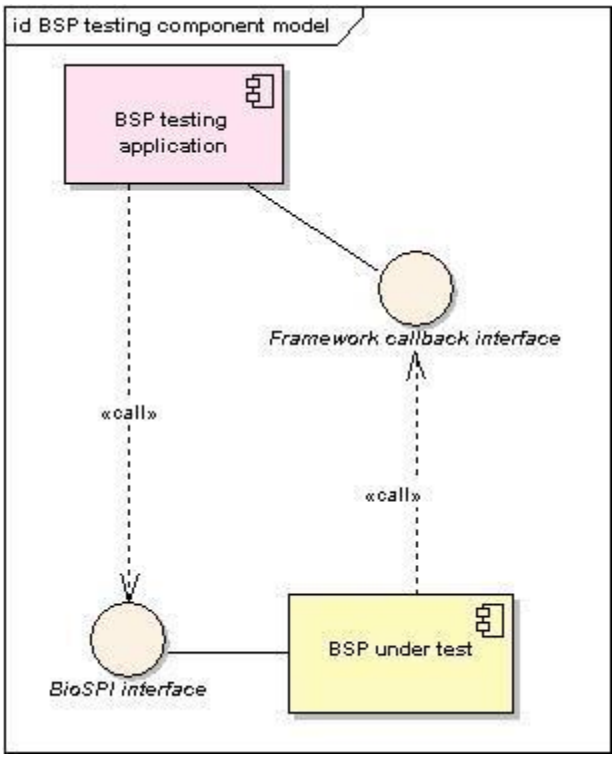

Figure 2 - Conformance testing model

5.3 Abstract Test Engine

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

5.3.2 The abstract test engine is a conceptual machine capable of performing conformance tests on a BioAPI BSP. The abstract test engine shall be able to do so by processing a formal assertion written in the assertion language specified in this Standard (see clause 6).

5.3.3 The abstract test engine shall have the ability to operate according to the conformance testing model (see 5.2). The abstract test engine shall be associated with the BSP-testing application.

5.3.4 During the processing of an assertion, the special testing component associated with the abstract test engine shall be able to make calls to the standard interface of the BSP under test. The testing component shall also be able to process incoming calls made by the BSP under test to the standard interfaces implemented by the testing component itself.

5.3.5 The outgoing calls being made by the testing component and the way of processing the incoming calls shall depend on the content of the assertion being processed.

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

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

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

5.3.7 The activities described in 5.3.6 b) may be indirectly observed by the abstract test engine as they may affect:

a) one or more subsequent calls that the standard component will make to the testing component; or

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

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

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

6 General Properties of the Assertion Language

6.1 General

6.1.1 Clauses 6 to 9 of this Standard specify a language whose purpose is to express assertions to be used in BioAPI conformance testing. The assertion language is an integral part of the BioAPI conformance testing methodology.

6.1.2 Subsequent clauses of this Standard contain a set of assertions written in the assertion language. Implementations of the conformance testing methodology (conformance test suites) that claim conformance to this Standard need to use all the assertions provided in this Standard, and only those assertions.

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

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

EXAMPLE (non-normative)

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

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

EXAMPLE (non-normative)

```
<?xml version='1.0' encoding="utf-8"?>
```

```
<package name="73668660-1583-1AD0-A3A5-09C0FF4756E3">
   <author>
      INCITS M1
   </author>
   <description>
      Abcde abcde abcde
   </description>

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

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

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

6.1.7 The means of assigning values to the input parameters of assertions is outside the scope of this Standard, but the values used in a test are documented in the standard test report (see clause 11).

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

6.1.9 Activities are represented as **`<activity>`** elements (see 8.9). An activity is a sequence, usually parameterized, of actions that may include invocation of functions of the standard interfaces (see clause 8). An activity may invoke other activities.

EXAMPLE (non-normative)

```
<?xml version='1.0' encoding="utf-8"?>
<package name="734ED660-1183-13D0-A3A5-0410FFD77AE9">
   <author>
      INCITS M1
   </author>
   <description>
      Abcde abcde abcde
   </description>

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

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

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

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

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

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

</package>
```

6.2 Variables

6.2.1 The names of variables in the assertion language shall consist of strings of ISO/IEC 10646-1 characters that match the "NCName" production in W3C XML Namespaces.

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

6.2.3 Global variables shall have a lifetime that extends over the processing of an entire assertion. They may be created within any activity (see 7.6.2.3, 7.7.2.3, 7.12.2.5.1, 7.17.2.9) but shall be associated with the entire assertion and shall not be destroyed until the processig of the assertion terminates. Global variables may also be created as input parameters of assertions (see 7.3.2.4).

6.2.4 Local variables may be created within any activity (see 7.5.2.5, 7.6.2.3, 7.7.2.3, 7.12.2.5.1, 7.17.2.9), shall be associated with that activity, and shall be destroyed as soon as that activity terminates.

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

6.2.6 In the assertion language, variables do not have a data type. The values of all variables are ISO/IEC 10646-1 character strings of unlimited length.

6.2.7 A value shall be interpreted as an integer in the following cases:

a) when it is evaluated by a numeric operation (see subclauses 7.23 to 7.28); or

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

6.2.8 In case a) of 6.2.7, a typical operation gets one or more values (character strings representing integers) as input and returns a value. Although the values may exist in some native numeric representation for some time during the execution of the operation, this fact is never reflected outside the operation.

6.2.9 In case b) of 6.2.7, the conversion from character string to integer is performed as part of the invocation of the function, and does not manifest itself in the syntax of the assertion language.

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

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

6.2.12 The same is true of data types other than integers, which are used on the standard interfaces. These native data types are enumerations, handles, addresses of functions, and addresses of data (see clause 8), and are all represented as character strings in the assertion language.

6.3 Built-in Variables

6.3.1 A number of global variables are built into the assertion language. Their names all begin with two consecutive LOW LINE characters ("_"). These variables are specified in clause 9.

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

NOTE - Names of global variables beginning with two consecutive LOW LINE characters ("_") cannot appear as the value of the **`setvar`** attribute of the elements **`<output>`**, **`<return>`**, and **`<wait_until>`** (see 7.6.2.3, 7.7.2.3 and 7.17.2.9.1, respectively), or as the value of the **`name`** attribute of the elements **`<input>`**, **`<set>`**, **`<add>`**, and **`<subtract>`** (see 7.3.2.3, 7.12.2.2, 7.13.2.2, and 7.14.2.2, respectively). Therefore, it is not possible to explicitly modify the value of a built-in variable.

6.3.3 In the execution of any activity, built-in variables shall only be updated during the time intervals in which the activity is interruptible (see 7.9.2.20). At the end of any such interval, the values of all related built-in variables shall be consistent with one another.

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

6.4 Representation of Integers

6.4.1 Non-negative integers shall be represented as strings of one or more ISO/IEC 10646-1 characters in the range DIGIT ZERO to DIGIT NINE ("**`0`**" to "**`9`**").

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

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

6.5 Representation of Booleans

6.5.1 The boolean TRUE shall be represented as the string of ISO/IEC 10646-1 characters "**`true`**". The boolean FALSE shall be represented as the string of ISO/IEC 10646-1 characters "**`false`**".

6.5.2 This representation is canonical.

6.6 Representation of Universally Unique Identifiers

6.6.1 Universally unique identifiers (UUIDs) shall be represented as strings of ISO/IEC 10646-1 characters. Each string shall contain characters from the union of the following sets:

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

b) LATIN CAPITAL LETTER A to LATIN CAPITAL LETTER F ("`A`" to "`F`"),

each representing a hexadecimal digit A through F;

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

d) HYPHEN-MINUS ("-").

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

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

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

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

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

6.6.3 The canonical representation of a UUID is the one in which the characters listed in 6.6.1 b) are not used.

6.7 Representation of Binary Data Blocks

6.7.1 Binary data blocks shall be represented as strings of ISO/IEC 10646-1 characters. Each string shall contain an even number (possibly zero) of characters from the union of the following sets:

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

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

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

6.7.2 The canonical representation of a binary data block is the one in which the characters listed in 6.7.1 b) are not used.

6.8 XML Documents

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

6.8.2 The root element of all such XML documents shall be the **<package>** element (see 7.1).

6.8.3 The XML documents shall be valid according to the XML Schema specified in Annex A.

7 Elements of the Assertion Language

7.1 Element <package>

7.1.1 Syntax

7.1.1.1 This element shall have the following attribute:

- **name** (required) **–** the value of this attribute shall be a valid package name (see 7.1.2.5).

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

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

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

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

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

7.1.2 Semantics

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

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

7.1.2.3 Assertions are not referenced from any package, but may be referenced from contexts external to the assertion language. In particular, they are referenced when setting up a test and within test reports (see clause 11).

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

7.1.2.5 The names of packages shall consist of strings of ISO/IEC 10646-1 characters that represent universally unique identifiers (see 6.6). Each package name shall be globally unique.

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

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

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

7.1.3 Example (non-normative)

```
<?xml version='1.0' encoding="utf-8"?>
<package name="734ED660-1183-13D0-A3A5-0410FFD77AE9">
   <author>
      INCITS M1
```

```
   </author>
   <description>
      Abcde abcde abcde
   </description>

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

7.2 Element *<assertion>* (child of *<package>*)

7.2.1 Syntax

7.2.1.1 This element shall have the following attributes:

a) **name** (required) – the value of this attribute shall be a valid assertion name (see 7.2.2.3);

b) **model** (required) – this attribute shall have the value "**BSPTesting**".

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

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

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

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

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

7.2.2 Semantics

7.2.2.1 The **model** indicates that the assertion tests BSPs and that the abstract test engine is associated with the BSP-testing applications.

NOTE - This attribute is specified here in order to simplify implementation of this Standard that are also implementations of other conformance testing methodologies where this attribute may have multiple possible values.

7.2.2.2 All calls required to be made to standard interface functions of the BSP under test as a result of processing the assertion shall be made by the testing component, and all incoming calls to the testing component shall be processed according to the content of the assertion.

7.2.2.3 The names of assertions shall consist of strings of ISO/IEC 10646-1 characters that match the "NCName" production in W3C XML Namespaces.

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

7.2.3 Example (non-normative)

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

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

7.3.1 Syntax

7.3.1.1 This element shall have the following attribute:

- **`name`** (required) – the value of this attribute shall be a valid global variable name (see 6.2), which is to be the name of an input parameter of the assertion.

7.3.1.2 The content of this element shall be empty.

7.3.2 Semantics

7.3.2.1 This element represents an input parameter of an assertion.

NOTE - Assertions cannot have output parameters.

7.3.2.2 Assertions with parameters cannot be processed unless a value is assigned to each parameter. The means of assigning values to input parameters of assertions is outside the scope of this Standard.

7.3.2.3 The value of the **`name`** attribute shall be a valid global variable name (see 6.2) and shall not begin with two consecutive LOW LINE characters ("_").

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

7.3.2.5 Assigning values to assertion parameters is an essential part of the preparation of a test. For some implementations under test, the results of a test may depend on the values assigned to

the parameters. The values used in a given test are documented in the standard test report (see clause 11).

7.3.2.6 The names of the input parameters of an assertion shall be valid global variable names (see 6.2). The names of all global variables (including input parameters) shall be distinct.

NOTE - The values assigned to assertion parameters cannot be changed during the processing of an assertion, because all the language elements that assign or modify a global variable (see 7.6.2.3, 7.7.2.3, 7.12.2.2, 7.13.2.2, 7.14.2.2, and 7.17.2.9.1) do not permit the use of an assertion parameter as the variable to be modified.

7.3.3 Example (non-normative)

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

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

7.4.1 Syntax

7.4.1.1 This element shall have the following attributes:

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

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

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

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

7.4.2 Semantics

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

7.4.2.2 The activity may contain **`<assert_condition>`** elements with a **`break_if_false`** attribute (see 7.18.2.3). If a break occurs during the execution of the activity, the processing of the entire assertion shall terminate.

7.4.2.3 The set of **`<input>`** elements of the invocation shall match the input parameters of the activity as specified in 7.5.2.2.

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

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

NOTE - This is similar to invoking an activity with output parameters from another activity without providing any **`<output>`** elements matching the output parameters. That is allowed, and is useful

when the invoking activity is not interested in the values of the output parameters of the invoked activity.

7.4.2.6 The **<invoke>** element of assertions shall not have a **break_on_break** attribute (see 7.15.2.6.5) and shall not have an **<only_if>** child element (see 8.16).

7.4.3 Example (non-normative)

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

7.5 Element <input> (child of <invoke>)

7.5.1 Syntax

7.5.1.1 This element shall have the following attributes:

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

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

c) **var** (optional) – if this attribute is present, its value shall be a valid name of a variable (see 6.2), whose value is to be assigned to the input parameter.

7.5.1.2 Exactly one of the attributes **value** and **var** shall be present.

7.5.1.3 The content of this element shall be empty.

7.5.2 Semantics

7.5.2.1 This element represents the assignment of a value to an input parameter of an activity or a standard interface function (see clause 8) in an invocation.

7.5.2.2 The set of **<input>** elements of an invocation shall match the input parameters of the activity or of the standard interface function being invoked, as specified in the two following subclauses.

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

7.5.2.2.2 In the case of function invocation, for each input parameter of the function, there shall be at most one **<input>** element in the invocation whose **name** attribute equals the name of the input parameter of the function. For each **<input>** element in the invocation, there shall be an

input parameter of the function whose name equals the **name** attribute of the former. The **<input>** elements in the invocation may appear in any order.

NOTE - Clause 8 specifies the names of all standard interface functions and the names of their input and output parameters, as they are to appear in the invocation of those functions.

7.5.2.3 If the **var** attribute is present, its value shall be the name of a variable (see 6.2) that already exists prior to the invocation. The value of the variable shall be assigned to the input parameter. The variable may be either global or local.

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

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

7.5.2.6 In the case of function invocation, the specified value shall be converted into a native form suitable for the corresponding parameter of the underlying standard interface function as specified in clause 8 and its individual subclauses.

7.5.3 Example (non-normative)

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

7.6 Element *<output>* (child of *<invoke>*)

7.6.1 Syntax

7.6.1.1 This element shall have the following attributes:

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

b) **setvar** (required) – the value of this attribute shall be a valid name of a variable (see 6.2), to which the value of the output parameter is to be assigned.

7.6.1.2 The content of this element shall be empty.

7.6.2 Semantics

7.6.2.1 This element represents the assignment of the value of an output parameter of an activity or standard interface function to a variable (see clause 8) in an invocation.

7.6.2.2 The set of **<output>** elements of an invocation shall match the output parameters of the activity or standard interface function being invoked, as specified in the two following subclauses.

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

7.6.2.2.2 In the case of function invocation, for each output parameter of the function, there shall be at most one **<output>** element in the invocation whose **name** attribute equals the name

of the output parameter of the function. For each **<output>** element in the invocation, there shall be an output parameter of the function whose name equals the **name** attribute of the former. The **<output>** elements in the invocation may appear in any order.

NOTE - Clause 8 specifies the names of all standard interface functions and the names of their input and output parameters, as they are to appear in the invocation of those functions.

7.6.2.3 The value of the **setvar** attribute shall be a valid global or local variable name (see 6.2), shall not be an input parameter of the assertion being processed (see 8.3), and shall not begin with two consecutive LOW LINE characters ("_"). The variable may already exist prior to the invocation, or may be a new variable that is to be created when the invoked activity or function terminates.

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

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

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

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

7.6.2.7 In the case of function invocation, the value of the corresponding output parameter of the underlying standard interface function shall be converted from its native form into a character string as specified in the individual subclauses of clause 8, and the variable shall be set to the resulting character string.

7.7 Element *<return>* (child of *<invoke>*)

7.7.1 Syntax

7.7.1.1 This element shall have the following attribute:

- **setvar** (required) – the value of this attribute shall be a valid name of a variable (see 6.2), to which the return value is to be assigned.

7.7.1.2 The content of this element shall be empty.

7.7.2 Semantics

7.7.2.1 This element represents the assignment of the return value of a standard interface function to a variable (see clause 8 and its individual subclauses) in an invocation.

7.7.2.2 There shall be at most one **<return>** element in the invocation of any standard interface function.

7.7.2.3 The value of the **setvar** attribute shall be a valid global or local variable name (see 6.2), shall not be an input parameter of the assertion being processed (see 8.3), and shall not begin

with two consecutive LOW LINE characters ("_"). This variable may already exist prior to the invocation, or may be a new variable that is to be created when the invoked activity or function terminates.

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

7.7.2.5 The return value of the standard interface function shall be converted from its native form into a character string as specified in clause 8, and the variable shall be set to the resulting character string.

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

7.8.1 Syntax

7.8.1.1 This element shall have the following attributes:

a) **`function`** (required) – the value of this attribute shall be the name of a standard interface function (see clause 8);

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

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

7.8.1.2 The content of this element shall be empty.

7.8.2 Semantics

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

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

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

7.8.2.3 The standard interface function identified by the **`function`** attribute shall be one of the functions exposed by the testing component.

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

7.8.2.5 The bound activity shall have input and output parameters as specified in the subclause of clause 8 with the heading "Bound Activity Parameters" relative to the standard interface function identified by the **`function`** attribute.

7.8.2.6 The bound activity shall be invoked each time the testing component receives an incoming call to the standard interface function, as specified in 7.9.2.7.

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

7.9.1 Syntax

7.9.1.1 This element shall have the following attributes:

a) **`name`** (required) **–** the value of this attribute shall be a valid activity name (see 7.9.2.7);

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

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

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

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

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

1) **`<set>`** – this element represents the assignment of a value to a new variable or to an existing variable, and is specified in 7.12;

2) **`<add>`** – this element represents the addition of an integer to the value (representing an integer) of an existing variable, and is specified in 7.13;

3) **`<subtract>`** – this element represents the subtraction of an integer from the value (representing an integer) of an existing variable, and is specified in 7.14;

4) **`<wait_until>`** – this element represents a suspension of the execution of the current activity until a condition is verified, and is specified in **Error! Reference source not found.**;

5) **`<assert_condition>`** – this element represents the issuance of a conformity response based on a given condition, and is specified in 7.18;

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

7.9.2 Semantics

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

a) assigning a variable;

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

c) invoking a standard interface function;

d) issuing a conformity response based on a condition;

e) invoking an activity.

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

7.9.2.3 An activity shall have a priority, which can be low, medium, or high. At most one activity may be in execution at any given time during the processing of an assertion. The abstract test engine shall have the ability to interrupt the execution of an activity in order to execute an activity with a higher priority (see 7.9.2.20).

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

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

7.9.2.6 When one of the following elements:

a) a **`<wait_until>`** element; or

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

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

7.9.2.7 When the testing component receives an incoming call, the actions specified in the four following subclauses shall be performed.

7.9.2.7.1 If there is an activity bound to the function being called (see 7.8), the abstract test engine shall interrupt the execution of the current low-priority activity and shall invoke the bound activity (but see 7.9.2.8) with its priority set as follows:

b) if the function belongs to the BioSPI interface, the priority shall be medium;

c) otherwise, the priority shall be high.

7.9.2.7.2 The input parameters of the bound activity invocation shall be set from the native input parameters of the incoming call as specified in the subclause of clause 8 with the heading "Bound Activity Invocation Input" relative to the standard interface function.

7.9.2.7.3 When the bound activity terminates, the native output parameters and the return value of the incoming call shall be set from the output parameters of the activity as specified in the subclause of clause 8 with the heading "Bound Activity Invocation Output" relative to the standard interface function.

7.9.2.7.4 If there is no activity bound to the function being called, then the abstract test engine shall set the native output parameters and return value of the incoming call as specified in the individual subclauses of clause 8 with the heading "Default Output".

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

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

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

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

a) if the invocation of the activity provides an **`<input>`** element for that output parameter, the provided value (see 7.5.2.5);

b) otherwise, an empty string.

7.9.2.12 The elements of the activity shall then be processed in order, from the first element to the last element, unless a break occurs as specified in 7.9.2.16.

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

a) if the invocation of the activity provides an **`<output>`** element for that output parameter, the final value of the output parameter shall be assigned to the variable referenced in the **`<output>`** element as specified in 7.6.2.3;

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

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

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

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

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

b) if an activity contains an **`<invoke>`** element with a **`break_on_break`** attribute with a value of "**`true`**", when the invoked activity terminates because of a break (see 7.9.2.17), a break is said to occur within the invoking activity.

7.9.2.17 When a break occurs within an activity (because of either 7.9.2.16 a) or b)), the abstract test engine shall:

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

b) skip the assignment of the values of the output parameters to the variables

referenced in the corresponding **<output>** elements in the invocation of the activity (if any);

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

NOTE 1 - Item b) implies that, if a variable corresponding to an output parameter did not exist before the invocation, it is not created when the invoking activity is resumed.

NOTE 2 – If 7.9.2.16 b) applies to the invoking activity, the execution of the invoking activity will be abandoned immediately after being resumed.

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

7.9.2.19 An activity may not be interrupted by another activity with the same or a lower priority (see 7.9.2.8).

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

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

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

NOTE - Interruptions are allowed while the function is in execution.

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

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

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

7.9.2.21 The names of activities shall consist of strings of ISO/IEC 10646-1 characters that match the "NCName" production in W3C XML Namespaces.

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

7.9.3 Examples (non-normative)

```
<?xml version='1.0' encoding="utf-8"?>
<package name="73668660-1583-1AD0-A3A5-09C0FF4756E3">
   <author>
      INCITS M1
   </author>
   <description>
      Abcde abcde abcde
   </description>

<activity name="LoadAndAttach">
   <input name="BSPUuid"/>
```

```
   <input name="BSPVersion"/>
   <input name="deviceIDOrNull"/>
   <input name="BSP"/>
   <input name="eventtimeouttime"/>

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

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

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

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

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

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

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

</activity>


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

   <set name="_BSPUuid" var="BSPUuid"/>
```

```
   <set name="_context" var="context"/>

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

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

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

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

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

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

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

</activity>
```

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

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

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

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

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

 ...
</activity>

</package>
```

7.10 Element *<input>* (child of *<activity>*)

7.10.1 Syntax

7.10.1.1 This element shall have the following attribute:

- **name** (required) – the value of this attribute shall be a valid local variable name (see 6.2), which is to be the name of an input parameter of the activity.

7.10.1.2 The content of this element shall be empty.

7.10.2 Semantics

7.10.2.1 This element represents an input parameter of an activity.

7.10.2.2 Input parameters of activities are local variables (see 6.2).

7.10.2.3 The names of the input parameters of an activity shall be valid local variable names (see 6.2).

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

7.11 8.11 Element `<output>` (child of `<activity>`)

7.11.1 Syntax

7.11.1.1 This element shall have the following attribute:

- **`name`** (required) – the value of this attribute shall be a valid local variable name (see 6.2), which is to be the name of an output parameter of the activity.

7.11.1.2 The content of this element shall be empty.

7.11.2 Semantics

7.11.2.1 This element represents an output parameter of an activity.

7.11.2.2 Output parameters of activities are local variables (see 6.2).

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

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

7.12 Element `<set>`

7.12.1 Syntax

7.12.1.1 This element shall have the following attributes:

a) **`name`** (required) – the value of this attribute shall be a valid name of a variable (see 6.2), which is to be assigned a value (or created and then assigned a value);

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

c) **`var`** (optional) – if this attribute is present, its value shall be a valid name of a variable (see 6.2), whose value is to be assigned to the variable referenced in a);

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

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

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

7.12.2 Semantics

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

7.12.2.2 The value of the **`name`** attribute shall be a valid global or local variable name (see 6.2), shall not be an input parameter of the assertion being processed (see 8.3), and shall not begin with two consecutive LOW LINE characters ("_"). The variable may exist prior to the assignment or may be a new variable created by the assignment.

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

7.12.2.4 The result of the evaluation shall be a valid representation of a boolean (see 6.5).

7.12.2.5 If the result of the evaluation is "**`true`**", the variable whose name is the value of the **`name`** attribute shall be set as specified in the three following subclauses.

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

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

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

7.13 Element <add>

7.13.1 Syntax

7.13.1.1 This element shall have the following attributes:

a) **`name`** (required) – the value of this attribute shall be a valid name of a variable (see 6.2) whose value is to be modified;

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

c) **`var`** (optional) – if this attribute is present, its value shall be a valid name of a variable (see 6.2) whose value represents an integer to be added to the current value of the variable referenced in a);

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

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

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

7.13.2 Semantics

7.13.2.1 This element represents the addition of an integer to the value (representing an integer) of a variable (see 6.4), possibly subject to a condition.

7.13.2.2 The value of the **`name`** attribute shall be a valid global or local variable name (see 6.2), shall not be an input parameter of the assertion being processed (see 8.3), and shall not begin with two consecutive LOW LINE characters ("_"). The variable shall exist prior to the assignment and its value shall be a valid representation of an integer (see 6.4).

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

7.13.2.4 The result of the evaluation shall be a valid representation of a boolean (see 6.5).

7.13.2.5 If the result of the evaluation is "**`true`**", the variable whose name is the value of the **`name`** attribute shall be set as specified in the three following subclauses.

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

7.13.2.5.2 If the **`var`** attribute is present, its value shall be the name of an existing variable. The value of that variable shall be a valid representation of an integer (see 6.4). This integer is the second operand.

7.13.2.5.3 If the **`value`** attribute is present, its value shall be a valid representation of an integer (see 6.4). This integer is the second operand.

7.14 Element <subtract>

7.14.1 Syntax

7.14.1.1 This element shall have the following attributes:

a) **`name`** (required) – the value of this attribute shall be a valid name of a variable (see 6.2) whose value is to be modified;

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

c) **`var`** (optional) – if this attribute is present, its value shall be a valid name of a variable (see 6.2) whose value represents an integer to be subtracted from the current value of the variable referenced in a);

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

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

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

7.14.2 Semantics

7.14.2.1 This element represents the subtraction of an integer from the value (representing an integer) of a variable (see 6.4), possibly subject to a condition.

7.14.2.2 The value of the **`name`** attribute shall be a valid global or local variable name (see 6.2), shall not be an input parameter of the assertion being processed (see 8.3), and shall not begin with two consecutive LOW LINE characters ("_"). The variable shall exist prior to the assignment and its value shall be a valid representation of an integer (see 6.4).

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

7.14.2.4 The result of the evaluation shall be a valid representation of a boolean (see 6.5).

7.14.2.5 If the result of the evaluation is "**`true`**", the variable whose name is the value of the **`name`** attribute shall be set as specified in the three following subclauses.

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

7.14.2.5.2 If the **`var`** attribute is present, its value shall be the name of an existing variable. The value of that variable shall be a valid representation of an integer (see 6.4). This integer is the second operand.

7.14.2.5.3 If the **`value`** attribute is present, its value shall be a valid representation of an integer (see 6.4). This integer is the second operand.

7.15 8.15 Element `<invoke>` (child of `<activity>`)

7.15.1 Syntax

7.15.1.1 This element shall have the following attributes:

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

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

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

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

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

f) **`timeout_var`** (optional) **–** if this attribute is present, its value shall be a valid name of a variable (see 6.2), whose value indicates the maximum duration of time (in milliseconds) allowed for the invocation;

g) **`setvar`** (optional) – if this attribute is present, its value shall be a valid variable name (see 6.2);

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

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

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

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

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

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

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

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

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

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

7.15.2 Semantics

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

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

7.15.2.3 The result of the evaluation shall be a valid representation of a boolean (see 6.5).

7.15.2.4 If the result of the evaluation is "**true**", the invocation shall be performed as specified in the following subclauses, otherwise no further action shall be performed for this **<invoke>** element.

7.15.2.5 In case of activity invocation, the activity being invoked shall be an existing activity that may be either in the same package as the invoking activity, or in a different package (see 7.15.2.6.4). In case of function invocation, the function being invoked shall be one of the standard interface functions specified in clause 8.

7.15.2.6 In the case of activity invocation, the five following subclauses apply.

7.15.2.6.1 The set of **<input>** elements of the invocation shall match the input parameters of the activity as specified in 7.5.2.2.

7.15.2.6.2 The set of **<output>** elements of the invocation shall match the output parameters of the activity as specified in 7.6.2.2.

7.15.2.6.3 The **<return>** element shall not be present.

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

7.15.2.6.5 If the attribute **break_on_break** is present and its value is "**true**", then if the activity being invoked is abandoned because of a break (see 7.9.2.16), the current activity shall be abandoned as well.

7.15.2.7 In the case of function invocation, the three following subclauses apply.

7.15.2.7.1 The set of **`<input>`** elements of the invocation shall match the input parameters of the function as specified in 7.5.2.2.

7.15.2.7.2 The set of **`<output>`** elements of the invocation shall match the output parameters of the function as specified in 7.6.2.2.

7.15.2.7.3 The **`<return>`** element may be present (see 7.7.2.2).

7.15.2.8 It the **`timeout_value`** attribute is present, its value shall be a valid representation of an integer (see 6.4) that specifies the maximum duration allowed for the invocation (in milliseconds). If the **`timeout_var`** attribute is present, its value shall be the name of an existing variable and the value of that variable shall be a valid representation of an integer (see 6.4) that specifies the maximum duration. Otherwise, there shall be no maximum duration.

NOTE - An implementation of this Standard may abandon the execution of a test that has lasted beyond a reasonable amount of time. Such a behavior would not be in violation of this subclause.

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

7.15.2.10 If the **`setvar`** attribute is present, the four following subclauses apply.

7.15.2.10.1 The value of the **`setvar`** attribute shall be a valid global or local variable name (see 6.2), shall not be an input parameter of the assertion being processed (see 8.3), and shall not begin with two consecutive LOW LINE characters ("_"). The variable may exist prior to the assignment or may be a new variable created by the assignment.

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

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

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

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

7.15.2.12 Before the activity or function being invoked starts, all input parameters for which an **`<input>`** element was provided in the invocation (if any) shall be set from the variables or values provided in those **`<input>`** elements, as specified in 7.5.2.5. The remaining input parameters shall be assigned from an empty string.

7.15.2.13 If the invoked activity or function terminates without exceeding the maximum duration, the two following subclauses apply.

7.15.2.13.1 The values of each output parameter for which an **<output>** element was provided in the invocation (if any) shall be assigned to the variable referenced in the **<output>** element as specified in 7.6.2.3.

7.15.2.13.2 In case of function invocation, if there is a **<return>** element in the invocation, the return value shall be assigned to the variable referenced in the **<return>** element as specified in 7.7.2.3.

7.15.2.14 If the maximum duration is exceeded before the invoked function returns, the two following subclauses apply.

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

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

7.15.2.15 The execution of the invoking activity (if any) shall then resume (but see 7.9.2.16).

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

7.15.3 Examples (non-normative)

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

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

7.16 8.16 Element <only_if>

7.16.1 Syntax

7.16.1.1 This element shall have the following attribute:

- **var** (optional) **–** if this attribute is present, its value shall be a valid variable name (see 6.2).

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

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

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

1) **<and>** – this element represents the logical operator AND, and is

specified in 7.19;

2) **<or>** – this element represents the logical operator OR, and is specified in 7.20;

3) **<xor>** – this element represents the logical operator XOR, and is specified in 7.21;

4) **<not>** – this element represents the logical operator NOT, and is specified in 7.22;

5) **<equal_to>** – this element represents the numeric operator "equal_to", and is specified in 7.23;

6) **<not_equal_to>** – this element represents the numeric operator "not equal to", and is specified in 7.24;

7) **<greater_than>** – this element represents the numeric operator "greater than", and is specified in 7.25;

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

9) **<less_than>** – this element represents the numeric operator "less than", and is specified in 7.27;

10) **<less_than_or_equal_to>** – this element represents the numeric operator "less than or equal to", and is specified in 7.28;

11) **<same_as>** – this element represents the character-string operator "equals", and is specified in 7.29;

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

13) **<existing>** – this element expresses the condition that a variable exists, and is specified in 7.31;

14) **<not_existing>** – this element expresses the condition that a variable does not exist, and is specified in 7.32.

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

7.16.2 Semantics

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

7.16.2.2 If the **var** attribute is present, its value shall be the name of an existing variable, and the value of that variable shall be a valid representation of a boolean (see 6.5). The result of the evaluation shall be "**true**" if the value of the variable is "**true**", and shall be "**false**" otherwise.

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

7.16.2.4 If there are child elements, the result of the evaluation of each child element shall be a valid representation of a boolean (see 6.5). The result of the evaluation of the condition shall be "**true**" if all the child elements evaluate to "**true**", and shall be "**false**" otherwise.

7.16.2.5 The condition is evaluated during the processing of the parent element (see 7.12.2.3 and 7.15.2.2). If the condition evaluates to "**true**", then the containing element is processed normally (as though it did not have an **<only_if>** child element), otherwise its processing has no effect.

7.17 Element *<wait_until>*

7.17.1 Syntax

7.17.1.1 This element shall have the following attributes:

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

b) **timeout_var** (optional) **–** if this attribute is present, its value shall be a valid name of a variable (see 6.2), whose value indicates the maximum duration of time (in milliseconds) allowed for the wait;

c) **setvar** (optional) – if this attribute is present, its value shall be a valid variable name (see 6.2);

d) **var** (optional) – if this attribute is present, its value shall be a valid variable name (see 6.2).

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

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

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

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

7.17.2 Semantics

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

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

7.17.2.3 If the **var** attribute is present, its value shall be the name of an existing variable, and the value of that variable shall be a valid representation of a boolean (see 6.5). The result of the evaluation shall be "**true**" if the value of the variable is "**true**", and shall be "**false**" otherwise.

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

7.17.2.5 If there are child elements, the result of the evaluation of each child element shall be a valid representation of a boolean (see 6.5). The result of the evaluation of the condition shall be "**`true`**" if all the child elements evaluate to "**`true`**", and shall be "**`false`**" otherwise.

7.17.2.6 If the **`timeout_value`** attribute is present, its value shall be a valid representation of an integer (see 6.4) that specifies the maximum duration allowed for the wait (in milliseconds). If the **`timeout_var`** attribute is present, its value shall be the name of an existing variable and the value of that variable shall be a valid representation of an integer (see 6.4) that specifies the maximum duration. Otherwise, the wait shall last (conceptually) for an indefinite time.

NOTE - The NOTE to 7.15.2.8 applies.

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

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

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

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

7.17.2.9 If the **`setvar`** attribute is present, the four following subclauses apply.

7.17.2.9.1 The value of the **`setvar`** attribute shall be a valid global or local variable name (see 6.2), shall not be an input parameter of the assertion being processed (see 8.3), and shall not begin with two consecutive LOW LINE characters ("_"). The variable may exist prior to the assignment or may be a new variable created by the assignment.

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

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

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

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

7.18 Element `<assert_condition>`

7.18.1 Syntax

7.18.1.1 This element shall have the following attributes:

a) **`response_if_true`** (optional) – if this attribute is present, it shall have one of

the values "**pass**", "**undecided**"; it indicates which conformity response shall be issued when the condition is "**true**"; the default is "**pass**";

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

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

d) **var** (optional) – if this attribute is present, its value shall be a valid variable name (see 6.2).

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

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

7.18.2 Semantics

7.18.2.1 This element specifies:

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

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

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

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

7.18.2.3 The **break_if_false** attribute indicates whether a result of "**false**" for the condition shall cause the abstract test engine to abandon (break) the execution of the current activity (see 7.9.2.17). It supports those common cases in which a series of actions – following the issuance of a certain conformity response – are meaningless or inappropriate.

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

7.18.2.5 If the **var** attribute is present, its value shall be the name of an existing variable, and the value of that variable shall be a valid representation of a boolean (see 6.5). The result of the evaluation shall be "**true**" if the value of the variable is "**true**", and shall be "**false**" otherwise.

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

7.18.2.7 If there are child elements, the result of the evaluation of each child element shall be a valid representation of a boolean (see 6.5). The result of the evaluation of the condition shall be "**`true`**" if all the child elements evaluate to "**`true`**", and shall be "**`false`**" otherwise.

7.18.2.8 If the condition evaluates to "**`true`**", then the corresponding conformity response (see 7.18.2.1 b)) shall be issued as specified in clause 10.

7.18.2.9 If the condition evaluates to "**`false`**", then the corresponding conformity response (see 7.18.2.1 c)) shall be issued as specified in clause 10; in addition, if the **`break_if_false`** attribute is "**`true`**", the current activity shall be abandoned (see 7.9.2.17).

7.19 Element <and>

7.19.1 Syntax

7.19.1.1 This element shall have the following attributes:

a) **`var1`** (optional) – if this attribute is present, its value shall be a valid variable name (see 6.2);

b) **`var2`** (optional) – if this attribute is present, its value shall be a valid variable name (see 6.2).

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

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

7.19.2 Semantics

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

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

7.19.2.3 If the **`var1`** attribute is present, its value shall be the name of an existing variable, and the value of that variable shall be a valid representation of a boolean (see 6.5).

7.19.2.4 If the **`var2`** attribute is present, its value shall be the name of an existing variable, and the value of that variable shall be a valid representation of a boolean (see 6.5).

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

7.19.2.6 Each operand shall be a valid representation of a boolean (see 6.5). The result of the evaluation shall be "**`true`**" if the operands are all "**`true`**", otherwise it shall be "**`false`**".

7.20 Element <or>

7.20.1 Syntax

7.20.1.1 This element shall have the following attributes:

a) **var1** (optional) – if this attribute is present, its value shall be a valid variable name (see 6.2);

b) **var2** (optional) – if this attribute is present, its value shall be a valid variable name (see 6.2).

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

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

7.20.2 Semantics

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

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

7.20.2.3 If the **var1** attribute is present, its value shall be the name of an existing variable, and the value of that variable shall be a valid representation of a boolean (see 6.5).

7.20.2.4 If the **var2** attribute is present, its value shall be the name of an existing variable, and the value of that variable shall be a valid representation of a boolean (see 6.5).

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

7.20.2.6 Each operand shall be a valid representation of a boolean (see 6.5). The result of the evaluation shall be "**true**" if one or more of the operands are "**true**", otherwise it shall be "**false**".

7.21 Element <xor>

7.21.1 Syntax

7.21.1.1 This element shall have the following attributes:

a) **var1** (optional) – if this attribute is present, its value shall be a valid variable name (see 6.2);

b) **var2** (optional) – if this attribute is present, its value shall be a valid variable name (see 6.2).

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

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

7.21.2 Semantics

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

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

7.21.2.3 If the **`var1`** attribute is present, its value shall be the name of an existing variable, and the value of that variable shall be a valid representation of a boolean (see 6.5).

7.21.2.4 If the **`var2`** attribute is present, its value shall be the name of an existing variable, and the value of that variable shall be a valid representation of a boolean (see 6.5).

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

7.21.2.6 Each operand shall be a valid representation of a boolean (see 6.5). The result of the evaluation shall be "**`true`**" if exactly one of the two operands is "**`true`**", otherwise it shall be "**`false`**".

7.22 Element `<not>`

7.22.1 Syntax

7.22.1.1 This element shall have the following attribute:

- **`var`** (optional) – if this attribute is present, its value shall be a valid variable name (see 6.2).

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

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

7.22.2 Semantics

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

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

7.22.2.3 If the **var** attribute is present, its value shall be the name of an existing variable, and the value of that variable shall be a valid representation of a boolean (see 6.5).

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

7.22.2.5 The operand shall be a valid representation of a boolean (see 6.5). The result of the evaluation shall be "**false**" if the operand is "**true**", and shall be "**true**" otherwise.

7.23 Element *<equal_to>*

7.23.1 Syntax

7.23.1.1 This element shall have the following attributes:

a) **var1** (optional) – if this attribute is present, its value shall be a valid variable name (see 6.2);

b) **var2** (optional) – if this attribute is present, its value shall be a valid variable name (see 6.2);

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

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

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

7.23.1.3 At most one of the attributes **var1** and **value1** shall be present.

7.23.1.4 At most one of the attributes **var2** and **value2** shall be present.

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

7.23.2 Semantics

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

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

7.23.2.3 If the **var1** attribute is present, its value shall be the name of an existing variable, and the value of that variable shall be a valid representation of an integer (see 6.4).

7.23.2.4 If the **var2** attribute is present, its value shall be the name of an existing variable, and the value of that variable shall be a valid representation of an integer (see 6.4).

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

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

7.23.2.7 Each operand shall be a valid representation of an integer (see 6.4). The result of the evaluation shall be "**true**" if the integer represented by the first operand is equal to the integer represented by the second, otherwise it shall be "**false**".

7.24 Element *<not_equal_to>*

7.24.1 Syntax

This element has the same syntax as the element **<equal_to>** (see 7.23).

7.24.2 Semantics

7.24.2.1 This element has the same semantics as the element **<equal_to>**, except for the following.

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

7.25 Element *<greater_than>*

7.25.1 Syntax

This element has the same syntax as the element **<equal_to>** (see 7.23).

7.25.2 Semantics

7.25.2.1 This element has the same semantics as the element **<equal_to>**, except for the following.

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

7.26 Element *<greater_than_or_equal_to>*

7.26.1 Syntax

This element has the same syntax as the element **<equal_to>** (see 7.23).

7.26.2 Semantics

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

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

7.27 Element `<less_than>`

7.27.1 Syntax

This element has the same syntax as the element **`<equal_to>`** (see 7.23).

7.27.2 Semantics

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

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

7.28 Element `<less_than_or_equal_to>`

7.28.1 Syntax

This element has the same syntax as the element **`<equal_to>`** (see 7.23).

7.28.2 Semantics

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

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

7.29 Element `<same_as>`

7.29.1 Syntax

7.29.1.1 This element shall have the following attributes:

a) **`var1`** (optional) – if this attribute is present, its value shall be a valid variable name (see 6.2);

b) **`var2`** (optional) – if this attribute is present, its value shall be a valid variable name (see 6.2);

c) **`value1`** (optional);

d) **`value2`** (optional).

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

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

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

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

7.29.2 Semantics

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

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

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

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

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

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

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

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

7.30 Element `<not_same_as>`

7.30.1 Syntax

This element has the same syntax as the element **`<same_as>`** (see 7.29).

7.30.2 Semantics

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

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

7.31 Element `<existing>`

7.31.1 Syntax

7.31.1.1 This element shall have the following attribute:

- **`var`** (mandatory) – the value of this attribute shall be a valid variable name (see 6.2).

7.31.1.2 The content of this element shall be empty.

7.31.2 Semantics

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

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

7.32 Element `<not_existing>`

7.32.1 Syntax

This element has the same syntax as the element **`<existing>`** (see 7.31).

7.32.2 Semantics

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

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

8 Standard Interface Functions

8.1 General

8.1.1 This clause 8 specifies the use of the standard interface functions of BioAPI in conformance testing. Each major subclause of this clause specifies the following properties of a standard interface function (as applicable):

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

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

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

5) bound activity parameters - the names of input and output parameters of an activity bound to the standard interface function;

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

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

8.1.2 In this clause 8, the term "input value" (for a given input parameter) means:

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

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

8.1.3 In this clause 8, the term "output value" (for a given output parameter) means the value to be assigned to the variable whose name is the value of the **`setvar`** attribute of the corresponding **`<output>`** element (if this element is provided in the invocation).

8.1.4 In this clause 8, the term "return value" means the value to be assigned to the variable whose name is the value of the **`setvar`** attribute of the **`<return>`** element (if this element is provided in the invocation).

8.1.5 For all function invocations, the return value shall be set to the canonical representation of the integer in the range 0 to 4294967295 returned by the underlying function (see 6.4.3).

8.1.6 A standard interface function of the framework callback interface shall be used as follows:

a) the BSP-testing application shall expose the function;

b) the BSP under test may call the function exposed by the BSP-testing application;

c) no other ways of calling the function are supported.

8.1.7 A standard interface function of the BioSPI interface shall be used as follows:

a) the BSP under test may expose the function;

b) the BSP-testing application is allowed to call the function exposed by the BSP under test;

c) no other ways of calling the function are supported.

8.1.8 The BioSPI interface consists of the following standard interface functions:

- **BioSPI_ModuleLoad**
- **BioSPI_ModuleUnload**
- **BioSPI_ModuleAttach**
- **BioSPI_ModuleDetach**
- **BioSPI_FreeBIRHandle**
- **BioSPI_GetBIRFromHandle**
- **BioSPI_GetHeaderFromHandle**
- **BioSPI_EnableEvents**
- **BioSPI_SetGUICallbacks**
- **BioSPI_SetStreamCallback**
- **BioSPI_StreamInputOutput**
- **BioSPI_Capture**
- **BioSPI_CreateTemplate**
- **BioSPI_Process**
- **BioSPI_VerifyMatch**
- **BioSPI_IdentifyMatch**
- **BioSPI_Enroll**
- **BioSPI_Verify**
- **BioSPI_Identify**
- **BioSPI_Import**
- **BioSPI_SetPowerMode**
- **BioSPI_DbOpen**
- **BioSPI_DbClose**
- **BioSPI_DbCreate**
- **BioSPI_DbDelete**
- **BioSPI_DbSetCursor**
- **BioSPI_DbFreeCursor**
- **BioSPI_DbStoreBIR**
- **BioSPI_DbGetBIR**

- **BioSPI_DbGetNextBIR**
- **BioSPI_DbQueryBIR**
- **BioSPI_DbDeleteBIR**

8.1.9 The framework callback interface consists of the following standard interface functions:

- **BioSPI_ModuleEventHandler**
- **BioSPI_GUI_STATE_CALLBACK**
- **BioSPI_GUI_STREAMING_CALLBACK**
- **BioSPI_STREAM_CALLBACK**

NOTE - In BioAPI 1.1, the standard interface functions of the framework callback interface do not have function names, but are specified as function pointer types (which have names). In this Standard, these standard interface functions are given names, both for ease of reference in the clauses of this Standard, and for use in the assertion language.

8.2 Parameter Groups

The following subclauses specify groups of parameters (of functions or activities). Each group is referenced either by one or more subsequent groups or by one or more subclauses of this clause 8 relative to standard interface functions.

8.2.1 Parameter Group "Factors"

8.2.1.1 The parameter group "Factors" consists of the following parameters:

- **FactorMultiple**
- **FactorFacialFeatures**
- **FactorVoice**
- **FactorFingerprint**
- **FactorIris**
- **FactorRetina**
- **FactorHandGeometry**
- **FactorSignatureDynamics**
- **FactorKeystrokeDynamics**
- **FactorLipMovement**
- **FactorThermalFaceImage**
- **FactorThermalHandImage**
- **FactorGait**
- **FactorPassword**

8.2.1.2 This parameter group represents a value of the native type **BioAPI_BIR_AUTH_FACTORS** (which is an integer type).

8.2.1.3 An input value of these parameters shall be either a valid representation of a boolean (see 6.5) or an empty string. An output value of these parameters shall be the canonical representation of a boolean (see 6.5.2).

8.2.1.4 The value (say, V) of the native type **BioAPI_BIR_AUTH_FACTORS** represented by this parameter group shall be determined from the values of its members as specified in the two following subclauses.

8.2.1.4.1 The value V shall be initially set to zero.

8.2.1.4.2 For each member of the parameter group whose value is "**true**", an integer shall be added to V according to the following table.

FactorMultiple	$1=2^0$
FactorFacialFeatures	$2=2^1$
FactorVoice	$4=2^2$
FactorFingerprint	$8=2^3$
FactorIris	$16=2^4$
FactorRetina	$32=2^5$
FactorHandGeometry	$64=2^6$
FactorSignatureDynamics	$128=2^7$
FactorKeystrokeDynamics	$256=2^8$
FactorLipMovement	$512=2^9$
FactorThermalFaceImage	$1024=2^{10}$
FactorThermalHandImage	$2048=2^{11}$
FactorGait	$4096=2^{12}$
FactorPassword	$2147483648=2^{31}$

8.2.1.5 Given an integer (say, V) that is a value of the native type **BioAPI_BIR_AUTH_FACTORS**, the values of the members of the parameter group that canonically represent V shall be determined as specified in the two following subclauses.

8.2.1.5.1 The integer V shall be decomposed into a sum of powers of two, each occurring at most once.

8.2.1.5.2 For each power of two listed in the table above that occurs in the decomposition of V, the value of the parameter in the corresponding row of to the table shall be "**true**". The value of all the remaining parameters (if any) shall be "**false**".

8.2.2 Parameter Group "Events"

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

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

8.2.2.2 This parameter group represents a value of the native type **BioAPI_MODULE_EVENT_MASK** (which is an integer type).

8.2.2.3 An input value of each of these parameters shall be either a valid representation of a boolean (see 6.5) or an empty string. An output value shall be the canonical representation of a boolean (see 6.5.2).

8.2.2.4 The value (say, V) of the native type **BioAPI_MODULE_EVENT_MASK** represented by this parameter group shall be determined from the values of its members as specified in the two following subclauses.

8.2.2.4.1 The value V shall be initially set to zero.

8.2.2.4.2 For each member of the parameter group whose value is "**true**", an integer shall be added to V according to the following table.

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

8.2.3 Parameter Group "Biometric type"

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

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

8.2.3.2 This parameter group represents a value of the native type **BioAPI_BIR_DATA_TYPE** (which is an integer type).

8.2.3.3 An input value of **ProcessedLevel** shall be either a valid representation of an integer (see 6.4) in the range 0 to 15, or an empty string. An output value shall be the canonical representation of an integer (see 6.4.3) in the same range.

8.2.3.4 An input value of **Encrypted** and **Signed** shall be either a valid representation of a boolean (see 6.5) or an empty string. An output value shall be the canonical representation of a boolean (see 6.5.2).

8.2.3.5 The value (say, V) of the native type **BioAPI_BIR_DATA_TYPE** represented by this parameter group shall be determined from the values of its members as specified in the two following subclauses.

8.2.3.5.1 The value V shall be initially set to the integer represented by the input value of **ProcessedLevel** (or zero, if the input value is an empty string).

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

Encrypted	$16=2^4$
Signed	$32=2^5$

8.2.3.6 Given an integer (say, V) that is a value of the native type **BioAPI_BIR_DATA_TYPE**, the values of the members of the parameter group that canonically represent V shall be determined as specified in the two following subclauses.

8.2.3.6.1 The integer V shall be decomposed into a sum of:

a) an integer less than 16;

b) the value 16; and

c) the value 32,

each occurring at most once.

8.2.3.6.2 For each power of two listed in the table above (16 and 32) that occurs in the decomposition of V, the value of the parameter in the corresponding row of the table shall be "**true**". The value of **ProcessedLevel** shall be the canonical representation of the integer in a) above. The value of all remaining parameters (if any) shall be "**false**".

8.2.4 Parameter Group "BIR header"

8.2.4.1 The parameter group "BIR header" consists of the following parameters:

- **Length**
- **HeaderVersion**
- the members of the parameter group "Biometric type" (see 8.2.3)
- **FormatOwner**
- **FormatID**
- **Quality**
- **Purpose**
- the members of the parameter group "Factors" (see 9.2.1).

8.2.4.2 This parameter group represents a value of the native type **BioAPI_BIR_HEADER**.

8.2.4.3 An input value of **Length** shall be either a valid representation of an integer (see 6.4) in the range 0 to 4294967295, or an empty string. An output value shall be the canonical representation of an integer (see 6.4.3) in the same range.

8.2.4.4 An input value of **HeaderVersion** and **Purpose** shall be either a valid representation of an integer in the range 0 to 255, or an empty string. An output value shall be the canonical representation of an integer in the same range.

8.2.4.5 An input value of **FormatOwner** and **FormatID** shall be either a valid representation of an integer in the range 0 to 65535, or an empty string. An output value shall be the canonical representation of an integer in the same range.

8.2.4.6 An input value of **Quality** shall be either a valid representation of an integer in the range -128 to 127, or an empty string. An output value shall be the canonical representation of an integer in the same range.

8.2.4.7 The value (say, V) of the native type **BioAPI_BIR_HEADER** represented by this parameter group shall be determined from the values of its members as specified in the four following subclauses.

8.2.4.7.1 The integer represented by the value of **Length**, **HeaderVersion**, **Quality**, and **Purpose** (or zero if the value is an empty string) shall be written to the field of V with the same name.

8.2.4.7.2 The value of type **BioAPI_BIR_DATA_TYPE** represented (see 8.2.3.5) by the value of the parameter group "Biometric type" shall be written to the field **Type** of V.

8.2.4.7.3 The integer represented by the value of **FormatOwner** and **FormatID** (or zero if the value is an empty string) shall be written to the field **FormatOwner** or **FormatID** (respectively) of the field **Format** of V.

8.2.4.7.4 The value of type **BioAPI_BIR_AUTH_FACTORS** represented (see 8.2.1.4) by the value of the parameter group "Factors" shall be written to the field **FactorsMask** of V.

8.2.4.8 Given a value (say, V) of the native type **BioAPI_BIR_HEADER**, the values of the members of the parameter group that canonically represent V shall be determined as specified in the four following subclauses.

8.2.4.8.1 The value of **Length**, **HeaderVersion**, **Quality**, and **Purpose** shall be the canonical representation of the integer in the field of V with the same name.

8.2.4.8.2 The value of the parameter group "Biometric type" shall be the canonical representation (see 8.2.3.4) of the value of type **BioAPI_BIR_DATA_TYPE** in the field **Type** of V.

8.2.4.8.3 The value of **FormatOwner** and **FormatID** shall be the canonical representation of the integer in the field **FormatOwner** or **FormatID** (respectively) of the field **Format** of V.

8.2.4.8.4 The value of the parameter group "Factors" shall be the canonical representation (see 8.2.1.5) of the value of type **BioAPI_BIR_AUTH_FACTORS** in the field **FactorsMask** of V.

8.2.5 Parameter Group "BIR"

8.2.5.1 The parameter group "BIR" consists of the following parameters:

- the members of the parameter group "BIR header" (see 9.2.4)
- **BiometricData**
- **Signature**

8.2.5.2 This parameter group represents a value of the native type **BioAPI_BIR**.

8.2.5.3 An input value of **BiometricData** and **Signature** shall be a valid representation of a binary data block (see 7.7). An output value shall be the canonical representation of a binary data block (see 6.7.2).

8.2.5.4 The value (say, V) of the native type **BioAPI_BIR** represented by this parameter group shall be determined from the values of its members as specified in the three following subclauses.

8.2.5.4.1 The value of type **BioAPI_BIR_HEADER** represented (see 8.2.4.7) by the value of the parameter group "BIR header" shall be written to the field **Header** of V.

8.2.5.4.2 The binary data block represented by the value of **BiometricData** shall be written to a memory block of sufficient size, whose address shall be written to the field **BiometricData** of V.

8.2.5.4.3 The binary data block represented by the value of **Signature** shall be written to a memory block of sufficient size. The address of that memory block shall be written to the field **Data** of a variable of type **BioAPI_DATA**, and the length (in octets) of the binary data block shall be written to the field **Length** of that variable. The address of that variable shall be written to the field **Signature** of V.

8.2.5.5 Given a value (say, V) of the native type **BioAPI_BIR**, the values of the members of the parameter group that canonically represent V shall be determined as specified in the three following subclauses.

8.2.5.5.1 The value of the parameter group "BIR header" shall be the canonical representation (see 8.2.4.8) of the value of type **BioAPI_BIR_HEADER** in the field **Header** of V.

8.2.5.5.2 The value of **BiometricData** shall be the canonical representation of the binary data block in the memory block whose address is in the field **BiometricData** of V, and whose length is the difference between the value of the field **Length** of the field **Header** of V and the size (in octets) of a variable of type **BioAPI_BIR_HEADER**.

8.2.5.5.3 The value of **Signature** shall be the canonical representation of the binary data block in the memory block whose address is in the field **Data** of the variable of type **BioAPI_DATA** pointed to by the field **Signature** of V, and whose length is in the field **Length** of that variable.

8.2.6 Parameter Group "Input BIR"

8.2.6.1 The parameter group "Input BIR" consists of the following parameters:

- **Form**
- **DbHandle**
- **KeyValue**
- **BIRHandle**
- the members of the parameter group "BIR" (see 9.2.5).

8.2.6.2 This parameter group represents a value of the native type **BioAPI_INPUT_BIR**.

8.2.6.3 An input value of **Form** shall be either a valid representation of an integer (see 6.4) in the range 0 to 255, or an empty string. An output value shall be the canonical representation of an integer (see 6.4.3) in the same range.

8.2.6.4 An input value of **DbHandle** and **BIRHandle** shall be either a valid representation of an integer in the range -2147483648 to 2147483647, or an empty string. An output value shall be the canonical representation of an integer in the same range.

8.2.6.5 An input value of **KeyValue** shall be either a valid representation of a UUID (see 6.6), or an empty string. An output value shall be the canonical representation of a UUID (see 6.6.3).

8.2.6.6 The value (say, V) of the native type **BioAPI_INPUT_BIR** represented by this parameter group shall be determined from the values of its members as specified in the five following subclauses.

8.2.6.6.1 The integer (say, F) represented by the value of **Form** (or zero if the value is an empty string) shall be written to the field **Form** of V.

8.2.6.6.2 If F is 1, then:

a) the integer represented by the value of **DbHandle** (or zero if the value is an empty string) shall be written to the field **DbHandle** of a variable of type **BioAPI_DBBIR_ID**;

b) the UUID represented by the value of **KeyValue** (or the UUID "**00000000-0000-0000-0000-000000000000**" if the value is an empty string) shall be written to the field **KeyValue** of that variable; and

c) the address of that variable shall be written to the field **BIRInDb** of the field **InputBIR** of V.

8.2.6.6.3 If F is 2 then:

a) the integer represented by the value of **BIRHandle** (or zero if the value is an empty string) shall be written to a variable of type **BioAPI_BIR_HANDLE**; and

b) the address of that variable shall be written to the field **BIRInBSP** of the field **InputBIR** of V.

8.2.6.6.4 If F is 3, then:

a) the value of type **BioAPI_BIR** represented (see 8.2.5.4) by the value of the parameter group "BIR" shall be written to a variable of type **BioAPI_BIR**; and

b) the address of that variable shall be written to the field **BIR** of the field **InputBIR** of V.

8.2.6.6.5 If F is less than 1 or greater than 3, the field **BIR** of the field **InputBIR** of V shall be set to NULL.

8.2.7 Parameter Group "Identify population"

8.2.7.1 The parameter group "Identify population" consists of the following parameters:

- **Type**
- **BIRDataBase**
- **NumberOfMembers**
- the members of the parameter group "BIR" (see 9.2.5), with their names prefixed by "**BIR_1_**"
- the members of the parameter group "BIR" (see 9.2.5), with their names prefixed

by "**BIR_2_**"

- the members of the parameter group "BIR" (see 9.2.5), with their names prefixed by "**BIR_3_**"
- the members of the parameter group "BIR" (see 9.2.5), with their names prefixed by "**BIR_4_**"

8.2.7.2 This parameter group represents a value of the native type **BioAPI_IDENTIFY_POPULATION**.

8.2.7.3 An input value of **Type** shall be either a valid representation of an integer (see 6.4) in the range 0 to 255, or an empty string. An output value shall be the canonical representation of an integer (see 6.4.3) in the same range.

8.2.7.4 An input value of **BIRDataBase** shall be either a valid representation of an integer in the range -2147483648 to 2147483647, or an empty string. An output value shall be the canonical representation of an integer in the same range.

8.2.7.5 An input value of **NumberOfMembers** shall be either a valid representation of an integer in the range 0 to 4, or an empty string. An output value of **NumberOfMembers** shall be the canonical representation of an integer in the range 0 to 4294967295.

8.2.7.6 The value (say, V) of the native type **BioAPI_IDENTIFY_POPULATION** represented by this parameter group shall be determined from the values of its members as specified in the four following subclauses.

8.2.7.6.1 The integer (say, T) represented by the value of **Type** (or zero if the value is an empty string) shall be written to the field **Type** of V.

8.2.7.6.2 If T is 1, then:

a) the integer represented by the value of **BIRDataBase** (or zero if the value is an empty string) shall be written to a variable of type **BioAPI_DB_HANDLE**; and

b) the address of that variable shall be written to the field **BIRDataBase** of the field **BIRs** of V.

8.2.7.6.3 If T is 2, then:

a) the integer (say, N) represented by the value of **NumberOfMembers** (or zero if the value is an empty string) shall be written to the field **NumberOfMembers** of a variable of type **BioAPI_BIR_ARRAY_POPULATION**;

b) for the first N instances of the parameter group "BIR", the value of type **BioAPI_BIR** represented by the value of each instance shall be written to a variable of type **BioAPI_BIR**;

c) the addresses of all the variables of type **BioAPI_BIR** assigned in b) shall be written, in order, to an array of pointers;

d) the address of that array of pointers shall be written to the field **Members** of the variable of type **BioAPI_BIR_ARRAY_POPULATION** referenced in a);

e) the address of that variable shall be written to the field **BIRArray** of the field

BIRs of V.

8.2.7.6.4 If T is less than 1 or greater than 2, then the field **BIRArray** of the field **BIRs** of V shall be set to NULL.

8.2.8 Parameter Group "Candidate"

8.2.8.1 The parameter group "Candidate" consists of the following parameters:

- **Type**
- **BIRInDataBase**
- **BIRInArray**
- **FARAchieved**
- **FRRAchieved**

8.2.8.2 This parameter group represents a value of the native type **BioAPI_CANDIDATE**.

8.2.8.3 An input value of **Type** shall be either a valid representation of an integer (see 6.4) in the range 0 to 255, or an empty string. An output value shall be the canonical representation of an integer (see 6.4.3) in the same range.

8.2.8.4 An input value of **BIRInDataBase** shall be either a valid representation of a UUID (see 6.6), or an empty string. An output value shall be the canonical representation of a UUID (see 6.6.3).

8.2.8.5 An input value of **BIRInArray** shall be either a valid representation of an integer in the range 0 to 4294967295, or an empty string. An output value shall be the canonical representation of an integer in the same range.

8.2.8.6 An input value of **FARAchieved** and **FRRAchieved** shall be either a valid representation of an integer in the range -2147483648 to 2147483647, or an empty string. An output value shall be the canonical representation of an integer in the same range.

8.2.8.7 The value (say, V) of the native type **BioAPI_CANDIDATE** represented by this parameter group shall be determined from the values of its members as specified in the five following subclauses.

8.2.8.7.1 The integer (say, T) represented by the value of **Type** (or zero if the value is an empty string) shall be written to the field **Type** of V.

8.2.8.7.2 If T is 1, then the UUID represented by the value of **BIRInDataBase** (or the UUID "**00000000-0000-0000-0000-000000000000**" if the value is an empty string) shall be written to a variable of type **BioAPI_UUID**, and the address of that variable shall be written to the field **BIRInDataBase** of the field **BIR** of V.

8.2.8.7.3 If T is 2, then the integer represented by the value of **BIRInArray** (or zero if the value is an empty string) shall be written to a variable of type **uint32**, and the address of that variable shall be written to the field **BIRInArray** of the field **BIR** of V.

8.2.8.7.4 If T is less than 1 or greater than 2, then the field **BIRInArray** of the field **BIR** of V shall be set to NULL.

8.2.8.7.5 The integer represented by the value of **FARAchieved** and **FRRAchieved** (or zero if the value is an empty string) shall be written to the field of V with the same name.

8.2.8.8 Given a value (say, V) of the native type **BioAPI_CANDIDATE**, the values of the members of the parameter group that canonically represent V shall be determined as specified in the four following subclauses.

8.2.8.8.1 The value of **Type** shall be the canonical representation of the integer (say, T) in the field of V with the same name.

8.2.8.8.2 If T is 1, then the value of **BIRInDataBase** shall be the canonical representation of the UUID in the variable of type **BioAPI_UUID** pointed to by the field **BIRInDataBase** of V. Otherwise, the value of **BIRInDataBase** shall be an empty string.

8.2.8.8.3 If T is 2, then the value of **BIRInArray** shall be the canonical representation of the integer in the variable of type **uint32** pointed to by the field **BIRInArray** of V. Otherwise, the value of **BIRInArray** shall be an empty string.

8.2.8.8.4 The value of **FARAchieved** and **FRRAchieved** shall be the canonical representation of the integer (say, T) in the field of V with the same name.

8.2.9 Parameter Group "GUI state"

8.2.9.1 The parameter group "GUI state" consists of the following parameters:

- **SampleAvailable**
- **MessageProvided**
- **ProgressProvided**

8.2.9.2 This parameter group represents a value of the native type **BioAPI_GUI_STATE** (which is an integer type).

8.2.9.3 An input value of these parameters shall be either a valid representation of a boolean (see 6.5) or an empty string. An output value of these parameters shall be the canonical representation of a boolean (see 6.5.2).

8.2.9.4 The value (say, V) of the native type **BioAPI_GUI_STATE** represented by this parameter group shall be determined from the values of its members as specified in the two following subclauses.

8.2.9.4.1 The value V shall be initially set to zero.

8.2.9.4.2 For each member of the parameter group whose value is "**true**", an integer shall be added to V according to the following table.

SampleAvailable	$1=2^0$
MessageProvided	$2=2^1$
ProgressProvided	$4=2^2$

8.2.9.5 Given an integer (say, V) that is a value of the native type **BioAPI_BIR_AUTH_FACTORS**, the values of the members of the parameter group that canonically represent V shall be determined as specified in the two following subclauses.

8.2.9.5.1 The integer V shall be decomposed into a sum of powers of two, each occurring at most once.

8.2.9.5.2 For each power of two that occurs in the decomposition of V, the value of the corresponding parameter (according to the table above) shall be "**true**". The value of all the remaining parameters (if any) shall be "**false**".

8.3 BioSPI_ModuleLoad

8.3.1 General

This function belongs to the BioSPI interface, and has the following native prototype:

```
BioAPI_RETURN BioAPI
BioSPI_ModuleLoad(
   const void *Reserved,
   const BioAPI_UUID *BSPUuid,
   BioSPI_ModuleEventHandler BioAPINotifyCallback,
   void* BioAPINotifyCallbackCtx);
```

8.3.2 Function Invocation Scheme

This function has the following input parameters:

- **Reserved**
- **BSPUuid**
- **BioAPINotifyCallback**
- **BioAPINotifyCallbackCtx**

and no output parameters.

8.3.3 Function Invocation Input

8.3.3.1 The input value of **Reserved** shall be a valid representation of an integer in the range 0 to 4294967295 (see 6.4).

8.3.3.2 The input value of **BSPUuid** shall be a valid representation of a UUID (see 6.6).

8.3.3.3 The input value of **BioAPINotifyCallback** and **BioAPINotifyCallbackCtx** shall be either "**0**" or "*****".

8.3.3.4 The integer represented by the input value of **Reserved** shall be assigned to the native parameter with the same name.

8.3.3.5 The UUID represented by the input value of **BSPUuid** shall be written to a variable of type **BioAPI_UUID**, whose address shall be assigned to the native parameter **BSPUuid**.

8.3.3.6 If the input value of **AppNotifyCallback** is "**0**", a NULL pointer value shall be assigned to the native parameter **AppNotifyCallback**. If it is "*****", a non-NULL pointer value,

which is the address of a native function (within the testing component) implementing the standard interface function **BioAPI_EventHandler**, shall be assigned to the native parameter **AppNotifyCallback**.

> **NOTE -** If the input value of **AppNotifyCallback** is not "**0**", any subsequent incoming calls to the standard interface function **BioAPI_EventHandler** will result in an invocation of the activity bound to this function, if such a binding exists.

8.3.3.7 If the input value of **AppNotifyCallbackCtx** is "**0**", a NULL pointer value shall be assigned to the native parameter **AppNotifyCallbackCtx**. If it is "*****", a non-NULL pointer value, which is the address of a variable of type **void*** set to NULL, shall be assigned to the native parameter **AppNotifyCallbackCtx**.

8.3.4 Function Invocation Output

There are no output parameters.

8.4 BioSPI_ModuleUnload

8.4.1 General

This function belongs to the BioSPI interface, and has the following native prototype:

```
BioAPI_RETURN BioAPI
BioSPI_ModuleUnload(
   const void *Reserved,
   const BioAPI_UUID *BSPUuid,
   BioSPI_ModuleEventHandler BioAPINotifyCallback,
   void* BioAPINotifyCallbackCtx);
```

8.4.2 Function Invocation Scheme

This function has the following input parameters:

- **Reserved**
- **BSPUuid**
- **BioAPINotifyCallback**
- **BioAPINotifyCallbackCtx**

and no output parameters.

8.4.3 Function Invocation Input

8.4.3.1 The input value of **Reserved** shall be a valid representation of an integer in the range 0 to 4294967295 (see 6.4).

8.4.3.2 The input value of **BSPUuid** shall be a valid representation of a UUID (see 6.6).

8.4.3.3 The input value of **BioAPINotifyCallback** and **BioAPINotifyCallbackCtx** shall be either "**0**" or "*****".

8.4.3.4 The integer represented by the input value of **Reserved** shall be assigned to the native parameter with the same name.

8.4.3.5 The UUID represented by the input value of **BSPUuid** shall be written to a variable of type **BioAPI_UUID**, whose address shall be assigned to the native parameter **BSPUuid**.

8.4.3.6 If the input value of **BioAPINotifyCallback** is "**0**", a NULL pointer value shall be assigned to the native parameter **BioAPINotifyCallback**. If it is "*****", a non-NULL pointer value, which is the address of a native function (within the testing component) implementing the standard interface function **BioAPI_EventHandler**, shall be assigned to the native parameter **BioAPINotifyCallback**.

8.4.3.7 If the input value of **BioAPINotifyCallbackCtx** is "**0**", a NULL pointer value shall be assigned to the native parameter **BioAPINotifyCallbackCtx**. If it is "*****", a non-NULL pointer value, which is the address of a variable of type **void*** set to NULL, shall be assigned to the native parameter **BioAPINotifyCallbackCtx**.

8.4.4 Function Invocation Output

There are no output parameters.

8.5 BioSPI_ModuleAttach

8.5.1 General

This function belongs to the BioSPI interface, and has the following native prototype:

```
BioAPI_RETURN BioAPI
BioSPI_ModuleAttach(
   const BioAPI_UUID *BSPUuid,
   const BioAPI_VERSION *Version,
   BioAPI_DEVICE_ID DeviceID,
   uint32 Reserved1,
   uint32 Reserved2,
   BioAPI_HANDLE ModuleHandle,
   uint32 Reserved3,
   const void *Reserved4,
   const void *Reserved5,
   const void *Reserved6,
   const BioAPI_UPCALLS *Upcalls,
   BioAPI_MODULE_FUNCS_PTR *FuncTbl);
```

8.5.2 Function Invocation Scheme

This function has the following input parameters:

- **BSPUuid**
- **VersionMajor**
- **VersionMinor**
- **DeviceID**
- **Reserved1**
- **Reserved2**
- **ModuleHandle**
- **Reserved3**
- **Reserved4**

- **Reserved5**
- **Reserved6**

and no output parameters.

8.5.3 Function Invocation Input

8.5.3.1 The input value of **BSPUuid** shall be a valid representation of a UUID (see 6.6).

8.5.3.2 The input value of **VersionMajor**, **VersionMinor**, **DeviceID**, **Reserved1**, **Reserved2**, **ModuleHandle**, **Reserved3**, **Reserved4**, **Reserved5**, and **Reserved6** shall be a valid representation of an integer (see 6.4) in the range 0 to 4294967295.

8.5.3.3 The UUID represented by the input value of **BSPUuid** shall be written to a variable of type **BioAPI_UUID**, whose address shall be assigned to the native parameter **BSPUuid**.

8.5.3.4 The integer represented by the input value of **VersionMajor** shall be written to the field **Major** of a variable of type **BioAPI_VERSION**, and the integer represented by the input value of **VersionMinor** shall be written to the field **Minor** of that variable. The address of that variable shall be assigned to the input parameter **Version**.

8.5.3.5 The integer represented by the input value of **DeviceID**, **Reserved1**, **Reserved2**, **ModuleHandle**, **Reserved3**, **Reserved4**, **Reserved5**, and **Reserved6** shall be assigned to the native parameter with the same name.

8.5.3.6 The setting of the native parameters **Upcalls** and **FuncTbl** is not specified.

8.5.4 Function Invocation Output

There are no output parameters.

8.6 BioSPI_ModuleDetach

8.6.1 General

This function belongs to the BioSPI interface, and has the following native prototype:

```
BioAPI_RETURN BioAPI
BioSPI_ModuleDetach(
   BioAPI_HANDLE ModuleHandle);
```

8.6.2 Function Invocation Scheme

This function has the following input parameter:

- **ModuleHandle**

and no output parameters.

8.6.3 Function Invocation Input

8.6.3.1 The input value of **ModuleHandle** shall be a valid representation of an integer (see 6.4) in the range 0 to 4294967295.

8.6.3.2 The integer represented by the input value of **`ModuleHandle`** shall be assigned to the native parameter with the same name.

8.6.4 Function Invocation Output

There are no output parameters.

8.7 BioSPI_FreeBIRHandle

8.7.1 General

8.7.1.1 This function belongs to the BioSPI interface, and has the following native prototype:

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

8.7.2 Function Invocation Scheme

This function has the following input parameters:

- **`ModuleHandle`**
- **`Handle`**

and no output parameters.

8.7.3 Function Invocation Input

8.7.3.1 The input value of **`ModuleHandle`** shall be a valid representation of an integer (see 6.4) in the range 0 to 4294967295.

8.7.3.2 The input value of **`Handle`** shall be a valid representation of an integer in the range -2147483648 to 2147483647.

8.7.3.3 The integer represented by the input value of **`ModuleHandle`** and **`Handle`** shall be assigned to the native parameter with the same name.

8.7.4 Function Invocation Output

There are no output parameters.

8.8 BioSPI_GetBIRFromHandle

8.8.1 General

This function belongs to the BioSPI interface, and has the following native prototype:

```
BioAPI_RETURN BioAPI
BioSPI_GetBIRFromHandle(
   BioAPI_HANDLE ModuleHandle,
   BioAPI_BIR_HANDLE Handle,
   BioAPI_BIR_PTR *BIR);
```

8.8.2 Function Invocation Scheme

This function has the following input parameters:

- **ModuleHandle**
- **Handle**
- **no_BIR**

and the following output parameters:

- the members of the parameter group "BIR" (see 9.2.5)

8.8.3 Function Invocation Input

8.8.3.1 The input value of **ModuleHandle** shall be a valid representation of an integer (see 6.4) in the range 0 to 4294967295.

8.8.3.2 The input value of **Handle** shall be a valid representation of an integer in the range -2147483648 to 2147483647.

8.8.3.3 The input value of **no_BIR** shall be either a valid representation of a boolean (see 6.5) or an empty string.

8.8.3.4 The integer represented by the input value of **ModuleHandle** and **Handle** shall be assigned to the native parameter with the same name.

8.8.3.5 If the input value of **no_BIR** is "**true**", then the native parameter **BIR** shall be set to NULL, otherwise it shall be set to the address of a variable of type **BioAPI_BIR**.

8.8.4 Function Invocation Output

8.8.4.1 If the native parameter **BIR** is not NULL, then the output value of the parameter group "BIR" shall be the canonical representation (see 8.2.5.5) of the value of type **BioAPI_BIR** pointed to by the native parameter **BIR**.

8.8.4.2 If the native parameter **BIR** is NULL, then the output values of all members of the parameter group "BIR" shall be empty strings.

8.9 BioSPI_GetHeaderFromHandle

8.9.1 General

This function belongs to the BioSPI interface, and has the following native prototype:

```
BioAPI_RETURN BioAPI
BioSPI_GetHeaderFromHandle(
   BioAPI_HANDLE ModuleHandle,
   BioAPI_BIR_HANDLE Handle,
   BioAPI_BIR_HEADER_PTR Header);
```

8.9.2 Function Invocation Scheme

This function has the following input parameters:

- **ModuleHandle**
- **Handle**
- **no_Header**

and the following output parameters:

- the members of the parameter group "BIR header" (see 9.2.4)

8.9.3 Function Invocation Input

8.9.3.1 The input value of **`ModuleHandle`** shall be a valid representation of an integer (see 6.4) in the range 0 to 4294967295.

8.9.3.2 The input value of **`Handle`** shall be a valid representation of an integer in the range -2147483648 to 2147483647.

8.9.3.3 The input value of **`no_Header`** shall be either a valid representation of a boolean (see 6.5) or an empty string.

8.9.3.4 The integer represented by the input value of **`ModuleHandle`** and **`Handle`** shall be assigned to the native parameter with the same name.

8.9.3.5 If the input value of **`no_Header`** is "**`true`**", then the native parameter **`Header`** shall be set to NULL, otherwise it shall be set to the address of a variable of type **`BioAPI_BIR_HEADER`**.

8.9.4 Function Invocation Output

8.9.4.1 If the native parameter **`Header`** is not NULL, then the output value of the parameter group "BIR header" shall be the canonical representation (see 8.2.4.8) of the value of type **`BioAPI_BIR_HEADER`** pointed to by the native parameter **`Header`**.

8.9.4.2 If the native parameter **`Header`** is NULL, then the output values of all members of the parameter group "BIR header" shall be empty strings.

8.10 BioSPI_EnableEvents

8.10.1 General

This function belongs to the BioSPI interface, and has the following native prototype:

```
BioAPI_RETURN BioAPI
BioSPI_EnableEvents(
   BioAPI_HANDLE ModuleHandle,
   BioAPI_MODULE_EVENT_MASK *Events);
```

8.10.2 Function Invocation Scheme

This function has the following input parameters:

- **`ModuleHandle`**
- the members of the parameter group "Events" (see 9.2.2)

and no output parameters.

8.10.3 Function Invocation Input

8.10.3.1 The input value of **`ModuleHandle`** shall be a valid representation of an integer (see 6.4) in the range 0 to 4294967295.

8.10.3.2 The integer represented by the input value of **`ModuleHandle`** shall be assigned to the native parameter with the same name.

8.10.3.3 The value of type **BioAPI_MODULE_EVENT_MASK** represented (see 8.2.2.4) by the input value of the parameter group "Events" shall be written to a variable of type **BioAPI_MODULE_EVENT_MASK**, whose address shall be assigned to the native parameter **Events**.

8.10.4 Function Invocation Output

There are no output parameters.

8.11 BioSPI_SetGUICallbacks

8.11.1 General

This function belongs to the BioSPI interface, and has the following native prototype:

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

8.11.2 Function Invocation Scheme

This function has the following input parameters:

- **ModuleHandle**
- **GuiStreamingCallback**
- **GuiStreamingCallbackCtx**
- **GuiStateCallback**
- **GuiStateCallbackCtx**

and no output parameters.

8.11.3 Function Invocation Input

8.11.3.1 The input value of **ModuleHandle** shall be a valid representation of an integer (see 6.4) in the range 0 to 4294967295.

8.11.3.2 The input value of **GuiStreamingCallback**, **GuiStreamingCallbackCtx**, **GuiStateCallback**, and **GuiStateCallbackCtx** shall be either "**0**" or "*****".

8.11.3.3 The integer represented by the input value of **ModuleHandle** shall be assigned to the native parameter with the same name.

8.11.3.4 If the input value of **GuiStreamingCallback** is "**0**", a NULL pointer value shall be assigned to the native parameter **GuiStreamingCallback**. If it is "*****", a non-NULL pointer value, which is the address of a native function (within the testing component) implementing the standard interface function **BioSPI_GUI_STREAMING_CALLBACK**, shall be assigned to the native parameter **GuiStreamingCallback**.

> **NOTE -** If the input value of **GuiStreamingCallback** is not "**0**", any subsequent incoming calls to the standard interface function **BioSPI_GUI_STREAMING_CALLBACK** will result in an invocation of the

activity bound to this function, if such a binding exists.

8.11.3.5 If the input value of **GuiStreamingCallbackCtx** is "**0**", a NULL pointer value shall be assigned to the native parameter **GuiStreamingCallbackCtx**. If it is "*****", a non-NULL pointer value, which is the address of a variable of type **void*** set to NULL, shall be assigned to the native parameter **GuiStreamingCallbackCtx**.

8.11.3.6 If the input value of **GuiStateCallback** is "**0**", a NULL pointer value shall be assigned to the native parameter **GuiStateCallback**. If it is "*****", a non-NULL pointer value, which is the address of a native function (within the testing component) implementing the standard interface function **BioSPI_GUI_STATE_CALLBACK**, shall be assigned to the native parameter **GuiStateCallback**.

NOTE - If the input value of **GuiStateCallback** is not "**0**", any subsequent incoming calls to the standard interface function **BioSPI_GUI_STATE_CALLBACK** will result in an invocation of the activity bound to this function, if such a binding exists.

8.11.3.7 If the input value of **GuiStateCallbackCtx** is "**0**", a NULL pointer value shall be assigned to the native parameter **GuiStateCallbackCtx**. If it is "*****", a non-NULL pointer value, which is the address of a variable of type **void*** set to NULL, shall be assigned to the native parameter **GuiStateCallbackCtx**.

8.11.4 Function Invocation Output

There are no output parameters.

8.12 BioSPI_SetStreamCallback

8.12.1 General

This function belongs to the BioSPI interface, and has the following native prototype:

```
BioAPI_RETURN BioAPI
BioAPI_SetStreamCallback(
   BioAPI_HANDLE ModuleHandle,
   BioAPI_STREAM_CALLBACK StreamCallback,
   void *StreamCallbackCtx);
```

8.12.2 Function Invocation Scheme

This function has the following input parameters:

- **ModuleHandle**
- **StreamCallback**
- **StreamCallbackCtx**

and no output parameters.

8.12.3 Function Invocation Input

8.12.3.1 The input value of **ModuleHandle** shall be a valid representation of an integer (see 6.4) in the range 0 to 4294967295.

8.12.3.2 The input value of **StreamCallback** and **StreamCallbackCtx** shall be either "**0**" or "*****".

8.12.3.3 The integer represented by the input value of **ModuleHandle** shall be assigned to the native parameter with the same name.

8.12.3.4 If the input value of **StreamCallback** is "**0**", a NULL pointer value shall be assigned to the native parameter **StreamCallback**. If it is "*****", a non-NULL pointer value, which is the address of a native function (within the testing component) implementing the standard interface function **BioSPI_STREAM_CALLBACK**, shall be assigned to the native parameter **StreamCallback**.

> **NOTE -** If the input value of **StreamCallback** is not "**0**", any subsequent incoming calls to the standard interface function **BioSPI_STREAM_CALLBACK** will result in an invocation of the activity bound to this function, if such a binding exists.

8.12.3.5 If the input value of **StreamCallbackCtx** is "**0**", a NULL pointer value shall be assigned to the native parameter **StreamCallbackCtx**. If it is "*****", a non-NULL pointer value, which is the address of a variable of type **void*** set to NULL, shall be assigned to the native parameter **StreamCallbackCtx**.

8.12.4 Function Invocation Output

There are no output parameters.

8.13 BioSPI_StreamInputOutput

8.13.1 General

This function belongs to the BioSPI interface, and has the following native prototype:

```
BioAPI_RETURN BioAPI
BioAPI_StreamInputOutput(
   BioAPI_HANDLE ModuleHandle,
   BioAPI_DATA_PTR InMessage,
   BioAPI_DATA_PTR OutMessage);
```

8.13.2 Function Invocation Scheme

This function has the following input parameters:

- **ModuleHandle**
- **InMessage**
- **no_OutMessage**

and the following output parameters:

- **OutMessage**

8.13.3 Function Invocation Input

8.13.3.1 The input value of **ModuleHandle** shall be a valid representation of an integer (see 6.4) in the range 0 to 4294967295.

8.13.3.2 The input value of **InMessage** shall be a valid representation of a binary data block (see 7.7).

8.13.3.3 The input value of **no_OutMessage** shall be either a valid representation of a boolean (see 6.5) or an empty string.

8.13.3.4 The integer represented by the input value of **ModuleHandle** shall be assigned to the native parameter with the same name.

8.13.3.5 The binary data block represented by the input value of **InMessage** shall be written to a memory block of sufficient size. The address of that memory block shall be written to the field **Data** of a variable of type **BioAPI_DATA**, and the length (in octets) of the binary data block shall be written to the field **Length** of that variable. The address of that variable shall be assigned to the native parameter **InMessage**.

8.13.3.6 If the input value of **no_OutMessage** is "**true**", then the native parameter **OutMessage** shall be set to NULL, otherwise it shall be set to the address of a variable of type **BioAPI_DATA**.

8.13.4 Function Invocation Output

8.13.4.1 If the native parameter **OutMessage** is not NULL, then the output value of the parameter with the same name shall be the canonical representation of a binary data block (see 6.7.2). The binary data block shall be read from the memory block whose address is in the field **Data** of the variable of type **BioAPI_DATA** pointed to by the native parameter **OutMessage**, and whose length is in the field **Length** of that variable.

8.13.4.2 If the native parameter **OutMessage** is NULL, then the output value of the parameter with the same name shall be an empty string.

8.14 BioSPI_Capture

8.14.1 General

This function belongs to the BioSPI interface, and has the following native prototype:

```
BioAPI_RETURN BioAPI
BioSPI_Capture(
   BioAPI_HANDLE ModuleHandle,
   BioAPI_BIR_PURPOSE Purpose,
   BioAPI_BIR_HANDLE_PTR CapturedBIR,
   sint32 Timeout,
   BioAPI_BIR_HANDLE_PTR AuditData);
```

8.14.2 Function Invocation Scheme

This function has the following input parameters:

- **ModuleHandle**
- **Purpose**
- **no_CapturedBIR**
- **Timeout**
- **no_AuditData**

and the following output parameters:

- **CapturedBIR**
- **AuditData**

8.14.3 Function Invocation Input

8.14.3.1 The input value of **ModuleHandle** shall be a valid representation of an integer (see 6.4) in the range 0 to 4294967295.

8.14.3.2 The input value of **Purpose** shall be a valid representation of an integer in the range 0 to 255.

8.14.3.3 The input value of **no_CapturedBIR** and **no_AuditData** shall be either a valid representation of a boolean (see 6.5) or an empty string.

8.14.3.4 The input value of **Timeout** shall be a valid representation of an integer in the range -2147483648 to 2147483647.

8.14.3.5 The integer represented by the input value of **ModuleHandle**, **Purpose**, and **Timeout** shall be assigned to the native parameter with the same name.

8.14.3.6 If the input value of **no_CapturedBIR** is "**true**", then the native parameter **CapturedBIR** shall be set to NULL, otherwise it shall be set to the address of a variable of type **BioAPI_BIR_HANDLE**.

8.14.3.7 If the input value of **no_AuditData** is "**true**", then the native parameter **AuditData** shall be set to NULL, otherwise it shall be set to the address of a variable of type **BioAPI_BIR_HANDLE**.

8.14.4 Function Invocation Output

8.14.4.1 If the native parameter **CapturedBIR** is not NULL, then the output value of the parameter with the same name shall be the canonical representation of an integer in the range -2147483648 to 2147483647 (see 6.4.3). The integer shall be read from the variable of type **BioAPI_BIR_HANDLE** pointed to by the native parameter **CapturedBIR**.

8.14.4.2 If the native parameter **AuditData** is not NULL, then the output value of the parameter with the same name shall be the canonical representation of an integer in the range -2147483648 to 2147483647. The integer shall be read from the variable of type **BioAPI_BIR_HANDLE** pointed to by the native parameter **AuditData**.

8.14.4.3 If the native parameter **CapturedBIR** or **AuditData** is NULL, then the output value of the parameter with the same name shall be an empty string.

8.15 BioSPI_CreateTemplate

8.15.1 General

This function belongs to the BioSPI interface, and has the following native prototype:

```
BioAPI_RETURN BioAPI
BioSPI_CreateTemplate(
```

```
BioAPI_HANDLE ModuleHandle,
const BioAPI_INPUT_BIR *CapturedBIR,
const BioAPI_INPUT_BIR *StoredTemplate,
BioAPI_BIR_HANDLE_PTR NewTemplate,
const BioAPI_DATA *Payload);
```

8.15.2 Function Invocation Scheme

This function has the following input parameters:

- **ModuleHandle**
- the members of the parameter group "Input BIR" (see 9.2.6), with their names prefixed by "**CapturedBIR_**"
- the members of the parameter group "Input BIR" (see 9.2.6), with their names prefixed by "**StoredTemplate_**"
- **no_NewTemplate**
- **Payload**

and the following output parameter:

- **NewTemplate**

8.15.3 Function Invocation Input

8.15.3.1 The input value of **ModuleHandle** shall be a valid representation of an integer (see 6.4) in the range 0 to 4294967295.

8.15.3.2 The input value of **no_NewTemplate** shall be either a valid representation of a boolean (see 6.5) or an empty string.

8.15.3.3 The input value of **Payload** shall be a valid representation of a binary data block (see 7.7).

8.15.3.4 The integer represented by the input value of **ModuleHandle** shall be assigned to the native parameter with the same name.

8.15.3.5 The value of type **BioAPI_INPUT_BIR** represented (see 8.2.6.6) by the input value of the parameter group "Input BIR" with the prefix "**CapturedBIR_**" shall be written to a variable of type **BioAPI_INPUT_BIR**, whose address shall be assigned to the native parameter **CapturedBIR**.

8.15.3.6 The value of type **BioAPI_INPUT_BIR** represented (see 8.2.6.6) by the input value of the parameter group "Input BIR" with the prefix "**StoredTemplate_**" shall be written to a variable of type **BioAPI_INPUT_BIR**, whose address shall be assigned to the native parameter **StoredTemplate**.

8.15.3.7 If the input value of **no_NewTemplate** is "**true**", then the native parameter **NewTemplate** shall be set to NULL, otherwise it shall be set to the address of a variable of type **BioAPI_BIR_HANDLE**.

8.15.3.8 If the input value of **Payload** is not an empty string, then the binary data block it represents shall be written to a memory block of sufficient size; the address of that memory

block shall be written to the field **Data** of a variable of type **BioAPI_DATA**, and the length (in octets) of the binary data block shall be written to the field **Length** of that variable; the address of that variable shall be assigned to the native parameter **Payload**. Otherwise, the native parameter **Payload** shall be set to NULL.

8.15.4 Function Invocation Output

8.15.4.1 If the native parameter **NewTemplate** is not NULL, then the output value of the parameter with the same name shall be the canonical representation of an integer in the range -2147483648 to 2147483647 (see 6.4.3). The integer shall be read from the variable of type **BioAPI_BIR_HANDLE** pointed to by the native parameter **NewTemplate**.

8.15.4.2 If the native parameter **NewTemplate** is NULL, then the output value of the parameter with the same name shall be an empty string.

8.16 BioSPI_Process

8.16.1 General

This function belongs to the BioSPI interface, and has the following native prototype:

```
BioAPI_RETURN BioAPI
BioSPI_Process(
   BioAPI_HANDLE ModuleHandle,
   const BioAPI_INPUT_BIR *CapturedBIR,
   BioAPI_BIR_HANDLE_PTR ProcessedBIR);
```

8.16.2 Function Invocation Scheme

This function has the following input parameters:

- **ModuleHandle**
- the members of the parameter group "Input BIR" (see 9.2.6), with their names prefixed by "**CapturedBIR_**"
- **no_ProcessedBIR**

and the following output parameter:

- **ProcessedBIR**

8.16.3 Function Invocation Input

8.16.3.1 The input value of **ModuleHandle** shall be a valid representation of an integer (see 6.4) in the range 0 to 4294967295.

8.16.3.2 The input value of **no_ProcessedBIR** shall be either a valid representation of a boolean (see 6.5) or an empty string.

8.16.3.3 The integer represented by the input value of **ModuleHandle** shall be assigned to the native parameter with the same name.

8.16.3.4 The value of type **BioAPI_INPUT_BIR** represented (see 8.2.6.6) by the input value of the parameter group "Input BIR" shall be written to a variable of type **BioAPI_INPUT_BIR**, whose address shall be assigned to the native parameter **CapturedBIR**.

8.16.3.5 If the input value of **no_ProcessedBIR** is "**true**", then the native parameter **ProcessedBIR** shall be set to NULL, otherwise it shall be set to the address of a variable of type **BioAPI_BIR_HANDLE**.

8.16.4 Function Invocation Output

8.16.4.1 If the native parameter **ProcessedBIR** is not NULL, then the output value of the parameter with the same name shall be the canonical representation of an integer in the range -2147483648 to 2147483647 (see 6.4.3). The integer shall be read from the variable of type **BioAPI_BIR_HANDLE** pointed to by the native parameter **ProcessedBIR**.

8.16.4.2 If the native parameter **ProcessedBIR** is NULL, then the output value of the parameter with the same name shall be an empty string.

8.17 BioSPI_VerifyMatch

8.17.1 General

This function belongs to the BioSPI interface, and has the following native prototype:

```
BioAPI_RETURN BioAPI
BioSPI_VerifyMatch(
   BioAPI_HANDLE ModuleHandle,
   const BioAPI_FAR *MaxFARRequested,
   const BioAPI_FRR *MaxFRRRequested,
   const BioAPI_BOOL *FARPrecedence,
   const BioAPI_INPUT_BIR *ProcessedBIR,
   const BioAPI_INPUT_BIR *StoredTemplate,
   BioAPI_BIR_HANDLE *AdaptedBIR,
   BioAPI_BOOL *Result,
   BioAPI_FAR_PTR FARAchieved,
   BioAPI_FRR_PTR FRRAchieved,
   BioAPI_DATA_PTR *Payload);
```

8.17.2 Function Invocation Scheme

This function has the following input parameters:

- **ModuleHandle**
- **MaxFARRequested**
- **MaxFRRRequested**
- **FARPrecedence**
- the members of the parameter group "Input BIR" (see 9.2.6), with their names prefixed by "**ProcessedBIR_**"
- the members of the parameter group "Input BIR" (see 9.2.6), with their names prefixed by "**StoredTemplate_**"
- **no_AdaptedBIR**
- **no_Result**
- **no_FARAchieved**
- **no_FRRAchieved**
- **no_Payload**

and the following output parameters:

- **AdaptedBIR**
- **Result**
- **FARAchieved**
- **FRRAchieved**
- **Payload**

8.17.3 Function Invocation Input

8.17.3.1 The input value of **ModuleHandle** shall be a valid representation of an integer (see 6.4) in the range 0 to 4294967295.

8.17.3.2 The input value of **MaxFARRequested** shall be a valid representation of an integer in the range -2147483648 to 2147483647, or an empty string.

8.17.3.3 The input value of **MaxFRRRequested** shall be either a valid representation of an integer in the range -2147483648 to 2147483647, or an empty string.

8.17.3.4 The input value of **FARPrecedence** shall be either a valid representation of a boolean (see 6.5), or an empty string.

8.17.3.5 The input value of **no_AdaptedBIR**, **no_Result**, **no_FARAchieved**, **no_FRRAchieved**, and **no_Payload** shall be either a valid representation of a boolean (see 6.5) or an empty string.

8.17.3.6 The integer represented by the input value of **ModuleHandle** shall be assigned to the native parameter with the same name.

8.17.3.7 If the input value of **MaxFARRequested** is not an empty string, the integer it represents shall be written to a variable of type **BioAPI_FAR**, whose address shall be assigned to the native parameter **MaxFARRequested**. Otherwise, the native parameter **MaxFARRequested** shall be set to NULL.

8.17.3.8 If the input value of **MaxFRRRequested** is not an empty string, the integer it represents shall be written to a variable of type **BioAPI_FRR**, whose address shall be assigned to the native parameter **MaxFRRRequested**. Otherwise, the native parameter **MaxFRRRequested** shall be set to NULL.

8.17.3.9 If the input value of **FARPrecedence** is not an empty string, the boolean it represents shall be written to a variable of type **BioAPI_BOOL**, whose address shall be assigned to the native parameter **FARPrecedence**. Otherwise, the native parameter **FARPrecedence** shall be set to NULL.

8.17.3.10 The value of type **BioAPI_INPUT_BIR** represented (see 8.2.6.6) by the input value of the parameter group "Input BIR" with the prefix "**ProcessedBIR_**" shall be written to a variable of type **BioAPI_INPUT_BIR**, whose address shall be assigned to the native parameter **ProcessedBIR**.

8.17.3.11 The value of type **BioAPI_INPUT_BIR** represented (see 8.2.6.6) by the input value of the parameter group "Input BIR" with the prefix "**StoredTemplate_**" shall be written to a variable of type **BioAPI_INPUT_BIR**, whose address shall be assigned to the native parameter **StoredTemplate**.

8.17.3.12 If the input value of **no_AdaptedBIR** is "**true**", then the native parameter **AdaptedBIR** shall be set to NULL, otherwise it shall be set to the address of a variable of type **BioAPI_BIR_HANDLE**.

8.17.3.13 If the input value of **no_Result** is "**true**", then the native parameter **Result** shall be set to NULL, otherwise it shall be set to the address of a variable of type **BioAPI_BOOL**.

8.17.3.14 If the input value of **no_FARAchieved** is "**true**", then the native parameter **FARAchieved** shall be set to NULL, otherwise it shall be set to the address of a variable of type **BioAPI_FAR**.

8.17.3.15 If the input value of **no_FRRAchieved** is "**true**", then the native parameter **FRRAchieved** shall be set to NULL, otherwise it shall be set to the address of a variable of type **BioAPI_FRR**.

8.17.3.16 If the input value of **no_Payload** is "**true**", then the native parameter **Payload** shall be set to NULL, otherwise it shall be set to the address of a variable of type **BioAPI_DATA**.

8.17.4 Function Invocation Output

8.17.4.1 If the native parameter **AdaptedBIR** is not NULL, then the output value of the parameter with the same name shall be the canonical representation of an integer in the range -2147483648 to 2147483647 (see 6.4.3). The integer shall be read from the variable of type **BioAPI_BIR_HANDLE** pointed to by the native parameter **AdaptedBIR**.

8.17.4.2 If the native parameter **FARAchieved** is not NULL, then the output value of the parameter with the same name shall be the canonical representation of an integer in the range -2147483648 to 2147483647 (see 6.4.3). The integer shall be read from the variable of type **BioAPI_FAR** pointed to by the native parameter **FARAchieved.**

8.17.4.3 If the native parameter **FRRAchieved** is not NULL, then the output value of the parameter with the same name shall be the canonical representation of an integer in the range -2147483648 to 2147483647 (see 6.4.3). The integer shall be read from the variable of type **BioAPI_FRR** pointed to by the native parameter **FRRAchieved.**

8.17.4.4 If the native parameter **Result** is not NULL, then the output value of the parameter with the same name shall be the canonical representation of a boolean (see 6.5.2). The boolean shall be read from the variable of type **BioAPI_BOOL** pointed to by the native parameter **Result**.

8.17.4.5 If the native parameter **Payload** is not NULL, then the output value of the parameter with the same name shall be the canonical representation of a binary data block (see 6.7.2). The binary data block shall be read from the memory block whose address is in the field **Data** of the variable of type **BioAPI_DATA** pointed to by the native parameter **Payload**, and whose length is in the field **Length** of that variable.

8.17.4.6 If the native parameter **AdaptedBIR**, **FARAchieved**, **FRRAchieved**, **Result**, **or Payload** is NULL, then the output value of the parameter with the same name shall be an empty string.

8.18 BioSPI_IdentifyMatch

8.18.1 General

This function belongs to the BioSPI interface, and has the following native prototype:

```
BioAPI_RETURN BioAPI
BioSPI_IdentifyMatch(
   BioAPI_HANDLE ModuleHandle,
   const BioAPI_FAR *MaxFARRequested,
   const BioAPI_FRR *MaxFRRRequested,
   const BioAPI_BOOL *FARPrecedence,
   const BioAPI_INPUT_BIR *ProcessedBIR,
   const BioAPI_IDENTIFY_POPULATION *Population,
   BioAPI_BOOL Binning,
   uint32 MaxNumberOfResults,
   uint32 *NumberOfResults,
   BioAPI_CANDIDATE_ARRAY_PTR *Candidates,
   sint32 Timeout);
```

8.18.2 Function Invocation Scheme

This function has the following input parameters:

- **ModuleHandle**
- **MaxFARRequested**
- **MaxFRRRequested**
- **FARPrecedence**
- the members of the parameter group "Input BIR" (see 9.2.6), with their names prefixed by "**ProcessedBIR_**"
- the members of the parameter group "Identify population" (see 9.2.7)
- **Binning**
- **MaxNumberOfResults**
- **no_NumberOfResults**
- **no_Candidates**
- **Timeout**

and the following output parameters:

- **NumberOfResults**
- the members of the parameter group "Candidate" (see 9.2.8), with their names prefixed by "**Candidate_1_**"
- the members of the parameter group "Candidate" (see 9.2.8), with their names prefixed by "**Candidate_2_**"
- the members of the parameter group "Candidate" (see 9.2.8), with their names prefixed by "**Candidate_3_**"

- the members of the parameter group "Candidate" (see 9.2.8), with their names prefixed by "**Candidate_4_**"

8.18.3 Function Invocation Input

8.18.3.1 The input value of **ModuleHandle** shall be a valid representation of an integer (see 6.4) in the range 0 to 4294967295.

8.18.3.2 The input value of **MaxFARRequested** shall be a valid representation of an integer in the range -2147483648 to 2147483647, or an empty string.

8.18.3.3 The input value of **MaxFRRRequested** shall be either a valid representation of an integer in the range -2147483648 to 2147483647, or an empty string.

8.18.3.4 The input value of **FARPrecedence** shall be either a valid representation of a boolean (see 6.5), or an empty string.

8.18.3.5 The input value of **Binning** shall be a valid representation of a boolean.

8.18.3.6 The input value of **MaxNumberOfResults** shall be a valid representation of an integer in the range 0 to 4294967295.

8.18.3.7 The input value of **no_NumberOfResults** and **no_Candidates** shall be either a valid representation of a boolean (see 6.5) or an empty string.

8.18.3.8 The input value of **Timeout** shall be a valid representation of an integer in the range -2147483648 to 2147483647.

8.18.3.9 The integer represented by the input value of **ModuleHandle**, **MaxNumberOfResults**, and **Timeout** shall be assigned to the native parameter with the same name.

8.18.3.10 The boolean represented by the input value of **Binning** shall be assigned to the native parameter with the same name.

8.18.3.11 If the input value of **MaxFARRequested** is not an empty string, the integer it represents shall be written to a variable of type **BioAPI_FAR**, whose address shall be assigned to the native parameter **MaxFARRequested**. Otherwise, the native parameter **MaxFARRequested** shall be set to NULL.

8.18.3.12 If the input value of **MaxFRRRequested** is not an empty string, the integer it represents shall be written to a variable of type **BioAPI_FRR**, whose address shall be assigned to the native parameter **MaxFRRRequested**. Otherwise, the native parameter **MaxFRRRequested** shall be set to NULL.

8.18.3.13 If the input value of **FARPrecedence** is not an empty string, the boolean it represents shall be written to a variable of type **BioAPI_BOOL**, whose address shall be assigned to the native parameter **FARPrecedence**. Otherwise, the native parameter **FARPrecedence** shall be set to NULL.

8.18.3.14 The value of type **BioAPI_INPUT_BIR** represented (see 8.2.6.6) by the input value of the parameter group "Input BIR" shall be written to a variable of type **BioAPI_INPUT_BIR**, whose address shall be assigned to the native parameter **ProcessedBIR**.

8.18.3.15 The value of type **BioAPI_IDENTIFY_POPULATION** represented (see 8.2.7.6) by the input value of the parameter group "Identify population" shall be written to a variable of type **BioAPI_IDENTIFY_POPULATION**, whose address shall be assigned to the native parameter **Population**.

8.18.3.16 If the input value of **no_NumberOfResults** is "**true**", then the native parameter **NumberOfResults** shall be set to NULL, otherwise it shall be set to the address of a variable of type **uint32**.

8.18.3.17 If the input value of **no_Candidates** is "**true**", then the native parameter **Candidates** shall be set to NULL, otherwise it shall be set to the address of a variable of type **BioAPI_CANDIDATE_ARRAY**.

8.18.4 Function Invocation Output

8.18.4.1 If the native parameter **NumberOfResults** is not NULL, then the output value of the parameter with the same name shall be the canonical representation of an integer in the range 0 to 4294967295 (see 6.4.3). The integer shall be read from the variable of type **uint32** pointed to by the native parameter **NumberOfResults**.

8.18.4.2 If the native parameter **NumberOfResults** is NULL, then the output value of the parameter with the same name shall be an empty string.

8.18.4.3 If the native parameter **Candidates** is not NULL, then the output value of the parameter group "Candidate" with the prefix "**Candidate_1_**", "**Candidate_2_**", "**Candidate_3_**", and "**Candidate_4_**" shall be the canonical representation (see 8.2.5.5) of the value of type **BioAPI_CANDIDATE** in the corresponding position in the array pointed to by the native parameter **Candidates**, provided that:

a) the native parameter **NumberOfResults** is not NULL (and therefore points to an integer variable whose value is, say, N);

b) the position in the array is not beyond N; and

c) the pointer at that position in the array is not NULL.

In all other cases, the values of all members of an instance of the parameter group "Candidate" shall be empty strings.

8.19 BioSPI_Enroll

8.19.1 General

This function belongs to the BioSPI interface, and has the following native prototype:

```
BioAPI_RETURN BioAPI
BioSPI_Enroll(
   BioAPI_HANDLE ModuleHandle,
   BioAPI_BIR_PURPOSE Purpose,
   const BioAPI_INPUT_BIR *StoredTemplate,
```

```
BioAPI_BIR_HANDLE_PTR NewTemplate,
const BioAPI_DATA *Payload,
sint32 Timeout,
BioAPI_BIR_HANDLE_PTR AuditData);
```

8.19.2 Function Invocation Scheme

This function has the following input parameters:

- **ModuleHandle**
- **Purpose**
- the members of the parameter group "Input BIR" (see 9.2.6), with their names prefixed by "**StoredTemplate_**"
- **no_NewTemplate**
- **Payload**
- **Timeout**
- **no_AuditData**

and the following output parameters:

- **NewTemplate**
- **AuditData**

8.19.3 Function Invocation Input

8.19.3.1 The input value of **ModuleHandle** shall be a valid representation of an integer (see 6.4) in the range 0 to 4294967295.

8.19.3.2 The input value of **Purpose** shall be a valid representation of an integer in the range 0 to 255.

8.19.3.3 The input value of **no_NewTemplate** and **no_AuditData** shall be either a valid representation of a boolean (see 6.5) or an empty string.

8.19.3.4 The input value of **Payload** shall be a valid representation of a binary data block (see 6.7).

8.19.3.5 The input value of **Timeout** shall be a valid representation of an integer in the range -2147483648 to 2147483647.

8.19.3.6 The integer represented by the input value of **ModuleHandle**, **Purpose**, and **Timeout** shall be assigned to the native parameter with the same name.

8.19.3.7 The value of type **BioAPI_INPUT_BIR** represented (see 8.2.6.6) by the input value of the parameter group "Input BIR" shall be written to a variable of type **BioAPI_INPUT_BIR**, whose address shall be assigned to the native parameter **StoredTemplate**.

8.19.3.8 If the input value of **no_NewTemplate** is "**true**", then the native parameter **NewTemplate** shall be set to NULL, otherwise it shall be set to the address of a variable of type **BioAPI_BIR_HANDLE**.

8.19.3.9 If the input value of **no_AuditData** is "**true**", then the native parameter **AuditData** shall be set to NULL, otherwise it shall be set to the address of a variable of type **BioAPI_BIR_HANDLE**.

8.19.3.10 If the input value of **Payload** is not an empty string, then the binary data block it represents shall be written to a memory block of sufficient size; the address of that memory block shall be written to the field **Data** of a variable of type **BioAPI_DATA**, and the length (in octets) of the binary data block shall be written to the field **Length** of that variable; the address of that variable shall be assigned to the native parameter **Payload**. Otherwise, the native parameter **Payload** shall be set to NULL.

8.19.4 Function Invocation Output

8.19.4.1 If the native parameter **NewTemplate** is not NULL, then the output value of the parameter with the same name shall be the canonical representation of an integer in the range -2147483648 to 2147483647 (see 6.4.3). The integer shall be read from the variable of type **BioAPI_BIR_HANDLE** pointed to by the native parameter **NewTemplate**.

8.19.4.2 If the native parameter **AuditData** is not NULL, then the output value of the parameter with the same name shall be the canonical representation of an integer in the range -2147483648 to 2147483647 (see 6.4.3). The integer shall be read from the variable of type **BioAPI_BIR_HANDLE** pointed to by the native parameter **AuditData**.

8.19.4.3 If the native parameter **NewTemplate** or **AuditData** is NULL, then the output value of the parameter with the same name shall be an empty string.

8.20 BioSPI_Verify

8.20.1 General

This function belongs to the BioSPI interface, and has the following native prototype:

```
BioAPI_RETURN BioAPI
BioSPI_Verify(
   BioAPI_HANDLE ModuleHandle,
   const BioAPI_FAR *MaxFARRequested,
   const BioAPI_FRR *MaxFRRRequested,
   const BioAPI_BOOL *FARPrecedence,
   const BioAPI_INPUT_BIR *StoredTemplate,
   BioAPI_BIR_HANDLE_PTR AdaptedBIR,
   BioAPI_BOOL *Result,
   BioAPI_FAR_PTR FARAchieved,
   BioAPI_FRR_PTR FRRAchieved,
   BioAPI_DATA_PTR *Payload,
   sint32 Timeout,
   BioAPI_BIR_HANDLE_PTR AuditData);
```

8.20.2 Function Invocation Scheme

This function has the following input parameters:

- **ModuleHandle**
- **MaxFARRequested**
- **MaxFRRRequested**

- **FARPrecedence**
- the members of the parameter group "Input BIR" (see 9.2.6), with their names prefixed by "**StoredTemplate_**"
- **no_AdaptedBIR**
- **no_Result**
- **no_FARAchieved**
- **no_FRRAchieved**
- **no_Payload**
- **Timeout**
- **no_AuditData**

and the following output parameters:

- **AdaptedBIR**
- **Result**
- **FARAchieved**
- **FRRAchieved**
- **Payload**
- **AuditData**

8.20.3 Function Invocation Input

8.20.3.1 The input value of **ModuleHandle** shall be a valid representation of an integer (see 6.4) in the range 0 to 4294967295.

8.20.3.2 The input value of **MaxFARRequested** shall be a valid representation of an integer in the range -2147483648 to 2147483647, or an empty string.

8.20.3.3 The input value of **MaxFRRRequested** shall be either a valid representation of an integer in the range -2147483648 to 2147483647, or an empty string.

8.20.3.4 The input value of **FARPrecedence** shall be either a valid representation of a boolean (see 6.5), or an empty string.

8.20.3.5 The input value of **no_AdaptedBIR**, **no_Result**, **no_FARAchieved**, **no_FRRAchieved**, **no_Payload**, and **no_AuditData** shall be either a valid representation of a boolean (see 6.5) or an empty string.

8.20.3.6 The input value of **Timeout** shall be a valid representation of an integer in the range -2147483648 to 2147483647.

8.20.3.7 The integer represented by the input value of **ModuleHandle** and **Timeout** shall be assigned to the native parameter with the same name.

8.20.3.8 If the input value of **MaxFARRequested** is not an empty string, the integer it represents shall be written to a variable of type **BioAPI_FAR**, whose address shall be assigned to the native

parameter **MaxFARRequested**. Otherwise, the native parameter **MaxFARRequested** shall be set to NULL.

8.20.3.9 If the input value of **MaxFRRRequested** is not an empty string, the integer it represents shall be written to a variable of type **BioAPI_FRR**, whose address shall be assigned to the native parameter **MaxFRRRequested**. Otherwise, the native parameter **MaxFRRRequested** shall be set to NULL.

8.20.3.10 If the input value of **FARPrecedence** is not an empty string, the boolean it represents shall be written to a variable of type **BioAPI_BOOL**, whose address shall be assigned to the native parameter **FARPrecedence**. Otherwise, the native parameter **FARPrecedence** shall be set to NULL.

8.20.3.11 The value of type **BioAPI_INPUT_BIR** represented (see 8.2.6.6) by the input value of the parameter group "Input BIR" shall be written to a variable of type **BioAPI_INPUT_BIR**, whose address shall be assigned to the native parameter **StoredTemplate**.

8.20.3.12 If the input value of **no_AdaptedBIR** is "**true**", then the native parameter **AdaptedBIR** shall be set to NULL, otherwise it shall be set to the address of a variable of type **BioAPI_BIR_HANDLE**.

8.20.3.13 If the input value of **no_Result** is "**true**", then the native parameter **Result** shall be set to NULL, otherwise it shall be set to the address of a variable of type **BioAPI_BOOL**.

8.20.3.14 If the input value of **no_FARAchieved** is "**true**", then the native parameter **FARAchieved** shall be set to NULL, otherwise it shall be set to the address of a variable of type **BioAPI_FAR**.

8.20.3.15 If the input value of **no_FRRAchieved** is "**true**", then the native parameter **FRRAchieved** shall be set to NULL, otherwise it shall be set to the address of a variable of type **BioAPI_FRR**.

8.20.3.16 If the input value of **no_Payload** is "**true**", then the native parameter **Payload** shall be set to NULL, otherwise it shall be set to the address of a variable of type **BioAPI_DATA**.

8.20.3.17 If the input value of **no_AuditData** is "**true**", then the native parameter **AuditData** shall be set to NULL, otherwise it shall be set to the address of a variable of type **BioAPI_BIR_HANDLE**.

8.20.4 Function Invocation Output

8.20.4.1 If the native parameter **AdaptedBIR** is not NULL, then the output value of the parameter with the same name shall be the canonical representation of an integer in the range -2147483648 to 2147483647 (see 6.4.3). The integer shall be read from the variable of type **BioAPI_BIR_HANDLE** pointed to by the native parameter **AdaptedBIR**.

8.20.4.2 If the native parameter **AuditData** is not NULL, then the output value of the parameter with the same name shall be the canonical representation of an integer in the range -2147483648 to 2147483647. The integer shall be read from the variable of type **BioAPI_BIR_HANDLE** pointed to by the native parameter **AuditData**.

8.20.4.3 If the native parameter **FARAchieved** is not NULL, then the output value of the parameter with the same name shall be the canonical representation of an integer in the range -2147483648 to 2147483647. The integer shall be read from the variable of type **BioAPI_FAR** pointed to by the native parameter **FARAchieved**.

8.20.4.4 If the native parameter **FRRAchieved** is not NULL, then the output value of the parameter with the same name shall be the canonical representation of an integer in the range -2147483648 to 2147483647. The integer shall be read from the variable of type **BioAPI_FRR** pointed to by the native parameter **FRRAchieved**.

8.20.4.5 If the native parameter **Result** is not NULL, then the output value of the parameter with the same name shall be the canonical representation of a boolean (see 6.5.2). The boolean shall be read from the variable of type **BioAPI_BOOL** pointed to by the native parameter **Result**.

8.20.4.6 If the native parameter **Payload** is not NULL, then the output value of the parameter with the same name shall be the canonical representation of a binary data block (see 6.7.2). The binary data block shall be read from the memory block whose address is in the field **Data** of the variable of type **BioAPI_DATA** pointed to by the native parameter **Payload**, and whose length is in the field **Length** of that variable.

8.20.4.7 If the native parameter **AdaptedBIR**, **AuditData**, **FARAchieved**, **FRRAchieved**, **Result**, or **Payload** is NULL, then the output value of the parameter with the same name shall be an empty string.

8.21 BioSPI_Identify

8.21.1 General

This function belongs to the BioSPI interface, and has the following native prototype:

```
BioAPI_RETURN BioAPI
BioSPI_Identify(
   BioAPI_HANDLE ModuleHandle,
   const BioAPI_FAR *MaxFARRequested,
   const BioAPI_FRR *MaxFRRRequested,
   const BioAPI_BOOL *FARPrecedence,
   const BioAPI_IDENTIFY_POPULATION *Population,
   BioAPI_BOOL Binning,
   uint32 MaxNumberOfResults,
   uint32 *NumberOfResults,
   BioAPI_CANDIDATE_ARRAY_PTR *Candidates,
   sint32 Timeout,
   BioAPI_BIR_HANDLE_PTR AuditData);
```

8.21.2 Function Invocation Scheme

This function has the following input parameters:

- **ModuleHandle**
- **MaxFARRequested**
- **MaxFRRRequested**
- **FARPrecedence**
- the members of the parameter group "Identify population" (see 9.2.7)

- **Binning**
- **MaxNumberOfResults**
- **no_NumberOfResults**
- **no_Candidates**
- **Timeout**
- **no_AuditData**

and the following output parameters:

- **NumberOfResults**
- the members of the parameter group "Candidate" (see 9.2.8), with their names prefixed by "**Candidate_1_** "
- the members of the parameter group "Candidate" (see 9.2.8), with their names prefixed by "**Candidate_2_**"
- the members of the parameter group "Candidate" (see 9.2.8), with their names prefixed by "**Candidate_3_**"
- the members of the parameter group "Candidate" (see 9.2.8), with their names prefixed by "**Candidate_4_**"
- **AuditData**

8.21.3 Function Invocation Input

8.21.3.1 The input value of **ModuleHandle** shall be a valid representation of an integer (see 6.4) in the range 0 to 4294967295.

8.21.3.2 The input value of **MaxFARRequested** shall be a valid representation of an integer in the range -2147483648 to 2147483647, or an empty string.

8.21.3.3 The input value of **MaxFRRRequested** shall be either a valid representation of an integer in the range -2147483648 to 2147483647, or an empty string.

8.21.3.4 The input value of **FARPrecedence** shall be either a valid representation of a boolean, or an empty string.

8.21.3.5 The input value of **Binning** shall be a valid representation of a boolean.

8.21.3.6 The input value of **MaxNumberOfResults** shall be a valid representation of an integer in the range 0 to 4294967295.

8.21.3.7 The input value of **no_NumberOfResults, no_Candidates**, and **no_AuditData** shall be either a valid representation of a boolean (see 6.5) or an empty string.

8.21.3.8 The input value of **Timeout** shall be a valid representation of an integer in the range -2147483648 to 2147483647.

8.21.3.9 The integer represented by the input value of **ModuleHandle**, **MaxNumberOfResults**, and **Timeout** shall be assigned to the native parameter with the same name.

8.21.3.10 If the input value of **MaxFARRequested** is not an empty string, the integer it represents shall be written to a variable of type **BioAPI_FAR**, whose address shall be assigned to the native parameter **MaxFARRequested**. Otherwise, the native parameter **MaxFARRequested** shall be set to NULL.

8.21.3.11 If the input value of **MaxFRRRequested** is not an empty string, the integer it represents shall be written to a variable of type **BioAPI_FRR**, whose address shall be assigned to the native parameter **MaxFRRRequested**. Otherwise, the native parameter **MaxFRRRequested** shall be set to NULL.

8.21.3.12 If the input value of **FARPrecedence** is not an empty string, the boolean it represents shall be written to a variable of type **BioAPI_BOOL**, whose address shall be assigned to the native parameter **FARPrecedence**. Otherwise, the native parameter **FARPrecedence** shall be set to NULL.

8.21.3.13 The value of type **BioAPI_IDENTIFY_POPULATION** represented (see 8.2.7.6) by the input value of the parameter group "Identify population" shall be written to a variable of type **BioAPI_IDENTIFY_POPULATION**, whose address shall be assigned to the native parameter **Population**.

8.21.3.14 The boolean represented by the input value of **Binning** shall be assigned to the native parameter with the same name.

8.21.3.15 If the input value of **no_NumberOfResults** is "**true**", then the native parameter **NumberOfResults** shall be set to NULL, otherwise it shall be set to the address of a variable of type **uint32**.

8.21.3.16 If the input value of **no_Candidates** is "**true**", then the native parameter **Candidates** shall be set to NULL, otherwise it shall be set to the address of a variable of type **BioAPI_CANDIDATE_ARRAY**.

8.21.3.17 If the input value of **no_AuditData** is "**true**", then the native parameter **AuditData** shall be set to NULL, otherwise it shall be set to the address of a variable of type **BioAPI_BIR_HANDLE**.

8.21.4 Function Invocation Output

8.21.4.1 If the native parameter **NumberOfResults** is not NULL, then the output value of the parameter with the same name shall be the canonical representation of an integer in the range 0 to 4294967295 (see 6.4.3). The integer (say, N) shall be read from the variable of type **uint32** pointed to by the native parameter **NumberOfResults**.

8.21.4.2 If the native parameter **NumberOfResults** is NULL, then the output value of the parameter with the same name shall be an empty string.

8.21.4.3 If the native parameter **Candidates** is not NULL, then the output value of the parameter group "Candidate" with the prefix "**Candidate_1_**", "**Candidate_2_**", "**Candidate_3_**", and "**Candidate_4_**" shall be the canonical representation (see 8.2.5.5) of the value of type **BioAPI_CANDIDATE** in the corresponding position in the array pointed to by the native parameter **Candidates**, provided that:

a) the native parameter **NumberOfResults** is not NULL (and therefore points to an integer variable whose value is, say, N);

b) the position in the array is not beyond N; and

c) the pointer at that position in the array is not NULL.

In all other cases, the values of all members of an instance of the parameter group "Candidate" shall be empty strings.

8.21.4.4 If the native parameter **AuditData** is not NULL, then the output value of the parameter with the same name shall be the canonical representation of an integer in the range -2147483648 to 2147483647 (see 6.4.3). The integer shall be read from the variable of type **BioAPI_BIR_HANDLE** pointed to by the native parameter **AuditData**.

8.21.4.5 If the native parameter **AuditData** is NULL, then the output value of the parameter with the same name shall be an empty string.

8.22 BioSPI_Import

8.22.1 General

This function belongs to the BioSPI interface, and has the following native prototype:

```
BioAPI_RETURN BioAPI
BioSPI_Import(
   BioAPI_HANDLE ModuleHandle,
   const BioAPI_DATA *InputData,
   BioAPI_BIR_BIOMETRIC_DATA_FORMAT InputFormat,
   BioAPI_BIR_PURPOSE Purpose,
   BioAPI_BIR_HANDLE_PTR ConstructedBIR);
```

8.22.2 Function Invocation Scheme

This function has the following input parameters:

- **ModuleHandle**
- **InputData**
- **InputFormatOwner**
- **InputFormatID**
- **Purpose**
- **no_ConstructedBIR**

and the following output parameter:

- **ConstructedBIR**

8.22.3 Function Invocation Input

8.22.3.1 The input value of **ModuleHandle** shall be a valid representation of an integer (see 6.4) in the range 0 to 4294967295.

8.22.3.2 The input value of **InputData** shall be a valid representation of a binary data block (see 6.7).

8.22.3.3 The input value of **InputFormatOwner** and **InputFormatID** shall be a valid representation of an integer in the range 0 to 65535.

8.22.3.4 The input value of **Purpose** shall be a valid representation of an integer in the range 0 to 255.

8.22.3.5 The input value of **no_ConstructedBIR** shall be either a valid representation of a boolean (see 6.5) or an empty string.

8.22.3.6 The integer represented by the input value of **ModuleHandle** and **Purpose** shall be assigned to the native parameter with the same name.

8.22.3.7 If the input value of **InputData** is not an empty string, then the binary data block it represents shall be written to a memory block of sufficient size; the address of that memory block shall be written to the field **Data** of a variable of type **BioAPI_DATA**, and the length (in octets) of the binary data block shall be written to the field **Length** of that variable; the address of that variable shall be assigned to the native parameter **InputData**. Otherwise, the native parameter **InputData** shall be set to NULL.

8.22.3.8 The integer represented by the input value of **InputFormatOwner** shall be written to the field **FormatOwner** of a variable of type **BioAPI_BIR_BIOMETRIC_DATA_FORMAT**, and the integer represented by the input value of **InputFormatID** shall be written to the field **FormatID** of that variable. The address of that variable shall be assigned to the native parameter **InputFormat**.

8.22.3.9 If the input value of **no_ConstructedBIR** is "**true**", then the native parameter **ConstructedBIR** shall be set to NULL, otherwise it shall be set to the address of a variable of type **BioAPI_BIR_HANDLE**.

8.22.4 Function Invocation Output

8.22.4.1 If the native parameter **ConstructedBIR** is not NULL, then the output value of the parameter with the same name shall be the canonical representation of an integer in the range -2147483648 to 2147483647 (see 6.4.3). The integer shall be read from the variable of type **BioAPI_BIR_HANDLE** pointed to by the native parameter **ConstructedBIR**.

8.22.4.2 If the native parameter **ConstructedBIR** is NULL, then the output value of the parameter with the same name shall be an empty string.

8.23 BioSPI_SetPowerMode

8.23.1 General

This function belongs to the BioSPI interface, and has the following native prototype:

```
BioAPI_RETURN BioAPI
BioSPI_SetPowerMode(
   BioAPI_HANDLE ModuleHandle,
   BioAPI_POWER_MODE PowerMode);
```

8.23.2 Function Invocation Scheme

This function has the following input parameters:

- **ModuleHandle**
- **PowerMode**

and no output parameters.

8.23.3 Function Invocation Input

8.23.3.1 The input value of **ModuleHandle** shall be a valid representation of an integer (see 6.4) in the range 0 to 4294967295.

8.23.3.2 The input value of **PowerMode** shall be a valid representation of an integer in the range 0 to 4294967295.

8.23.3.3 The integer represented by the input value of **ModuleHandle** and **PowerMode** shall be assigned to the native parameter with the same name.

8.23.4 Function Invocation Output

There are no output parameters.

8.24 BioSPI_DbOpen

8.24.1 General

This function belongs to the BioSPI interface, and has the following native prototype:

```
BioAPI_RETURN BioAPI
BioSPI_DbOpen(
   BioAPI_HANDLE ModuleHandle,
   const uint8 *DbName,
   BioAPI_DB_ACCESS_TYPE AccessRequest,
   BioAPI_DB_HANDLE_PTR DbHandle,
   BioAPI_DB_CURSOR_PTR Cursor);
```

8.24.2 Function Invocation Scheme

This function has the following input parameters:

- **ModuleHandle**
- **DbName**
- **ReadAccessRequest**
- **WriteAccessRequest**
- **no_DbHandle**
- **no_Cursor**

and the following output parameters:

- **DbHandle**
- **Cursor**

8.24.3 Function Invocation Input

8.24.3.1 The input value of **ModuleHandle** shall be a valid representation of an integer (see 6.4) in the range 0 to 4294967295.

8.24.3.2 The input value of **DbName** shall be a character string.

8.24.3.3 The input value of **ReadAccessRequest** and **WriteAccessRequest** shall be either a valid representation of a boolean (see 6.5), or an empty string.

8.24.3.4 The input value of **no_DbHandle** and **no_Cursor** shall be either a valid representation of a boolean (see 6.5) or an empty string.

8.24.3.5 The integer represented by the input value of **ModuleHandle** shall be assigned to the native parameter with the same name.

8.24.3.6 The input value of **DbName** shall be written to a memory block of sufficient size, whose address shall be assigned to the native parameter **DbName.**

8.24.3.7 Two integer values, say V1 and V2, shall be set from the input values of **ReadAccessRequest** and **WriteAccessRequest** as follows:

a) if the input value of **ReadAccessRequest** is "**true**", then V1 shall be 1, otherwise it shall be 0;

b) if the input value of **WriteAccessRequest** is "**true**", then V2 shall be 2, otherwise it shall be 0.

8.24.3.8 The sum of the values V1 and V2 shall be assigned to the native parameter **AccessRequest**.

8.24.3.9 If the input value of **no_DbHandle** is "**true**", then the native parameter **DbHandle** shall be set to NULL, otherwise it shall be set to the address of a variable of type **BioAPI_DB_HANDLE**.

8.24.3.10 If the input value of **no_Cursor** is "**true**", then the native parameter **Cursor** shall be set to NULL, otherwise it shall be set to the address of a variable of type **BioAPI_DB_CURSOR**.

8.24.4 Function Invocation Output

8.24.4.1 If the native parameter **DbHandle** is not NULL, then the output value of the parameter with the same name shall be the canonical representation of an integer in the range -2147483648 to 2147483647 (see 6.4.3). The integer shall be read from the variable of type **BioAPI_DB_HANDLE** pointed to by the native parameter **DbHandle**.

8.24.4.2 If the native parameter **Cursor** is not NULL, then the output value of the parameter with the same name shall be the canonical representation of an integer in the range 0 to 4294967295. The integer shall be read from the variable of type **BioAPI_DB_CURSOR** pointed to by the native parameter **Cursor**.

8.24.4.3 If the native parameter **DbHandle** or **Cursor** is NULL, then the output value of the parameter with the same name shall be an empty string.

8.25 BioSPI_DbClose

8.25.1 General

This function belongs to the BioSPI interface, and has the following native prototype:

```
BioAPI_RETURN BioAPI
BioSPI_DbClose(
   BioAPI_HANDLE ModuleHandle,
   BioAPI_DB_HANDLE DbHandle);
```

8.25.2 Function Invocation Scheme

This function has the following input parameters:

- **ModuleHandle**
- **DbHandle**

and no output parameters.

8.25.3 Function Invocation Input

8.25.3.1 The input value of **ModuleHandle** shall be a valid representation of an integer (see 6.4) in the range 0 to 4294967295.

8.25.3.2 The input value of **DbHandle** shall be a valid representation of an integer in the range -2147483648 to 2147483647.

8.25.3.3 The integer represented by the input value of **ModuleHandle** and **DbHandle** shall be assigned to the native parameter with the same name.

8.25.4 Function Invocation Output

There are no output parameters.

8.26 BioSPI_DbCreate

8.26.1 General

This function belongs to the BioSPI interface, and has the following native prototype:

```
BioAPI_RETURN BioAPI
BioSPI_DbCreate(
   BioAPI_HANDLE ModuleHandle,
   const uint8 *DbName,
   BioAPI_DB_ACCESS_TYPE AccessRequest,
   BioAPI_DB_HANDLE_PTR DbHandle);
```

8.26.2 Function Invocation Scheme

This function has the following input parameters:

- **ModuleHandle**
- **DbName**
- **ReadAccessRequest**
- **WriteAccessRequest**
- **no_DbHandle**

and the following output parameter:

- **DbHandle**

8.26.3 Function Invocation Input

8.26.3.1 The input value of **ModuleHandle** shall be a valid representation of an integer (see 6.4) in the range 0 to 4294967295.

8.26.3.2 The input value of **DbName** shall be a character string.

8.26.3.3 The input value of **ReadAccessRequest** and **WriteAccessRequest** shall be either a valid representation of a boolean (see 6.5) or an empty string.

8.26.3.4 The input value of **no_DbHandle** shall be either a valid representation of a boolean (see 6.5) or an empty string.

8.26.3.5 The integer represented by the input value of **ModuleHandle** shall be assigned to the native parameter with the same name.

8.26.3.6 The input value of **DbName** shall be written to a memory block of sufficient size, whose address shall be assigned to the native parameter **DbName.**

8.26.3.7 Two integer values, say V1 and V2, shall be set from the input values of **ReadAccessRequest** and **WriteAccessRequest** as follows:

a) if the input value of **ReadAccessRequest** is "**true**", then V1 shall be one, otherwise it shall be zero;

b) if the input value of **WriteAccessRequest** is "**true**", then V2 shall be one, otherwise it shall be zero.

8.26.3.8 The sum of the values V1 and V2 shall be assigned to the native parameter **AccessRequest**.

8.26.3.9 If the input value of **no_DbHandle** is "**true**", then the native parameter **DbHandle** shall be set to NULL, otherwise it shall be set to the address of a variable of type **BioAPI_DB_HANDLE**.

8.26.4 Function Invocation Output

8.26.4.1 If the native parameter **DbHandle** is not NULL, then the output value of the parameter with the same name shall be the canonical representation of an integer in the range -2147483648 to 2147483647 (see 6.4.3). The integer shall be read from the variable of type **BioAPI_DB_HANDLE** pointed to by the native parameter **DbHandle**.

8.26.4.2 If the native parameter **DbHandle** is NULL, then the output value of the parameter with the same name shall be an empty string.

8.27 BioSPI_DbDelete

8.27.1 General

This function belongs to the BioSPI interface, and has the following native prototype:

```
BioAPI_RETURN BioAPI
BioSPI_DbDelete(
   BioAPI_HANDLE ModuleHandle,
```

```
   const uint8 *DbName);
```

8.27.2 Function Invocation Scheme

This function has the following input parameters:

- **ModuleHandle**
- **DbName**

and no output parameters.

8.27.3 Function Invocation Input

8.27.3.1 The input value of **ModuleHandle** shall be a valid representation of an integer (see 6.4) in the range 0 to 4294967295.

8.27.3.2 The input value of **DbName** shall be a character string.

8.27.3.3 The integer represented by the input value of **ModuleHandle** shall be assigned to the native parameter with the same name.

8.27.3.4 The input value of **DbName** shall be written to a memory block of sufficient size, whose address shall be assigned to the native parameter **DbName.**

8.27.4 Function Invocation Output

There are no output parameters.

8.28 BioSPI_DbSetCursor

8.28.1 General

This function belongs to the BioSPI interface, and has the following native prototype:

```
BioAPI_RETURN BioAPI
BioSPI_DbSetCursor(
   BioAPI_HANDLE ModuleHandle,
   BioAPI_DB_HANDLE DbHandle,
   const BioAPI_UUID *KeyValue,
   BioAPI_DB_CURSOR_PTR Cursor);
```

8.28.2 Function Invocation Scheme

This function has the following input parameters:

- **ModuleHandle**
- **DbHandle**
- **KeyValue**
- **no_Cursor**

and the following output parameter:

- **Cursor**

8.28.3 Function Invocation Input

8.28.3.1 The input value of **ModuleHandle** shall be a valid representation of an integer (see 6.4) in the range 0 to 4294967295.

8.28.3.2 The input value of **DbHandle** shall be a valid representation of an integer in the range -2147483648 to 2147483647.

8.28.3.3 The input value of **KeyValue** shall be a valid representation of a UUID (see 6.6).

8.28.3.4 The input value of **no_Cursor** shall be either a valid representation of a boolean (see 6.5) or an empty string.

8.28.3.5 The integer represented by the input value of **ModuleHandle** and **DbHandle** shall be assigned to the native parameter with the same name.

8.28.3.6 The UUID represented by the input value of **KeyValue** shall be written to a variable of type **BioAPI_UUID**, whose address shall be assigned to the native parameter **KeyValue**.

8.28.3.7 If the input value of **no_Cursor** is "**true**", then the native parameter **Cursor** shall be set to NULL, otherwise it shall be set to the address of a variable of type **BioAPI_DB_CURSOR**.

8.28.4 Function Invocation Output

8.28.4.1 If the native parameter **Cursor** is not NULL, then the output value of the parameter with the same name shall be the canonical representation of an integer in the range 0 to 4294967295 (see 6.4.3). The integer shall be read from the variable of type **BioAPI_DB_CURSOR** pointed to by the native parameter **Cursor**.

8.28.4.2 If the native parameter **Cursor** is NULL, then the output value of the parameter with the same name shall be an empty string.

8.29 BioSPI_DbFreeCursor

8.29.1 General

This function belongs to the BioSPI interface, and has the following native prototype:

```
BioAPI_RETURN BioAPI
BioSPI_DbFree Cursor(
   BioAPI_HANDLE ModuleHandle,
   BioAPI_DB_CURSOR_PTR Cursor);
```

8.29.2 Function Invocation Scheme

This function has the following input parameters:

- **ModuleHandle**
- **Cursor**

and no output parameters.

8.29.3 Function Invocation Input

8.29.3.1 The input value of **ModuleHandle** shall be a valid representation of an integer (see 6.4) in the range 0 to 4294967295.

8.29.3.2 The input value of **Cursor** shall be a valid representation of an integer in the range 0 to 4294967295.

8.29.3.3 The integer represented by the input value of **ModuleHandle** shall be assigned to the native parameter with the same name.

8.29.3.4 The integer represented by the input value of **Cursor** shall be written to a variable of type **BioAPI_DB_CURSOR**, whose address shall be assigned to the native parameter **Cursor**.

8.29.4 Function Invocation Output

There are no output parameters.

8.30 BioSPI_DbStoreBIR

8.30.1 General

This function belongs to the BioSPI interface, and has the following native prototype:

```
BioAPI_RETURN BioAPI
BioSPI_DbStoreBIR(
   BioAPI_HANDLE ModuleHandle,
   const BioAPI_INPUT_BIR *BIRToStore,
   BioAPI_DB_HANDLE DbHandle,
   BioAPI_UUID_PTR Uuid);
```

8.30.2 Function Invocation Scheme

This function has the following input parameters:

- **ModuleHandle**
- the members of the parameter group "Input BIR" (see 9.2.6), with their names prefixed by "**BIRToStore_**"
- **DBHandle**
- **no_Uuid**

and the following output parameter:

- **Uuid**

8.30.3 Function Invocation Input

8.30.3.1 The input value of **ModuleHandle** shall be a valid representation of an integer (see 6.4) in the range 0 to 4294967295.

8.30.3.2 The input value of **DbHandle** shall be a valid representation of an integer in the range -2147483648 to 2147483647.

8.30.3.3 The input value of **no_Uuid** shall be either a valid representation of a boolean (see 6.5) or an empty string.

8.30.3.4 The integer represented by the input value of **ModuleHandle** and **DbHandle** shall be assigned to the native parameter with the same name.

8.30.3.5 The value of type **BioAPI_INPUT_BIR** represented (see 8.2.6.6) by the input value of the parameter group "Input BIR" shall be written to a variable of type **BioAPI_INPUT_BIR**, whose address shall be assigned to the native parameter **BIRToStore**.

8.30.3.6 If the input value of **no_Uuid** is "**true**", then the native parameter **Uuid** shall be set to NULL, otherwise it shall be set to the address of a variable of type **BioAPI_UUID**.

8.30.4 Function Invocation Output

8.30.4.1 If the native parameter **Uuid** is not NULL, then the output value of the parameter with the same name shall be the canonical representation of a UUID (see 6.6.3). The UUID shall be read from the variable of type **BioAPI_UUID** pointed to by the native parameter **BioAPI_UUID**.

8.30.4.2 If the native parameter **Uuid** is NULL, then the output value of the parameter with the same name shall be an empty string.

8.31 BioSPI_DbGetBIR

8.31.1 General

This function belongs to the BioSPI interface, and has the following native prototype:

```
BioAPI_RETURN BioAPI
BioSPI_DbGetBIR(
   BioAPI_HANDLE ModuleHandle,
   BioAPI_DB_HANDLE DbHandle,
   const BioAPI_UUID *KeyValue,
   BioAPI_BIR_HANDLE_PTR RetrievedBIR,
   BioAPI_DB_CURSOR_PTR Cursor);
```

8.31.2 Function Invocation Scheme

This function has the following input parameters:

- **ModuleHandle**
- **DbHandle**
- **KeyValue**
- **no_RetrievedBIR**
- **no_Cursor**

and the following output parameters:

- **RetrievedBIR**
- **Cursor**

8.31.3 Function Invocation Input

8.31.3.1 The input value of **ModuleHandle** shall be a valid representation of an integer (see 6.4) in the range 0 to 4294967295.

8.31.3.2 The input value of **DbHandle** shall be a valid representation of an integer in the range -2147483648 to 2147483647.

8.31.3.3 The input value of **KeyValue** shall be a valid representation of a UUID (see 6.6).

8.31.3.4 The input value of **no_RetrievedBIR** and **no_Cursor** shall be either a valid representation of a boolean (see 6.5) or an empty string.

8.31.3.5 The integer represented by the input value of **ModuleHandle** and **DbHandle** shall be assigned to the native parameter with the same name.

8.31.3.6 The UUID represented by the input value of **KeyValue** shall be written to a variable of type **BioAPI_UUID**, whose address shall be assigned to the native parameter **KeyValue**.

8.31.3.7 If the input value of **no_RetrievedBIR** is "**true**", then the native parameter **RetrievedBIR** shall be set to NULL, otherwise it shall be set to the address of a variable of type **BioAPI_BIR_HANDLE**.

8.31.3.8 If the input value of **no_Cursor** is "**true**", then the native parameter **Cursor** shall be set to NULL, otherwise it shall be set to the address of a variable of type **BioAPI_DB_CURSOR**.

8.31.4 Function Invocation Output

8.31.4.1 If the native parameter **RetrievedBIR** is not NULL, then the output value of the parameter with the same name shall be the canonical representation of an integer in the range -2147483648 to 2147483647 (see 6.4.3). The integer shall be read from the variable of type **BioAPI_BIR_HANDLE** pointed to by the native parameter **RetrievedBIR**.

8.31.4.2 If the native parameter **Cursor** is not NULL, then the output value of the parameter with the same name shall be the canonical representation of an integer in the range 0 to 4294967295. The integer shall be read from the variable of type **BioAPI_DB_CURSOR** pointed to by the native parameter **Cursor**.

8.31.4.3 If the native parameter **RetrievedBIR** or **Cursor** is NULL, then the output value of the parameter with the same name shall be an empty string.

8.32 BioSPI_DbGetNextBIR

8.32.1 General

This function belongs to the BioSPI interface, and has the following native prototype:

```
BioAPI_RETURN BioAPI
BioSPI_DbGetNextBIR(
   BioAPI_HANDLE ModuleHandle,
   BioAPI_DB_CURSOR_PTR Cursor,
   BioAPI_BIR_HANDLE_PTR RetrievedBIR,
   BioAPI_UUID_PTR Uuid);
```

8.32.2 Function Invocation Scheme

This function has the following input parameters:

- **ModuleHandle**

- **Cursor**
- **no_RetrievedBIR**
- **no_Uuid**

and the following output parameters:

- **RetrievedBIR**
- **Uuid**
- **out_Cursor**

8.32.3 Function Invocation Input

8.32.3.1 The input value of **ModuleHandle** shall be a valid representation of an integer (see 6.4) in the range 0 to 4294967295.

8.32.3.2 The input value of **Cursor** shall be a valid representation of an integer in the range 0 to 4294967295, or an empty string.

8.32.3.3 The input value of **no_RetrievedBIR** and **no_Uuid** shall be either a valid representation of a boolean (see 6.5) or an empty string.

8.32.3.4 The integer represented by the input value of **ModuleHandle** shall be assigned to the native parameter with the same name.

8.32.3.5 If the input value of **Cursor** is not an empty string, then the integer it represents shall be written to a variable of type **BioAPI_DB_CURSOR**, whose address shall be assigned to the native parameter **Cursor**. Otherwise, the native parameter **Cursor** shall be set to NULL.

8.32.3.6 If the input value of **no_RetrievedBIR** is "**true**", then the native parameter **RetrievedBIR** shall be set to NULL, otherwise it shall be set to the address of a variable of type **BioAPI_BIR_HANDLE**.

8.32.3.7 If the input value of **no_Uuid** is "**true**", then the native parameter **Uuid** shall be set to NULL, otherwise it shall be set to the address of a variable of type **BioAPI_UUID**.

8.32.4 Function Invocation Output

8.32.4.1 If the native parameter **RetrievedBIR** is not NULL, then the output value of the parameter with the same name shall be the canonical representation of an integer in the range -2147483648 to 2147483647 (see 6.4.3). The integer shall be read from the variable of type **BioAPI_BIR_HANDLE** pointed to by the native parameter **RetrievedBIR**.

8.32.4.2 If the native parameter **Uuid** is not NULL, then the output value of the parameter with the same name shall be the canonical representation of a UUID (see 6.6.3). The UUID shall be read from the variable of type **BioAPI_UUID** pointed to by the native parameter **Uuid**.

8.32.4.3 If the native parameter **Cursor** is not NULL, then the output value of the parameter **out_Cursor** shall be the canonical representation of an integer. The integer shall be read from the variable of type **BioAPI_DB_CURSOR** pointed to by the native parameter **Cursor**.

8.32.4.4 If the native parameter **RetrievedBIR** or **Uuid** is NULL, then the output value of the parameter with the same name shall be an empty string.

8.32.4.5 If the native parameter **Cursor** is NULL, then the output value of the parameter **out_Cursor** shall be an empty string.

8.33 BioSPI_DbQueryBIR

8.33.1 General

This function belongs to the BioSPI interface, and has the following native prototype:

```
BioAPI_RETURN BioAPI
BioSPI_DbQueryBIR(
   BioAPI_HANDLE ModuleHandle,
   BioAPI_DB_HANDLE DbHandle,
   const BioAPI_INPUT_BIR *BIRToQuery,
   BioAPI_UUID_PTR Uuid);
```

8.33.2 Function Invocation Scheme

This function has the following input parameters:

- **ModuleHandle**
- **DbHandle**
- the members of the parameter group "Input BIR" (see 9.2.6), with their names prefixed by "**BIRToQuery_**"
- **no_Uuid**

and the following output parameter:

- **Uuid**

8.33.3 Function Invocation Input

8.33.3.1 The input value of **ModuleHandle** shall be a valid representation of an integer (see 6.4) in the range 0 to 4294967295.

8.33.3.2 The input value of **DbHandle** shall be a valid representation of an integer in the range -2147483648 to 2147483647.

8.33.3.3 The input value of **no_Uuid** shall be either a valid representation of a boolean (see 6.5) or an empty string.

8.33.3.4 The integer represented by the input value of **ModuleHandle** and **DbHandle** shall be assigned to the native parameter with the same name.

8.33.3.5 The value of type **BioAPI_INPUT_BIR** represented (see 8.2.6.6) by the input value of the parameter group "Input BIR" shall be written to a variable of type **BioAPI_INPUT_BIR**, whose address shall be assigned to the native parameter **BIRToQuery**.

8.33.3.6 If the input value of **no_Uuid** is "**true**", then the native parameter **Uuid** shall be set to NULL, otherwise it shall be set to the address of a variable of type **BioAPI_UUID**.

8.33.4 Function Invocation Output

8.33.4.1 If the native parameter **Uuid** is not NULL, then the output value of the parameter with the same name shall be the canonical representation of a UUID (see 6.6.3). The UUID shall be read from the variable of type **BioAPI_UUID** pointed to by the native parameter **Uuid**.

8.33.4.2 If the native parameter **Uuid** is NULL, then the output value of the parameter with the same name shall be an empty string.

8.34 BioSPI_DbDeleteBIR

8.34.1 General

This function belongs to the BioSPI interface, and has the following native prototype:

```
BioAPI_RETURN BioAPI
BioSPI_DbDeleteBIR(
   BioAPI_HANDLE ModuleHandle,
   BioAPI_DB_HANDLE DbHandle,
   const BioAPI_UUID *KeyValue);
```

8.34.2 Function Invocation Scheme

This function has the following input parameters:

- **ModuleHandle**
- **DbHandle**
- **KeyValue**

and no output parameters.

8.34.3 Function Invocation Input

8.34.3.1 The input value of **ModuleHandle** shall be a valid representation of an integer (see 6.4) in the range 0 to 4294967295.

8.34.3.2 The input value of **DbHandle** shall be a valid representation of an integer in the range -2147483648 to 2147483647.

8.34.3.3 The input value of **KeyValue** shall be a valid representation of a UUID (see 6.6).

8.34.3.4 The integer represented by the input value of **ModuleHandle** and **DbHandle** shall be assigned to the native parameter with the same name.

8.34.3.5 The UUID represented by the input value of **KeyValue** shall be written to a variable of type **BioAPI_UUID**, whose address shall be assigned to the native parameter **KeyValue**.

8.34.4 Function Invocation Output

There are no output parameters.

8.35 BioSPI_ModuleEventHandler

8.35.1 General

This function belongs to the framework callback interface, and has the following native prototype:

```
typedef BioAPI_RETURN (BioAPI *BioAPI_ModuleEventHandler) (
   const BioAPI_UUID *BSPUuid,
   void* BioAPINotifyCallbackCtx,
   BioAPI_DEVICE_ID DeviceID,
   uint32 Reserved,
   BioAPI_MODULE_EVENT EventType);
```

8.35.2 Bound Activity Parameters

An activity bound to this function (see 8.8) shall have the following input parameters:

- **BSPUuid**
- **BioAPINotifyCallbackCtx**
- **DeviceID**
- **Reserved**
- **EventType**

and the following ouput parameter:

- **return**

8.35.3 Bound Activity Invocation Input

8.35.3.1 The input value of **BSPUuid** shall be the canonical representation of a UUID (see 6.6.3), read from the native parameter **BioAPI_UUID**.

8.35.3.2 The input value of **BioAPINotifyCallbackCtx**, **DeviceID**, **Reserved**, and **EventType** shall be the canonical representation of an integer in the range 0 to 4294967295 (see 6.4.3), read from the native parameter with the same name.

8.35.4 Bound Activity Invocation Output

The output value of **return** shall be a valid representation of an integer (see 6.4) in the range 0 to 4294967295. That integer shall be returned by the native function.

8.35.5 Default Output

If there is no activity bound to this function, then the native function shall return 0.

8.36 BioSPI_GUI_STATE_CALLBACK

8.36.1 General

This function belongs to the framework callback interface, and has the following native prototype:

```
typedef BioAPI_RETURN (BioAPI *BioAPI_GUI_STATE_CALLBACK) (
   void *GuiStateCallbackCtx,
   BioAPI_GUI_STATE GuiState,
```

```
BioAPI_GUI_RESPONSE *Response,
BioAPI_GUI_MESSAGE Message,
BioAPI_GUI_PROGRESS Progress,
BioAPI_GUI_BITMAP_PTR SampleBuffer);
```

8.36.2 Bound Activity Parameters

An activity bound to this function (see 8.8) shall have the following input parameters:

- **GuiStateCallbackCtx**
- the members of the parameter group "GUI state" (see 9.2.9)
- **Message**
- **Progress**
- **BitmapWidth**
- **BitmapHeight**
- **Bitmap**

and the following output parameters:

- **Response**
- **return**

8.36.3 Bound Activity Invocation Input

8.36.3.1 The input value of **GuiStateCallbackCtx** and **Message** shall be the canonical representation of an integer in the range 0 to 4294967295 (see 6.4.3), read from the native parameter with the same name.

8.36.3.2 The input value of the parameter group "GUI state" shall be the canonical representation (see 8.2.9.5) of a value of type **BioAPI_GUI_STATE**, read from the native parameter **GuiState**.

8.36.3.3 The input value of **Progress** shall be the canonical representation of an integer in the range 0 to 255, read from the native parameter with the same name.

8.36.3.4 The input value of **BitmapWidth** and **BitmapHeight** shall be the canonical representation of an integer in the range 0 to 4294967295, read from the field **Width** or **Height** (respectively) of the variable of type **BioAPI_GUI_BITMAP** pointed to by the native parameter **SampleBuffer**.

8.36.3.5 The input value of **Bitmap** shall be the canonical representation of a binary data block (see 7.7), read from a memory block whose address is in the field **Data** of the variable of type **BioAPI_DATA** pointed to by the field **Bitmap** of the variable of type **BioAPI_GUI_BITMAP** pointed to by the native parameter **SampleBuffer**, and whose length is in the field **Length** of that variable of type **BioAPI_DATA**.

8.36.4 Bound Activity Invocation Output

8.36.4.1 The output value of **Response** shall be a valid representation of an integer (see 6.4) in the range 0 to 255. That integer shall be written to a variable of type **BioAPI_GUI_RESPONSE**, whose address shall be assigned to the native parameter **Response**.

8.36.4.2 The output value of **`return`** shall be a valid representation of an integer (see 6.4) in the range 0 to 4294967295. That integer shall be returned by the native function.

8.36.5 Default Output

If there is no activity bound to this function, then the output value of **`Response`** shall be "3" and the native function shall return 0.

8.37 BioSPI_GUI_STREAMING_CALLBACK

8.37.1 General

This function belongs to the framework callback interface, and has the following native prototype:

```
typedef BioAPI_RETURN (BioAPI *BioAPI_GUI_STREAMING_CALLBACK) (
   void *GuiStreamingCallbackCtx,
   BioAPI_GUI_BITMAP_PTR Bitmap);
```

8.37.2 Bound Activity Parameters

An activity bound to this function (see 8.8) shall have the following input parameters:

- **`GuiStreamingCallbackCtx`**
- **`BitmapWidth`**
- **`BitmapHeight`**
- **`Bitmap`**

and the following output parameter:

- **`return`**

8.37.3 Bound Activity Invocation Inputf

8.37.3.1 The input value of **`GuiStateCallbackCtx`** shall be the canonical representation of an integer in the range 0 to 4294967295 (see 6.4.3), read from the native parameter with the same name.

8.37.3.2 The input value of **`BitmapWidth`** and **`BitmapHeight`** shall be the canonical representation of an integer in the range 0 to 4294967295, read from the field **`Width`** or **`Height`** (respectively) of the variable of type **`BioAPI_GUI_BITMAP`** pointed to by the native parameter **`SampleBuffer`**.

8.37.3.3 The input value of **`Bitmap`** shall be the canonical representation of a binary data block (see 7.7), read from a memory block whose address is in the field **`Data`** of the variable of type **`BioAPI_DATA`** pointed to by the field **`Bitmap`** of the variable of type **`BioAPI_GUI_BITMAP`** pointed to by the native parameter **`SampleBuffer`**, and whose length is in the field **`Length`** of that variable of type **`BioAPI_DATA`**.

8.37.4 Bound Activity Invocation Output

The output value of **`return`** shall be a valid representation of an integer (see 6.4) in the range 0 to 4294967295. That integer shall be returned by the native function.

8.37.5 Default Output

If there is no activity bound to this function, then the native function shall return 0.

8.38 BioSPI_STREAM_CALLBACK

8.38.1 General

This function belongs to the framework callback interface, and has the following native prototype:

```
typedef BioAPI_RETURN (BioAPI *BioAPI_STREAM_CALLBACK) (
   void *StreamCallbackCtx,
   BioAPI_DATA_PTR OutMessage,
   BioAPI_DATA_PTR InMessage);
```

8.38.2 Bound Activity Parameters

An activity bound to this function (see 8.8) shall have the following input parameters:

- **`StreamCallbackCtx`**
- **`no_OutMessage`**
- **`InMessage`**

and the following output parameters:

- **`OutMessage`**
- **`return`**

8.38.3 Bound Activity Invocation Input

8.38.3.1 The input value of **`StreamCallbackCtx`** shall be the canonical representation of an integer in the range 0 to 4294967295 (see 6.4.3), read from the native parameter with the same name.

8.38.3.2 The input value of **`no_OutMessage`** shall be "**`true`**" if the native parameter is NULL, and shall be "**`false`**" otherwise.

8.38.3.3 The input value of **`InMessage`** shall be the canonical representation of a binary data block (see 6.7.2), read from the memory block whose address is in the field **`Data`** of the variable of type **`BioAPI_DATA`** pointed to by the native parameter **`InMessage`**, and whose length is in the field **`Length`** of that variable.

8.38.4 Bound Activity Invocation Output

8.38.4.1 The output value of **`OutMessage`** shall be a valid representation of a binary data block (see 7.7). That binary data block shall be written to a memory block of sufficient size. The address of that memory block shall be written to the field **`Data`** of a variable of type **`BioAPI_DATA`**, and the length (in octets) of the binary data block shall be written to the field **`Length`** of that variable. The address of that variable shall be written to the native parameter **`OutMessage`**.

8.38.4.2 The output value of **`return`** shall be a valid representation of an integer (see 6.4) in the range 0 to 4294967295. That integer shall be returned by the native function.

8.38.5 Default Output

If there is no activity bound to this function, then the output value of **`OutMessage`** shall be an empty string (representing an empty memory block) and the native function shall return 0.

9 Predefined Variables

9.1 Variables Whose Value Never Changes

The following built-in variables have constant values.

```
__BioAPI_BIR_DATA_TYPE_RAW          "1"
__BioAPI_BIR_DATA_TYPE_INTERMEDIATE "2"
__BioAPI_BIR_DATA_TYPE_PROCESSED    "4"

__BioAPI_INVALID_BIR_HANDLE         "-1"
__BioAPI_UNSUPPORTED_BIR_HANDLE     "-2"

__BioAPI_PURPOSE_VERIFY                                "1"
__BioAPI_PURPOSE_IDENTIFY                              "2"
__BioAPI_PURPOSE_ENROLL                                "3"
__BioAPI_PURPOSE_ENROLL_FOR_VERIFICATION_ONLY          "4"
__BioAPI_PURPOSE_ENROLL_FOR_IDENTIFICATION_ONLY        "5"
__BioAPI_PURPOSE_AUDIT                                 "6"

__BioAPI_FALSE    "0"
__BioAPI_TRUE     "1"

__BioAPI_DB_INVALID_HANDLE    "-1"

__BioAPI_NOT_SET              "-1"
__BioAPI_NOT_SUPPORTED        "-2"

__BioAPI_CAPTURE_SAMPLE       "1"
__BioAPI_CANCEL               "2"
__BioAPI_CONTINUE             "3"
__BioAPI_VALID_SAMPLE         "4"
__BioAPI_INVALID_SAMPLE       "5"

__BioAPI_DB_TYPE              "1"
__BioAPI_ARRAY_TYPE           "2"

__BioAPI_DATABASE_ID_INPUT    "1"
__BioAPI_BIR_HANDLE_INPUT     "2"
```

```
__BioAPI_FULLBIR_INPUT        "3"

__BioAPI_NOTIFY_INSERT              "1"
__BioAPI_NOTIFY_REMOVE              "2"
__BioAPI_NOTIFY_FAULT               "3"
__BioAPI_NOTIFY_SOURCE_PRESENT      "4"
__BioAPI_NOTIFY_SOURCE_REMOVED      "5"

__BioAPI_POWER_NORMAL   "1"
__BioAPI_POWER_DETECT   "2"
__BioAPI_POWER_SLEEP    "3"

__BioAPI_OK "0"

__BioAPIERR_H_FRAMEWORK_INVALID_MODULE_HANDLE         "257"
__BioAPIERR_H_FRAMEWORK_NOT_INITIALIZED               "258"
__BioAPIERR_H_FRAMEWORK_INTERNAL_ERROR                "1"
__BioAPIERR_H_FRAMEWORK_MEMORY_ERROR                  "2"
__BioAPIERR_H_FRAMEWORK_INVALID_POINTER               "4"
__BioAPIERR_H_FRAMEWORK_INVALID_INPUT_POINTER         "5"
__BioAPIERR_H_FRAMEWORK_INVALID_OUTPUT_POINTER        "6"
__BioAPIERR_H_FRAMEWORK_FUNCTION_NOT_IMPLEMENTED      "7"
__BioAPIERR_H_FRAMEWORK_OS_ACCESS_DENIED              "9"
__BioAPIERR_H_FRAMEWORK_FUNCTION_FAILED               "10"
__BioAPIERR_H_FRAMEWORK_INVALID_UUID                  "12"
__BioAPIERR_H_FRAMEWORK_INCOMPATIBLE_VERSION          "65"
__BioAPIERR_H_FRAMEWORK_MODULE_LOAD_FAILED            "278"
__BioAPIERR_H_FRAMEWORK_MODULE_UNLOAD_FAILED          "280"
__BioAPIERR_H_FRAMEWORK_LIB_REF_NOT_FOUND             "281"
__BioAPIERR_H_FRAMEWORK_INVALID_MODULE_FUNCTION_TABLE "282"
__BioAPIERR_H_FRAMEWORK_MODULE_NOT_LOADED             "286"
__BioAPIERR_H_FRAMEWORK_INVALID_DEVICE_ID             "287"

__BioAPIERR_BSP_INTERNAL_ERROR                  "4097"
__BioAPIERR_BSP_MEMORY_ERROR                    "4098"
__BioAPIERR_BSP_INVALID_POINTER                 "4100"
__BioAPIERR_BSP_INVALID_INPUT_POINTER           "4101"
```

```
__BioAPIERR_BSP_INVALID_OUTPUT_POINTER          "4102"
__BioAPIERR_BSP_FUNCTION_NOT_IMPLEMENTED        "4103"
__BioAPIERR_BSP_OS_ACCESS_DENIED                "4105"
__BioAPIERR_BSP_FUNCTION_FAILED                 "4106"
__BioAPIERR_BSP_INVALID_DATA                    "4166"
__BioAPIERR_BSP_INVALID_DB_HANDLE               "4170"
__BioAPIERR_BSP_UNABLE_TO_CAPTURE               "4353"
__BioAPIERR_BSP_TOO_MANY_HANDLES                "4354"
__BioAPIERR_BSP_TIMEOUT_EXPIRED                 "4355"
__BioAPIERR_BSP_INVALID_BIR                     "4356"
__BioAPIERR_BSP_BIR_SIGNATURE_FAILURE           "4357"
__BioAPIERR_BSP_UNABLE_TO_WRAP_PAYLOAD          "4358"
__BioAPIERR_BSP_NO_INPUT_BIRS                   "4360"
__BioAPIERR_BSP_UNSUPPORTED_FORMAT              "4361"
__BioAPIERR_BSP_UNABLE_TO_IMPORT                "4362"
__BioAPIERR_BSP_FUNCTION_NOT_SUPPORTED          "4364"
__BioAPIERR_BSP_INCONSISTENT_PURPOSE            "4365"
__BioAPIERR_BSP_BIR_NOT_FULLY_PROCESSED         "4366"
__BioAPIERR_BSP_PURPOSE_NOT_SUPPORTED           "4367"
__BioAPIERR_BSP_UNABLE_TO_OPEN_DATABASE         "4608"
__BioAPIERR_BSP_DATABASE_IS_LOCKED              "4609"
__BioAPIERR_BSP_DATABASE_DOES_NOT_EXIST         "4610"
__BioAPIERR_BSP_DATABASE_ALREADY_EXISTS         "4611"
__BioAPIERR_BSP_INVALID_DATABASE_NAME           "4612"
__BioAPIERR_BSP_RECORD_NOT_FOUND                "4613"
__BioAPIERR_BSP_CURSOR_IS_INVALID               "4614"
__BioAPIERR_BSP_DATABASE_IS_OPEN                "4615"
__BioAPIERR_BSP_INVALID_ACCESS_REQUEST          "4616"
__BioAPIERR_BSP_END_OF_DATABASE                 "4617"
__BioAPIERR_BSP_UNABLE_TO_CREATE_DATABASE       "4618"
__BioAPIERR_BSP_UNABLE_TO_CLOSE_DATABASE        "4619"
__BioAPIERR_BSP_UNABLE_TO_DELETE_DATABASE       "4620"
```

9.2 Variables Whose Value May Change

NOTE - Some of the following built-in variables are related to one another. Each variable in a set of related variables is updated as specified in 6.3.3.

9.2.1 __exit

The value of this variable shall be "**false**" when the primary activity starts execution, but shall become "**true**" when the testing component receive a request to stop execution, and shall remain "**true**" until the primary activity terminates.

NOTE - The mechanism by which the testing component receives such a request is implementation-specific.

9.2.2 __native_functions_called

The value of this variable shall be the canonical representation of an integer which is the total number of native function calls, corresponding to invocations of standard interface functions from any activity, that have been made since the primary activity started execution.

9.2.3 __native_functions_returned

The value of this variable shall be the canonical representation of an integer which is the total number of native function calls, corresponding to invocations of standard interface functions from any activity, that have returned (with any return value) since the primary activity started execution.

NOTE - At any time, the value of this variable is less than or equal to the value of the variable **__native_functions_called** (but see 6.3.3).

10 Test Log

10.1 The abstract test engine shall produce an XML file as its main output. The root element of this XML file shall be a **<conformance_test_log>** element (see 10.2). The XML file shall be valid according to the schema specified in Annex C.

10.2 The **<conformance_test_log>** element shall have the following attributes:

a) **date_time** (required) – a valid date and time value in a format compliant with ISO 8601;

b) **standard** (required) - the value of this attribute shall be "**BioAPI 1.1**"; and shall have a content consisting of the following (in order):

a) one **<TestingLaboratory>** element (see 10.3) - this element contains contact information about the testing organization;

b) one **<Vendor>** element (see 10.4) - this element contains contact information about the vendor of the implementation under test;

c) one **<Biometric_Product>** element (see 10.6) - this element contains unambiguous identification of the biometric product under test;

d) one **<CTS_ID>** element - this element contains a character string that identifies the conformance test suite;

e) one **<test_assertion>** element - this element contains the name of the assertion executed, the name of the package that contains the assertion, and the values of all input parameters of the assertion;

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

1) **<function>** (see 10.10) – this element represents an invocation of a standard interface function (including callbacks) occurred during the execution of the test;

2) **<inline_response>** (see 10.12) – this element represents an output produced by an <assert_condition> element during the execution of the test;

10.3 The **<TestingLaboratory>** element shall have no attributes, and shall have a content consisting of the following (in order):

a) one **<Name>** element - a character string;

b) one **<Address>** element (see 10.5) - a street address;

c) one **<Phone>** element - a character string;

d) an optional **<Email>** element - an email address; and

e) an optional **<Url>** element – a URL.

10.4 The **<Vendor>** element shall have the same attributes and content as the **<TestingLaboratory>** element.

10.5 The **<Address>** element shall have no attributes, and shall have a content consisting of the following (in order):

a) one **<Street>** element - a character string;

b) one **<City>** element - a character string;

c) one **<StateOrProvince>** element - a character string;

d) one **<ZipOrPostalCode>** element - a character string;

e) one **<Country>** element - a character string.

10.6 The **<BiometricProduct>** element shall have the following attributes:

a) **Name** (required) – a character string;

b) **SerialNo** (required) – a character string.

and shall have a content consisting of the following:

- one **<Description>** element - a character string.

10.7 The **<test_assertion>** element shall have no attributes, and shall have a content consisting of the following (in order):

a) one **<package_name>** element - a character string representing a universally unique identifier (see 6.6);

b) one **<assertion_name>** element - a character string;

c) one **<description>** element – a character string containing the description of the assertion; and

d) zero or more occurrences of the **<input>** element (see 10.8), each containing the name and value of an input parameter of the assertion.

10.8 The **<input>** element shall have the following attributes:

a) **name** (required) – a character string matching the "NCName" production in W3C XML Namespaces;

b) **value** (required) – a character string;

and shall have an empty content.

10.9 The **<output>** element shall have the same attributes and content as the **<input>** element.

10.10 The **<function>** element shall have the following attribute:

- **dir** (required) – one of the character strings "**incoming**", "**outgoing**", indicating the direction of the function invocation relative to the testing component;

and shall have a content consisting of the following (in order):

a) one **<name>** element - a character string;

b) zero or more occurrences of the **<input>** element (see 10.8), each containing the name and value of an input parameter of the function invocation;

c) zero or more occurrences of the **<ouput>** element (see 10.9), each containing the name and value of an output parameter of the function invocation;

d) one **<return>** element (see 10.11) - containing the return value of the function invocation.

10.11 The **<return>** element shall have the following attribute:

a) **value** (required) – a character string

and shall have an empty content.

10.12 The **<inline_response>** element shall have no attributes, and shall have a content consisting of the following (in order):

a) one **<asserted_condition>** element - a character string;

b) one **<conformance>** element - one of the character strings "**error**", "**pass**", "**fail**", "**undecided**".

11 Test Report

11.1 The results of the execution of one or more assertions against an implementation under test shall be reported in an XML file. The root element of this XML file shall be a **`<conformance_test_report>`** element (see 11.2).

11.2 The **`<conformance_test_report>`** element shall have no attributes, and shall have a content consisting of the following (in order):

a) one **`<Title>`** element - the title of the report (a character string);

b) one **`<TestingLaboratory>`** element (see 10.3) - contact information about the testing organization;

c) one **`<ReportID>`** element - a unique identification of the test report (a character string);

d) one **`<Pages>`** element - the total number of pages of the test report (a character string);

e) one **`<Vendor>`** element (see 10.4) - contact information about the vendor of the implementation under test;

f) one **`<Biometric_Product>`** element (see 10.6) - an unambiguous identification of the biometric product under test;

g) one **`<IUT_Description>`** element - a description of the implementation under test (a character string);

h) one **`<Date_of_test>`** element - the date in which the test was performed (a character string in a format compliant with ISO 8601);

i) one **`<CTS_ID>`** element - a character string that identifies the conformance test suite;

j) one **`<Test_Configuration>`** element (see 11.3) - information about the configuration of the test;

k) one **`<Date_of_issue>`** element - the date in which the test report was issued (a character string in a format compliant with ISO 8601);

l) one **`<Report_authorization>`** element (see 11.5) - information about the persons accepting responsibility for the content of the test report.

11.3 The **`<Test_Configuration>`** element shall have no attributes, and shall have a content consisting of the following (in order):

- one or more occurrences of the **`<Assertion>`** element (see 11.4).

11.4 The **`<Assertion>`** element shall have the following attributes:

a) **`name`** (required) – a character string matching the "NCName" production in W3C XML Namespaces;

b) **`package`** (required) – a character string representing a universally unique identifier (see 6.6);

c) **description** (optional) – a character string;

and shall have a content consisting of the following:

a) an optional **<description>** element;

b) zero or more occurrences of the **<input>** element (see 10.8), each containing the name and value of an input parameter of the assertion;

c) one **<Conformance>** element, containing the overall conformity response of the assertion, determined from all the **<conformance>** elements present in the test log of the assertion. The overall conformity response shall be:

1) "**error**" if there are one or more **<conformance>** elements containing "**error**" in the test log of the assertion; or else

2) "**fail**" if there are one or more **<conformance>** elements containing "**fail**" but none containing "**error**"; or else

3) "**undecided**" if there are one or more **<conformance>** elements containing "**undecided**" but none containing either "**error**" or "**fail**"; or else

4) "**pass**" if there are one or more **<conformance>** elements containing "**pass**" but none containing "**error**", "**fail**", or "**undecided**"; or else

5) "**error**" if there are no **<conformance>** elements in the test log of the assertion

d) an optional **<Condition_evaluated>** element, containing the condition within the **<assert_condition>** element.

11.5 The **<Report_authorization>** element shall have no attributes, and shall have a content consisting of the following (in order):

- one or more occurrences of the **<Tester>** element (see 11.6).

11.6 The **<Tester>** element shall have the following attributes:

a) **Name** (required) – the name of the person (a character string);

b) **Title** (required) – the title of the person (a character string).

12 Conformance Test Suite

12.1 General Concepts

12.1.1 A Conformance Test Suite (CTS) is a combination of executable test cases, associated test data, test procedure, and CTS documentation. Only the requirements specified in the BioAPI specification standard are testable. The CTS cannot require features that are optional in a standard, but it could include tests for those features.

12.1.2 A set of data records may need to be prepared prior to execution of the CTS. The data set should include a number of pre-defined BIR records that may contain both valid and invalid data. The data set may include records that are to be used for validation of error handling by the IUT, as required by specific test cases.

12.1.3 The test procedure describes how the execution of the CTS is to be done and the directions for the tester to follow. Additionally, the procedure should be detailed enough so that validation of a given IUT can be repeated with no change in test results.

12.1.4 The CTS documentation, at a minimum should include the following:

a) All information necessary to install and configure the CTS

b) Known limitations of the CTS and release-specific changes

c) System requirements necessary to execute the CTS

d) A description of any CTS trouble-clearing procedures]

12.1.5 The CTS shall be automated to the maximum extent possible.

12.2 Conformance Test Suite Structure

12.2.1 As described in this Standard, the CTS has a hierarchical structure, in which an important level is the test case. Each test case normally has a single test purpose, such as that of verifying that the IUT has a certain required capability or exhibits a certain required behavior. Each test case consists of the following:

a) A description of the test purpose (i.e., what is being tested – the conditions, requirements, or capabilities that are to be addressed by a particular test)

b) The pass/fail criteria for the test case

c) A reference to the section in the BioAPI specification standard from which the test case is derived

12.2.2 As further described in this Standard, each test case provides objective, reproducible, unambiguous, and accurate test results. Each test case is traceable back to a statement or statements in the BioAPI specification and is written so as to explicitly state the behavior of a conforming IUT. Each test case, when executed on the IUT, will generate a pass/fail condition. Within the CTS, test cases can be combined in test groups or scenarios, each test group consisting of one or more test cases, grouped in a logical order. Associated with each test group may be a

test group objective. Test cases may be further modularized by using named subdivisions called test steps.

12.2.3 Test outcome is the series of events that occurred during execution of a test case. It includes all input to and output from the IUT at the point of control and observation.

12.2.4 A foreseen test outcome is one that has been defined by test case, i.e., the events, that occurred during execution of the test case. A foreseen test outcome always results in the assignment of a test verdict to the test case.

12.2.5 Test verdict is one of the following:

a) Pass means that the observed test outcome gives evidence of conformity to the conformity requirement(s) on which the test purpose of the test case is focused.

b) Fail means that the observed test outcome either demonstrates non-conformity with respect to at least one of the conformity requirements on which the test purpose of the test case is focused, or contains at least one invalid test event.

c) Inconclusive or Undecided means that the observed test outcome is such that neither a pass nor a fail verdict can be given.

The following Figure 3 illustrates usage of the test verdicts using, as an example one of the test cases described later in this standard.

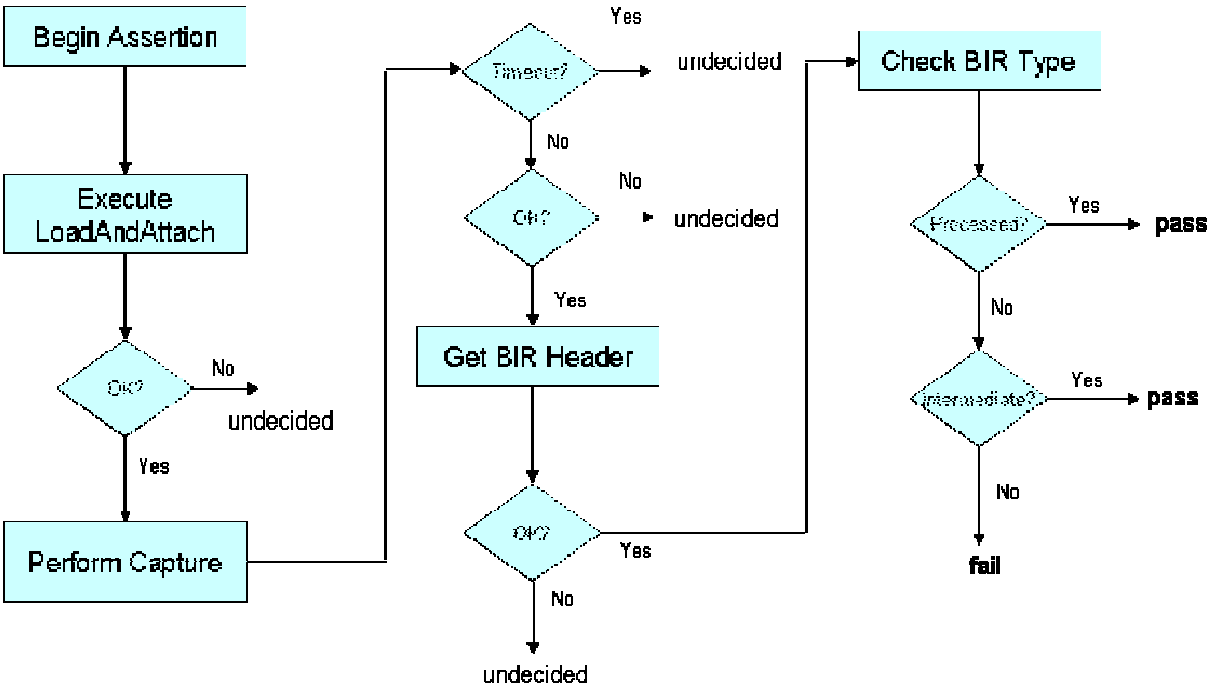

Figure 3. Sample test case flow

This test case has a test purpose of verifying that the captured BIR type is of either Processed or Intermediate type, which would result in returning the “Pass” test verdict. If

the BIR type is found to be of any other type (such as Raw), the “Fail” test verdict is returned. If any of the preceding steps included in this test case, as shown on the Figure 3, is unsuccessful, the “Undecided” test verdict is returned for the reason that the actual test purpose of this test case could not be verified.

12.2.6 An unforeseen test outcome is one that has not been identified by the test case; i.e., events that occurred during execution of the test case and did not match any sequence of the events defined in the test case. An unforeseen test outcome is the result in the recording of a test case error or an abnormal test case termination for the test case.

12.2.7 A test case error is recorded if an error is detected in either the test case itself or in its realization, i.e., an executable test case error.

12.2.8 An abnormal test case termination is recorded if the execution of the test case is prematurely terminated by the test system for reasons other than test case error.

12.2.9 The CTS is derived from a test purpose and specifies the following:

a) All sequences of test steps and foreseen test outcomes necessary to achieve the test purpose

b) The verdict to be assigned to each possible sequence of test events comprising a complete path through the test case

12.2.10 A standardized test notation is used for the specification of all assertions. The notation assumes that the internal structure or the source code of an IUT is not available to the tester, i.e., the conformance testing performed by the CTS is “black box” testing.

13 BioAPI Functions

The table below includes all the BioAPI functions and features.

Item	Question/Feature	Reference	Addressed	Not Addressed
1	BioAPI Registry	2.2	X	
2	BioSPI Module Load	3.3.1.1	X	
3	BioSPI Module Unload	3.3.1.2	X	
4	BioSPI Module Attach	3.3.1.3	X	
5	BioSPI Detach	3.3.1.4	X	
6	BioSPI Free BIR Handle	3.3.2.1	X	
7	BioSPI Get BIR From Handle	3.3.2.2	X	
8	BioSPI Get Header From Handle	3.3.2.3	X	
9	BioSPI Enable Events	3.3.3.1	X	
10	BioSPI Set GUI Callbacks	3.3.3.2		X
11	BioSPI Set Stream Callbacks	3.3.3.3		X
12	BioSPI Stream Input Output	3.3.3.4		X
13	**BioSPI Capture**	3.3.4.1	X	
13a	Return of raw/audit data	3.3.4.1	X	
13b	Return of quality in the captured BIR header	3.3.4.1 (2.1.46, 4.2.4.2)	X	
13c	BIR signing (by BSP)	1.5, 2.1.7	X	
13d	BIR encryption (by BSP)	1.5, 2.1.7	X	
13e	Detection of Source Presence	4.2.4.2		X
13f	Support of application control of the GUI	1.10, 4.2.4.2		X
14	**BioSPI Create Template**	3.3.4.2	X	
14a	Accept input of stored template to return update/adapted template	3.3.4.2	X	
14b	Acceptance of payload for inclusion of enrollment BIR	3.3.4.2	X	
14c	Return of quality in the processed BIR header	3.3.4.2 (2.1.46, 4.2.4.2)	X	
14d	BIR signing (by BSP)	1.5, 2.1.7	X	
14e	BIR encryption (by BSP)	1.5, 2.1.7	X	
15	**BioSPI Process**	3.3.4.3	X	
15a	Return of quality in the processed BIR header	3.3.4.3 (2.1.46, 4.2.4.2)	X	
15b	BIR signing (by BSP)	1.5, 2.1.7	X	
15c	BIR encryption (by BSP)	1.5, 2.1.7	X	
16	**BioSPI Verify Match**	3.3.4.4	X	
16a	Set threshold using FRR criteria	3.3.4.4		X
16b	Model/template adaptation	3.3.4.4	X	
16c	Return of continuous scores	1.7		Note E
16d	Return of achieved FRR score	3.3.4.4	X	
16e	Return of payload	3.3.4.4	X	
17	**BioSPI Identify Match**	3.3.4.5		X
17a	Set threshold using FRR criteria	3.3.4.5		X
17b	Return of continuous scores	1.7		Note E

Item	Question/Feature	Reference	Addressed	Not Addressed
17c	Return of achieved FRR score	3.3.4.5		X
17d	Support of binning	3.3.4.5		X
18	**BioSPI Enroll**	3.3.4.6	X	
18a	Template update	3.3.4.6	X	
18b	Acceptance of payload for inclusion of enrollment BIR	3.3.4.6	X?	
18c	Return of raw/audit data	3.3.4.6	X	
18d	Return of quality in the enrollment BIR header	3.3.4.6 (2.1.46, 4.2.4.2)	X	
18e	Support of application control of the GUI	1.10, 4.2.4.2		X
18f	BIR signing (by BSP)	1.5, 2.1.7	X	
18g	BIR encryption (by BSP)	1.5, 2.1.7	X	
18h	BSP client/server comms	1.6, 4.2.4.2		X
19	**BioSPI Verify**	3.3.4.7	X	
19a	Set threshold using FRR criteria	3.3.4.7		X
19b	Model/template adaptation	3.3.4.7	X	
19c	Return of continuous scores	1.7		Note E
19d	Return of achieved FRR score	3.3.4.7		X
19e	Return of payload	3.3.4.7	X	
19f	Return of raw/audit data	3.3.4.7	X	
19g	Support of application control of the GUI	1.10, 4.2.4.2		X
19h	BIR signing (by BSP)	1.5, 2.1.7	X	
19i	BIR encryption (by BSP)	1.5, 2.1.7	X	
19j	BSP client/server comms	1.6, 4.2.4.2		X
20	**BioSPI Identify**	3.3.4.8		X
20a	Set threshold using FRR criteria	3.3.4.8		X
20b	Return of continuous scores	1.7		X
20c	Return of achieved FRR score	3.3.4.8		X
20d	Support of binning	3.3.4.8		X
20e	Return of raw/audit data	3.3.4.8		X
20f	Support of application control of the GUI	1.10, 4.2.4.2		X
20g	BIR signing (by BSP)	1.5, 2.1.7		X
20h	BIR encryption (by BSP)	1.5, 2.1.7		X
20i	BSP client/server comms	1.6, 4.2.4.2		X
21	BioSPI Import	3.3.4.9	X	
22	BioSPI Set Power Mode	3.3.4.10		X
23	BioSPI db Open	3.3.5.1	X	
24	BioSPI db Close	3.3.5.2	X	
25	BioSPI db Create	3.3.5.3	X	
26	BioSPI db Delete	3.3.5.4	X	
27	BioSPI db Set Cursor	3.3.5.5	X	
28	BioSPI db Free Cursor	3.3.5.6	X	
29	BioSPI db Store BIR	3.3.5.7	X	
30	BioSPI db Get BIR	3.3.5.8	X	
31	BioSPI db Get Next BIR	3.3.5.9	X	
32	BioSPI db Query BIR	3.3.5.10	X	
33	BioSPI db Delete BIR	3.3.5.11	X	

Item	Question/Feature	Reference	Addressed	Not Addressed
Other conformance requirements				
101	BSP must implement all mandatory functions IAW SPI	A.2	X	
102	BSP accept all valid input param and return valid outputs	A.2	X	
103	Options implemented must be in accordance to spec (as shown in column 2 & 3 of Table)	A.2	X	
104	BSP provide all registry entries	A.2	X	
105	BSP has UUID according to data definition	A.2	X	
106	Conformant data structures –(Biometric data according to 2.1 & 3.2 including the BIR)	A.2	X	
107	Registered valid Format Owner and Format Type	A.2	X	
108	Error handling according to 2.3	A.2	X	
109	If GUI BSP must provide it	A.2	X	

General comment – if this is a mandatory requirements pre BioAPI conformity clause, they should be tested, and therefore should be included.

Notes:

a. BioAPI requirements apply to the biometric technology as implemented in a BioAPI compliant Biometric Service Provider (BSP) module.
b. DB functions only apply for 1) self contained devices, 2) BSP controlled databases, 2a) smart card (store on card), 2b) 1:N search/match engines.
c. See Appendix A, Section 4.2.4 of the BioAPI Specification for a description of optional capabilities.
d. Return of scores is not optional; however, the BSP has the option of returning continuous or stepwise/incremental scores. This is described but not specified by the BioAPI.

E. Return of continuous/incremental scores is not a testable parameter.

14 BioAPI Conformance Features And Test Assertions

Feature Num	Addressed	Assertion Name	*Reference*
1 BioAPI Registry			*1.11, 2.2, 4.2, 4.2.3*
1.1	Yes	**BioAPI_Registry_Installation**	
	Test Purpose	To test entries to the module registry on BSP installation.	
	Test Scenario	The BSP is not installed. The capabilities of the BSP, from the vendor's documentation, have been entered into a manifest. The BSP is then installed and the contents of the BSP Schema are compared with the manifest.	
	Expected Results	The BSP Schema matches the manifest. This assertion cannot be tested by calling BioSPI functions. The test lab will use some other means to determine if a BSP passes this assertion.	
1.2	Yes	**BioAPI_Registry_OpsSupported**	
	Test Purpose	To test that the BSP posts to the module registry whether it supports Local operation, Distribued operation, or both.	
	Test Scenario	The BSP is installed.	
	Expected Results	At least one of these three options of BioAPI_OPTIONS_MASK must be set, BioAPI_LOCAL_BSP, BioAPI_CLIENT_BSP or BioAPI_SERVER_BSP. This assertion cannot be tested by calling BioSPI functions. The test lab will use some other means to determine if a BSP passes this assertion.	
2 BioSPI Module Load			*3.3.1.1 (P. 97)*
2.1	Yes	**BioSPI_ModuleLoad_ValidParam**	
	Test Purpose	To test BioSPI_ModuleLoad with valid parameters.	
	Test Scenario	Call BioSPI_ModuleLoad with valid parameters.	
	Expected Results	Return code BioAPI_OK.	
2.2	Yes	**BioSPI_ModuleLoad_InvalidUUID**	
	Test Purpose	To test BioSPI_ModuleLoad with an invalid UUID.	
	Test Scenario	Call BioSPI_ModuleLoad with an invalid input parameter UUID.	
	Expected Results	Return code BioAPIERR_H_FRAMEWORK_INVALID_UUID.	
3 BioSPI Module Unload			*3.3.1.2*
3.1	Yes	**BioSPI_ModuleUnload_ValidParam**	
	Test Purpose	To test BioSPI_ModuleUnload with valid parameters.	
	Test Scenario	BioSPI_ModuleLoad succeeded.	
	Expected Results	Return code BioAPI_OK.	
3.2	Yes	**BioSPI_ModuleUnload_Unmatch**	
	Test Purpose	To test BioSPI_ModuleUnload unmatched with BioSPI_ModuleLoad.	
	Test Scenario	Initial state, or calling ModuleUnload twice.	
	Expected Results	Return code BioAPI_H_FRAMEWORK_MODULE_UNLOAD_FAILED.	

Feature Num	*Addressed*	*Assertion Name*	*Reference*
3.3	Yes	**BioSPI_ModuleUnload_InvalidUUID**	
	Test Purpose	To test BioSPI_ModuleUnload with an invalid UUID.	
	Test Scenario	BioSPI_ModuleLoad succeeded.	
	Expected Results	Return code BioAPI_H_FRAMEWORK_INVALID_UUID.	
3.4	Yes	**BioSPI_ModuleUnload_Confirm**	
	Test Purpose	To test that the BSP is truly unloaded.	
	Test Scenario	1) Call BioSPI_ModuleLoad with valid input parameters. The call is expected to succeed. 2) Call BioSPI_ModuleUnload with valid input parameters. 3) Call BioSPI_ModuleAttach with valid input parameters. It is expected to return BioAPIERR_H_FRAMEWORK_MODULE_NOT_LOADED.	
	Expected Results	Return code BioAPI_OK.	

4 BioSPI Module Attach *3.3.1.3*

4.1	Yes	**BioSPI_ModuleAttach_ValidParam**	
	Test Purpose	To test BioSPI_ModuleAttach with valid parameters.	
	Test Scenario	Call BioSPI_ModuleAttach with valid input parameters.	
	Expected Results	Return code BioAPI_OK.	
4.2	Yes	**BioSPI_ModuleAttach_InvalidUUID**	
	Test Purpose	To test BioSPI_ModuleAttach with an invalid UUID.	
	Test Scenario	Call BioSPI_ModuleAttach with an invalid UUID.	
	Expected Results	Return code BioAPI_H_FRAMEWORK_INVALID_UUID.	
4.3	Yes	**BioSPI_ModuleAttach_InvalidVersion**	
	Test Purpose	To test BioSPI_ModuleAttach with an invalid BioAPI Version.	
	Test Scenario	Call BioSPI_ModuleAttach with an invalid version number.	
	Expected Results	Return code other than BioAPI_OK.	
4.4	Yes	**BioSPI_ModuleAttach_InvalidModuleHandle**	
	Test Purpose	To test BioSPI_ModuleAttach with an invalid module handle.	
	Test Scenario	Call BioSPI_ModuleAttach with an invalid module handle.	
	Expected Results	Return code BioAPIERR_H_FRAMEWORK_INVALID_MODULE_HANDLE.	

5 BioSPI Module Detach *3.3.1.4*

5.1	Yes	**BioSPI_ModuleDetach_ValidParam**	
	Test Purpose	To test BioSPI_ModuleDetach with valid parameters.	
	Test Scenario	Call BioSPI_ModuleDetach with valid parameters.	
	Expected Results	Return code BioAPI_OK.	
5.2	Yes	**BioSPI_ModuleDetach_InvalidModuleHandle**	
	Test Purpose	To test BioAPI_ModuleDetach with an invalid module handle.	
	Test Scenario	Call BioSPI_ModuleDetach with an invalid module handle.	
	Expected Results	Return code BioAPIERR_H_FRAMEWORK_INVALID_MODULE_HANDLE.	

Feature			*Reference*
Num	*Addressed*	*Assertion Name*	
5.3	Yes	**BioSPI_ModuleDetach_Confirm**	
	Test Purpose	To test BioSPI_ModuelDetach truly terminates the attach session.	
	Test 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_Capture. This function is expected to fail with the error BioAPIERR_H_FRAMEWORK_MODULE_NOT_LOADED.	
	Expected Results	The call to BioSPI_Capture is expected to fail with the error BioAPIERR_H_FRAMEWORK_MODULE_NOT_LOADED.	

6 BioSPI Free BIR Handle *3.3.2.1*

6.1	Yes	**BioSPI_FreeBIRHandle_InvalidModuleHandle**	
	Test Purpose	To test BioSPI_FreeBIRHandle with an invalid module handle.	
	Test Scenario	Call BioSPI_FreeBIRHandle with an invalid module handle.	
	Expected Results	Return code other than BioAPI_OK.	
6.2	Yes	**BioSPI_FreeBIRHandle_InvalidBIRHandle**	
	Test Purpose	To test BioSPI_FreeBIRHandle with an invalid BIR handle.	
	Test Scenario	Call BioSPI_FreeBIRHandle with an invalid BIR handle.	
	Expected Results	Return code other than BioAPI_OK.	
6.3	Yes	**BioSPI_FreeBIRHandle_ValidParam**	
	Test Purpose	To test BioSPI_FreeBIRHandle with valid parameters.	
	Test Scenario	Call BioSPI_FreeBIRHandle with valid parameters.	
	Expected Results	Return code of BioAPI_OK.	

7 BioSPI Get BIR From Handle *3.3.2.2*

7.1	Yes	**BioSPI_GetBIRFromHandle_ValidParam**	
	Test Purpose	To test BioSPI_GetBIRFromHandle with valid parameters.	
	Test Scenario	Call BioSPI_GetBIRFromHandle with valid parameters.	
	Expected Results	1. Return code BioAPI_OK. 2. Valid BIR.	
7.2	Yes	**BioSPI_GetBIRFromHandle_InvalidModuleHandle**	
	Test Purpose	To test BioSPI_GetBIRFromHandle with an invalid module handle.	
	Test Scenario	Call BioSPI_GetBIRFromHandle with an invalid module handle.	
	Expected Results	Return code other than BioAPI_OK.	
7.3	Yes	**BioSPI_GetBIRFromHandle_InvalidBIRHandle**	
	Test Purpose	To test BioSPI_GetBIRFromHandle with an invalid BIR handle.	
	Test Scenario	Call BioSPI_GetBIRFromHandle with an invalid BIR handle.	
	Expected Results	Return code other than BioAPI_OK.	

Feature Num	Addressed	Assertion Name	Reference
8		**BioSPI Get Header from Handle**	3.3.2.3
8.1	Yes	**BioSPI_GetHeaderFromHandle_ValidParam**	
		Test Purpose To test BioSPI_GetHeaderFromHandle with valid parameters. ***Test Scenario*** Call BioSPI_GetHeaderFromHandle with valid parameters ***Expected Results*** 1. Return code BioAPI_OK. 2. Valid BIR header.	
8.2	Yes	**BioSPI_GetHeaderfromHandle_InvalidModuleHandle**	
		Test Purpose To test BioSPI_GetHeaderFromHandle with an invalid ModuleHandle. ***Test Scenario*** Call BioSPI_GetHeaderFromHandle with an invalid module handle. ***Expected Results*** Return code other than BioAPI_OK.	
8.3	Yes	**BioSPI_GetHeaderfromHandle_InvalidBIRHandle**	
		Test Purpose To test BioSPI_GetHeaderFromHandle with an invalid BIR Handle. ***Test Scenario*** Call BioSPI_GetHeaderFromHandle with an invalid BIR handle. ***Expected Results*** Return code other than BioAPI_OK.	
8.4	Yes	**BioSPI_GetHeaderFromHandle_ BIRHandleNotFreed**	
		Test Purpose To test the BIR handle is not freed after call BioSPI_GetHeaderFromHandle. ***Test 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. ***Expected Results*** The call to BioSPI_GetBIRFromHandle is expected to return BioAPI_OK.	
9		**BioSPI Enable Events**	3.3.3.1
9.1	Yes	**BioSPI_EnableEvents_ValidParam**	
		Test Purpose To test BioSPI_EnableEvents with valid parameters. ***Test Scenario*** Call BioSPI_EnableEvents with valid parameters. ***Expected Results*** Return code BioAPI_OK.	
9.2	Yes	**BioSPI_EnableEvents_InvalidModuleHandle**	
		Test Purpose To test BioSPI_EnableEvents with an invalid module handle. ***Test Scenario*** Call BioSPI_EnableEvents with an invalid module handle. ***Expected Results*** Return code other than BioAPI_OK.	
10		**BioSPI Set GUI Callbacks**	3.3.3.2
	No		
11		**BioSPI Set Stream Callbacks**	3.3.3.3
	No		

Feature Num	Addressed	Assertion Name	Reference
12		**BioSPI Stream Input Output**	*3.3.3.4*
	No		
13		**BioSPI Capture**	*3.3.4.1*
13.1	Yes	**BioSPI_Capture_InvalidModuleHandle**	
	Test Purpose	To test BioSPI_Capture with an invalid module handle.	
	Test Scenario	Call BioSPI_Capture with an invalid module handle.	
	Expected Results	Return code other than BioAPI_OK.	
13.2	Yes	**BioSPI_Capture_IntermediateProcessedBIR**	
	Test Purpose	To test that either an "intermediate" BIR or a "processed" BIR is returned for a specified process.	
	Test Scenario	A valid biometric is presented to BioSPI_Capture.	
	Expected Results	Return code BioAPI_OK and a valid "intermediate" or "processed" BIR.	
13.3	Yes	**BioSPI_Capture_PurposeInHeader**	
	Test Purpose	To test that Purpose is recorded in the header of the CapturedBIR.	
	Test Scenario	A valid biometric is presented to BioSPI_Capture.	
	Expected Results	Return code BioAPI_OK and Purpose in the header of the CapturedBIR.	
13.4	Yes	**BioSPI_Capture_Serialization**	
	Test Purpose	To test serialization.	
	Test Scenario	Two or more capture processes are started.	
	Expected Results	The capture results are expected to be returned to the proper capture process.	
13.5	Yes	**BioSPI_Capture_SerializationTimeout**	
	Test Purpose	To test serialization of capture with timeout.	
	Test Scenario	1. Start test. 2. Prompt "Start 2nd BioAPI_Capture Application (Long Timeout)" and pause test. 3. Start 2nd application, observe GUI. 4. Continue test (Timeout shorter than 2nd application) 5. Wait for timeout.	
	Expected Results	Return code BioAPIERR_BSP_TIMEOUT_EXPIRED.	
13.6	Yes	**BioSPI_Capture_Timeout**	
	Test Purpose	To test that BioAPIERR_BSP_TIMOUT_EXPIRED is returned.	
	Test Scenario	Call BioSPI_Capture_Timeout and allow it to timeout.	
	Expected Results	Return code BioAPIERR_BSP_TIMOUT_EXPIRED.	

Feature Num	Addressed	Assertion Name	Reference
13a		***Return of raw/audit data***	*3.3.4.1*
13a.1	Yes	**BioSPI_Capture_AuditData**	

Test Purpose To test the return of "raw" data if the BSP supports return of raw/audit data.
Test Scenario A valid biometric is presented to BioSPI_Capture.
Expected Results If the BSP supports AuditData, a handle to a "raw" BIR or a value of BioAPI_INVALID_BIR_HANDLE if no audit data is available. If AuditData is not supported, the BSP may return a handle value of BioAPI_UNSUPPORTED_BIR_HANDLE.

13b		***Return of quality in the captured BIR header***	*3.3.4.1, (2.1.46, 4.2.4.2)*
13b.1	Yes	**BioSPI_Capture_ReturnQuality**	

Test Purpose To test that the captured BIR contains a valid quality value (in the range 0-100).
Test Scenario Call BioSPI_Capture_ReturnQuality with valid parameters.
Expected Results The captured BIR contains a valid quality value (in the range 0-100).

13c		***BIR signing (by BSP)***	*1.5, 2.1.7*
13c.1	Yes	**BioSPI_Capture_BIRSigned**	

Test Purpose To test BIR signing if supported by the BSP.
Test Scenario If the BSP supports signing cal BioSPI_Capture with valid parameters. Present a valid biometric.
Expected Results Return code BioAPI_OK and a valid CapturedBIR that includes a signature.

13d		***BIR encryption (by BSP)***	*1.5, 2.1.7*
13d.1	Yes	**BioSPI_Capture_BIREncrypted**	

Test Purpose To test BIR encryption if supported by the BSP.
Test Scenario If the BSP supports BIR encryption call BioSPI_Enroll with valid parameters. Present a valid biometric.
Expected Results Return code BioAPI_OK and a valid CapturedBIR.

13e		***Detection of Source Presence***	*4.2.4.2*

No

13f		***Support of application control of the GUI***	*1.10, 4.2.4.2*

No

Feature Num	Addressed	Assertion Name	Reference
14		**BioSPI Create Template**	*3.3.4.2*
14.1	Yes	**BioSPI_CreateTemplate_OutputBIRPurpose**	
	Test Purpose	To test BioSPI_CreateTemplate with valid parameters.	
	Test Scenario	Call BioSPI_CreateTemplate with valid parameters.	
	Expected Results	Return code BioAPI_OK.	
14.2	Yes	**BioSPI_CreateTemplate_OutputBIRDataType**	
	Test Purpose	To test that BioSPI_CreateTemplate creates a processed BIR.	
	Test 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.	
	Expected Results	Return code BioAPI_OK.	
14.3	Yes	**BioSPI_CreateTemplate_InputBIRDataType**	
	Test Purpose	To test that BioSPI_CreateTemplate rejects an input BIR of a data type different from BioAPI_BIR_DATA_TYPE_INTERMEDIATE.	
	Test 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) Change the processed level of the BIR to either RAW or 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.	
	Expected Results	Return code other than BioAPI_OK.	
14.4	Yes	**BioSPI_CreateTemplate_Purpose**	
	Test Purpose	To test that the purpose of the template BIR is either "enroll_verify" and/or "enroll_identify".	
	Test 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) Change the propose to PURPOSE_VERIFY. 6) Call BioSPI_CreateTemplate specifying the input BIR. 7) Check if the return value is equal to __BioAPIERR_BSP_INCONSISTENT_PURPOSE. 8) Detach and unload the BSP under test.	
	Expected Results	Return code BioAPIERR_BSP_INCONSISTENT_PURPOSE.	

Feature			*Reference*
Num	*Addressed*	*Assertion Name*	
14a		***Accept input of stored template to return update/adapted template***	*3.3.4.2*
14a.1	Yes	**BioSPI_CreateTemplate_StoredTemplateUnchanged**	
		Test Purpose	If the BSP supports template adaptation, test that the StoredTemplate remains unchanged.
		Test Scenario	BioSPI_Create_Template is passed a valid CapturedBIR and StoredTemplate.
		Expected Results	Return code BioAPI_OK and an unchanged StoredTemplate.
14b		***Acceptance of payload for inclusion of enrollment BIR***	*3.3.4.2*
14b.1	Yes	**BioSPI_CreateTemplate_PayloadSupported**	
		Test Purpose	To test Payload if supported by the BSP.
		Test Scenario	If the BSP supports Payload, call BioSPI_CreateTemplate with valid parameters including Payload.
		Expected Results	Return code BioAPI_OK and a valid NewTemplate that includes the Payload.
14b.2	Yes	**BioSPI_CreateTemplate_PayloadNotCopied**	
		Test Purpose	If the BSP supports template adaptation and a BIR payload, test that the payload is not copied from StoredTemplate to NewTemplate.
		Test Scenario	BioSPI_Create_Template is passed a valid CapturedBIR and StoredTemplate. The StoredTemplate must contain a payload. The Payload parameter is NULL.
		Expected Results	Return code BioAPI_OK and a valid NewTemplate that does not contain a payload.
14c		***Return of quality in the processed BIR header***	*3.3.4.2, (2.1.46, 4.2.4.2)*
14c.1	Yes	**BioSPI_CreateTemplate_BIRHeaderQuality**	
		Test Purpose	To test the return of quality in the enrollment BIR header if supported by the BSP.
		Test Scenario	If the BSP supports return of quality, call BioSPI_CreateTemplate with valid parameters.
		Expected Results	Return code BioAPI_OK and a valid NewTemplate that includes the quality.
14d		***BIR signing (by BSP)***	*1.5, 2.1.7*
14d.1	Yes	**BioSPI_CreateTemplate_BIRSigned**	
		Test Purpose	To test BIR signing if supported by the BSP.
		Test Scenario	If the BSP supports signing cal BioSPI_CreateTemplate with valid parameters.
		Expected Results	Return code BioAPI_OK and a valid NewTemplate that includes a signature.
14e		***BIR encryption (by BSP)***	*1.5, 2.1.7*
14e.1	Yes	**BioSPI_CreateTemplate_BIREncrypted**	
		Test Purpose	To test BIR encryption if supported by the BSP.
		Test Scenario	If the BSP supports BIR encryption call BioSPI_CreateTemplatel with valid parameters.
		Expected Results	Return code BioAPI_OK and a valid NewTemplate.

Feature Num	Addressed	Assertion Name	Reference
15		**BioSPI Process**	*3.3.4.3*
15.1	Yes	**BioSPI_Process_ValidParam**	
	Test Purpose	If the attached BSP has processing capability, test that the BSP builds a ProcessedBIR from an "intermediate" CapturedBIR.	
	Test Scenario	BioSPI_Process is passed a valid "intermediate" CapturedBIR.	
	Expected Results	Return code BioAPI_OK and a valid ProcessedBIR.	
15.2	Yes	**BioSPI_Process_CannotProcess**	
	Test Purpose	If the attached BSP does not have processing capability, test that the BSP returns a NULL ProcessedBIR from an "intermediate" CapturedBIR.	
	Test Scenario	BioSPI_Process is passed a valid "intermediate" CapturedBIR.	
	Expected Results	Return code BioAPI_OK and ProcessedBIR is NULL.	
15.3	Yes	**BioSPI_Process_BuildsProcessedBIR**	
	Test Purpose	To test that the BIR returned by BioSPI_Process is processed.	
	Test 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.	
	Expected Results	If the BSP has processing capability, a processed BIR. Otherwise a NULL processed BIR is returned.	
15.4	Yes	**BioSPI_Process_InputBIRDataType**	
	Test Purpose	To test the BioSPI_Process with invalid input BIR data type.	
	Test 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) Change the processed level of the BIR to PROCESSED. 6) Call BioSPI_Process specifying the input BIR. 7) Check if the return value is different from BioAPI_OK. 8) Detach and Unload the BSP under test.	
	Expected Results	Return code BioAPI_OK.	
15.5	Yes	**BioSPI_Process_OutputBIRPurpose**	
	Test Purpose	To test BioSPI_Process with valid parameters and check that the returned purpose of the processed BIR is the same as the purpose of the captured BIR.	
	Test 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.	
	Expected Results	Return code BioAPI_OK.	

Feature			*Reference*
Num	*Addressed*	*Assertion Name*	
15a		***Return of quality in the processed BIR header***	*3.3.4.3, (2.1.46, 4.2.4.2)*
15a.1	Yes	**BioSPI_Process_BIRHeaderQuality**	

Test Purpose To test the return of quality in the ProcessedBIR header if supported by the BSP.

Test Scenario If the BSP supports return of quality, call BioSPI_Process with valid parameters.
Expected Results Return code BioAPI_OK and a valid ProcessedBIR that includes the quality in its header.

15b		***BIR signing (by BSP)***	*1.5, 2.1.7*
15b.1	Yes	**BioSPI_Process_BIRSigned**	

Test Purpose To test BIR signing if supported by the BSP.
Test Scenario If the BSP supports signing cal BioSPI_Process with valid parameters.
Expected Results Return code BioAPI_OK and a valid ProcessedBIR that includes a signature.

15c		***BIR encryption (by BSP)***	*1.5, 2.1.7*
15c.1	Yes	**BioSPI_Process_BIREncrypted**	

Test Purpose To test BIR encryption if supported by the BSP.
Test Scenario If the BSP supports BIR encryption call BioSPI_Process with valid parameters.
Expected Results Return code BioAPI_OK and a valid ProcessedBIR.

16		***BioSPI Verify Match***	*3.3.4.4*
16.1	Yes	**BioSPI_VerifyMatch_ValidParam**	

Test Purpose To test BioSPI_VerifyMatch with valid parameters.
Test Scenario Valid parameters are passed to BioSPI_VerifyMatch.
Expected Results Return code BioAPI_OK. Result can be either BioAPI_TRUE or BioAPI_FALSE. FARAchieved must be valid.

16.2 Yes **BioSPI_VerifyMatch_UnspecifiedFAR**

Test Purpose To test failure on unspecified FAR.
Test Scenario BioSPI_VerifyMatch is passed valid parameters except for MaxFARRequested.
Expected Results Return code other than BioAPI_OK.

16.3 Yes **BioSPI_VerifyMatch_Inconsistent_Purpose**

Test Purpose To test BioSPI_VerifyMatch
Test 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.

Expected Results Return code BioAPIERR_BSP_INCONSISTENT_PURPOSE.

Feature Num	Addressed	Assertion Name	Reference
16a	***Set threshold using FRR criteria***		*3.3.4.4*
	No		
16b	***Model/template adaptation***		*3.3.4.4*
16b.1	Yes	**BioSPI_VerifyMatch_Adaptation**	

Test Purpose To test Model/template adaptation if supported by the BSP.
Test Scenario If the BSP supports Model/template adaptation, call BioSPI_VerifyMatch with valid parameters.
Expected Results Return code BioAPI_OK and a valid AdaptedBIR.

16c	***Return of continuous scores***		*1.7*
	No		
16d	***Return of achieved FRR score***		*3.3.4.4*
16d.1	Yes	**BioSPI_VerifyMatch_AchievedFRR**	

Test Purpose To test return of achieved FRR score if supported by the BSP.
Test Scenario If the BSP supports the return of achieved FRR score, call BioSPI_VerifyMatch with valid parameters. ProcessedBIR and Stored template should be a "match."
Expected Results Return code BioAPI_OK and a valid FRRAchieved.

16e	***Return of payload***		*3.3.4.4*
16e.1	Yes	**BioSPI_VerifyMatch_Payload**	

Test Purpose To test Return of payload if supported by the BSP and verification succeeds.
Test Scenario If the BSP supports the return of payload, call BioSPI_VerifyMatch with valid parameters. ProcessedBIR and Stored template should be a "match."
Expected Results Return code BioAPI_OK, Result of BioAPI_TRUE and a valid Payload.

17	***BioSPI Identify Match***		*3.3.4.5*
	No		
17a	***Set threshold using FRR criteria***		*3.3.4.5*
	No		
17b	***Return of continuous scores***		*1.7*
	No		

Feature Num	Addressed	Assertion Name	Reference
17c		***Return of achieved FRR score***	*3.3.4.5*
	No		
17d		***Support of binning***	*3.3.4.5*
	No		
18		***BioSPI Enroll***	*3.3.4.6*
18.1	Yes	**BioSPI_Enroll_ValidParam**	
	Test Purpose	To test BioSPI_Enroll with valid parameters.	
	Test Scenario	BioSPI_Enroll is called with valid parameters and a valid biometric is presented.	
	Expected Results	Return code BioAPI_OK and NewTemplate is valid..	
18.2	Yes	**BioSPI_Enroll_InvalidPurpose**	
	Test Purpose	To test BioSPI_Enroll with an invalid Purpose.	
	Test Scenario	BioSPI_Enroll is called with valid parameters, except Purpose, which is invalid.. A valid biometric is presented.	
	Expected Results	Return code BioAPIERR_BSP_PURPOSE_NOT_SUPPORTED.	
18.3	Yes	**BioSPI_Enroll_Timeout**	
	Test Purpose	To test BioSPI_Enroll timeout feature.	
	Test Scenario	BioSPI_Enroll is called with valid parameters but is not presented with a biometric.	
	Expected Results	Return code BioAPIERR_BSP_TIMEOUT_EXPIRED.	
18.4	Yes	**BioSPI_Enroll_InvalidTimeout**	
	Test Purpose	To test failure on negative Timeout other than the value -1.	
	Test Scenario	BioSPI_Enroll is passed valid parameters except Timeout, with is a negative number other than -1.	
	Expected Results	Return code other than BioAPI_OK.	
18.5	Yes	**BioSPI_Enroll_Payload**	
	Test Purpose	To test that calling BioSPI_Enroll with a payload returns BioAPI_OK	
	Test 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.	
	Expected Results	Return code of BioAPI_OK.	

Feature			*Reference*
Num	*Addressed*	*Assertion Name*	

18a Template update — *3.3.4.6*

18a.1 Yes **BioSPI_Enroll_TemplateAdaptation**

Test Purpose	Test template adaptation if supported by the BSP.
Test Scenario	BioSPI_Enroll is passed valid parameters, including StoredTemplate.. A valid biometric is presented.
Expected Results	Return code BioAPI_OK and a valid NewTemplate.

18b Acceptance of payload for inclusion of enrollment BIR — *3.3.4.6*

18b.1 Yes **BioSPI_Enroll_PayloadTooLarge**

Test Purpose	To test failure on Payload too large if Payload supported by the BSP.
Test Scenario	BioSPI_Enroll is passed valid parameters except Payload, whose size exceeds the maximum payload size specified in the module registry.
Expected Results	Return code other than BioAPI_OK.

18c Return of raw/audit data — *3.3.4.6*

18c.1 Yes **BioSPI_Enroll_AuditData**

Test Purpose	To test AuditData.
Test Scenario	BioAPI_Enroll is passed valid parameters. A valid biometric is presented.
Expected Results	One of the following: 1. Return code BioAPI_OK and valid AuditData. 2. BioAPI_UNSUPPORTED_BIR_HANDLE if the BSP does not support audit data. 3. BioAPI_INVALID_BIR_HANDLE if the BSP supports audit data but no audit data is available.

18d Return of quality in the enrollment BIR header — *3.3.4.6, (2.1.46, 4.2.4.2)*

18d.1 Yes **BioSPI_Enroll_BIRHeaderQuality**

Test Purpose	To test the return of quality in the enrollment BIR header if supported by the BSP.
Test Scenario	If the BSP supports return of quality, call BioSPI_Enroll with valid parameters. Present a valid biometric.
Expected Results	Return code BioAPI_OK and a valid NewTemplate that includes the quality in its header.

18e Support of application control of the GUI — *1.10, 4.2.4.2*

No

Feature			*Reference*
Num	*Addressed*	*Assertion Name*	

18f BIR signing (by BSP) — *1.5, 2.1.7, 4.2.4.2*

18f.1 Yes **BioSPI_Enroll_BIRSigned**

Test Purpose	To test BIR signing if supported by the BSP.
Test Scenario	If the BSP supports signing cal BioSPI_Enroll with valid parameters. Present a valid biometric.
Expected Results	Return code BioAPI_OK and a valid NewTemplate that includes a signature.

18g BIR encryption (by BSP) — *1.5, 2.1.7*

18g.1 Yes **BioSPI_Enroll_BIREncrypted**

Test Purpose	To test BIR encryption if supported by the BSP.
Test Scenario	If the BSP supports BIR encryption call BioSPI_Enroll with valid parameters. Present a valid biometric.
Expected Results	Return code BioAPI_OK and a valid NewTemplate.

18h BSP client/server comms — *1.6, 4.2.4.2*

No

19 BioSPI Verify — *3.3.4.7*

19.1 Yes **BioSPI_Verify_ValidParam**

Test Purpose	To test BioSPI_Verify with valid parameters.
Test Scenario	Valid parameters are passed to BioSPI_Verify.
Expected Results	Return code BioAPI_OK. Result can be either BioAPI_TRUE or BioAPI_FALSE. FARAchieved must be valid.

19.2 Yes **BioSPI_Verify_FARUnspecified**

Test Purpose	To test failure on unspecified FAR.
Test Scenario	BioSPI_Verify is passed valid parameters except for MaxFARRequested.
Expected Results	Return code other than BioAPI_OK.

19.3 Yes **BioSPI_Verify_FRR**

Test Purpose	To test BioSPI_Verify using FRR as matching criteria. (The input MaxFRRRequested parameter is not NULL, and FARPrecedence set to BioAPI_FALSE).
Test Scenario	1) Load the BSP under test 2) Attach the BSP under test 3) Call the function BioSPI_Enroll to create a template. 4) Call the function BioSPI_Verify with FRR specified, check the output parameter FRRAchived. 5) Detach and unload BSP.
Expected Results	The returned FRRAchived is acceptable.

19a Set threshold using FRR criteria — *3.3.4.7*

No

Feature			*Reference*
Num	*Addressed*	*Assertion Name*	
19b		***Model/template adaptation***	*3.3.4.7*
19b.1	Yes	**BioSPI_Verify_TemplateAdaptation**	

Test Purpose To test Model/template adaptation if supported by the BSP.
Test Scenario If the BSP supports Model/template adaptation, call BioSPI_Verify with valid parameters.
Expected Results Return code BioAPI_OK and a valid AdaptedBIR.

19c Return of continuous scores — *1.7*

No

19d Return of achieved FRR score — *3.3.4.7*

No

19e Return of payload — *3.3.4.7*

19e.1 Yes **BioSPI_Verify_Payload**

Test Purpose To test Return of payload if supported by the BSP and verification succeeds.
Test Scenario If the BSP supports the return of payload, call BioSPI_Verify with valid parameters. ProcessedBIR and Stored template should be a "match."
Expected Results Return code BioAPI_OK, Result of BioAPI_TRUE and a valid Payload.

19f Return of raw/audit data — *3.3.4.7*

19f.1 Yes **BioSPI_Verify_AuditData**

Test Purpose To test the return of "raw" data if the BSP supports return of raw/audit data.
Test Scenario If the BSP supports the return of raw/audit data, call BioSPI_Verify with valid parameters, including AuditData.
Expected Results Return code BioAPI_OK and a valid AuditData.

19g Support of application control of the GUI — *1.10, 4.2.4.2*

No

Feature Num	Addressed	Assertion Name	Reference
19h		***BIR signing (by BSP)***	*1.5, 2.1.7*
19h.1	Yes	**BioSPI_Verify_BIRSignedTemplateAdaptation**	
		Test Purpose To test BIR signing if both BIR signing and Model/template adaptation are supported by the BSP.	
		Test Scenario If the BSP supports both BIR signing and Model/template adaptation, call BioSPI_Verify with valid parameters, including AdaptedBIR. The presented biometric should "match" the StoredTemplate.	
		Expected Results Return code BioAPI_OK, Result of BioAPI_TRUE and a valid, signed AdaptedBIR.	
19i		***BIR encryption (by BSP)***	*1.5, 2.1.7*
19i.1	Yes	**BioSPI_Verify_BIREncryptedTemplateAdaptation**	
		Test Purpose To test BIR encryption if both BIR encryption and Model/template adaptation are supported by the BSP.	
		Test Scenario If the BSP supports both BIR encryption and Model/template adaptation, call BioSPI_Verify with valid parameters, including AdaptedBIR. The presented biometric should "match" the StoredTemplate.	
		Expected Results Return code BioAPI_OK, Result of BioAPI_TRUE and a valid AdaptedBIR.	
19j		***BSP client/server comms***	*1.6, 4.2.4.2*
	No		
20		***BioSPI Identify***	*3.3.4.8*
	No		
20a		***Set threshold using FRR criteria***	*3.3.4.8*
	No		
20b		***Return of continuous scores***	*1.7*
	No		
20c		***Return of achieved FRR score***	*3.3.4.8*
	No		
20d		***Support of binning***	*3.3.4.8*
	No		

Feature Num	Addressed	Assertion Name	Reference
20e		***Return of raw/audit data***	*3.3.4.8*
	No		
20f		***Support of application control of the GUI***	*1.10, 4.2.4.2*
	No		
20g		***BIR signing (by BSP)***	*1.5, 2.1.7*
	No		
20h		***BIR encryption (by BSP)***	*1.5, 2.1.7*
	No		
20i		***BSP client/server comms***	*1.6, 4.2.4.2*
	No		
21		***BioSPI Import***	*3.3.4.9*
21.1	Yes	**BioSPI_Import_ValidParam**	
	Test Purpose	To test BioSPI_Import with valid parameters.	
	Test Scenario	Call BioSPI_Import with valid parameters.	
	Expected Results	Return code of BioAPI_OK and a valid ConstructedBIR.	
21.2	Yes	**BioSPI_Import_InvalidModuleHandle**	
	Test Purpose	To test BioSPI_Import with an invalid module handle.	
	Test Scenario	Call BioSPI_Import with an invalid module handle.	
	Expected Results	Return code other than BioAPI_OK and an invalid ConstructedBIR.	
21.3	Yes	**BioSPI_Import_InvalidInputData**	
	Test Purpose	To test BioSPI_Import with NULL input data.	
	Test Scenario	Call BioSPI_Import with NULL input data.	
	Expected Results	Return code other than BioAPI_OK.	
21.4	Yes	**BioSPI_Import_InvalidPurpose**	
	Test Purpose	To test BioSPI_Import with an invalid purpose.	
	Test Scenario	Call BioSPI_Import with an invalid purpose.	
	Expected Results	Return code other than BioAPI_OK.	

Feature Num	Addressed	Assertion Name	Reference
22		**BioSPI Set Power Mode**	*3.3.4.10*
	No		
23		**BioSPI Db Open**	*3.3.5.1*
23.1	Yes	**BioSPI_DbOpen_ValidParam**	
	Test Purpose	To test BioSPI_DbOpen with valid parameters if supported by the BSP.	
	Test Scenario	If supported by the BSP, call BioSPI_DbOpen with valid parameters.	
	Expected Results	Return code BioAPI_OK, a valid DbHandle and a valid Cursor.	
23.2	Yes	**BioSPI_DbOpen_InvalidModuleHandle**	
	Test Purpose	To test BioSPI_DbOpen with an invalid module handle if supported by the BSP.	
	Test Scenario	If supported by the BSP, call BioSPI_DbOpen with an invalid module handle.	
	Expected Results	Return code other than BioAPI_OK.	
23.3	Yes	**BioSPI_DbOpen_InvalidDBName**	
	Test Purpose	To test BioSPI_DbOpen with an invalid DB name if supported by the BSP.	
	Test Scenario	Call BioSPI_DbOpen with an invalid DB name.	
	Expected Results	Return code other than BioAPI_OK.	
23.4	Yes	**BioSPI_DbOpen_InvalidRequest**	
	Test Purpose	To test BioSPI_DbOpen with an invalid request if supported by the BSP.	
	Test Scenario	Call BioSPI_DbOpen with an invalid request.	
	Expected Results	Return code other than BioAPI_OK.	
24		**BioSPI Db Close**	*3.3.5.2*
24.1	Yes	**BioSPI_DbClose_ValidParam**	
	Test Purpose	To test BioSPI_DbClose with valid parameters if supported by the BSP.	
	Test Scenario	If supported by the BSP, call BioSPI_DbClose with valid parameters.	
	Expected Results	Return code BioAPI_OK.	
24.2	Yes	**BioSPI_DbClose_InvalidModuleHandle**	
	Test Purpose	To test BioSPI_DbClose with an invalid module handle if supported by the BSP.	
	Test Scenario	If supported by the BSP, call BioSPI_DbClose with an invalid module handle.	
	Expected Results	Return code other than BioAPI_OK.	
24.3	Yes	**BioSPI_DbClose_InvalidDBHandle**	
	Test Purpose	To test BioSPI_DbClose with an invalid DB handle if supported by the BSP.	
	Test Scenario	If supported by the BSP, call BioSPI_DbClose with an invalid DB handle.	
	Expected Results	Return code other than BioAPI_OK.	

Feature Num	Addressed	Assertion Name	Reference
25		**BioSPI Db Create**	*3.3.5.3*
25.1	Yes	**BioSPI_DbCreate_ValidParam**	
		Test Purpose To test BioSPI_DbCreate with valid parameters if suported by the BSP. ***Test Scenario*** If supported by the BSP, call BioSPI_DbCreate with valid parameters. ***Expected Results*** Return code BioAPI_OK and a valid DbHandle.	
25.2	Yes	**BioSPI_DbCreate_InvalidModuleHandle**	
		Test Purpose To test BioSPI_DbCreate with an invalid module handle if suported by the BSP. ***Test Scenario*** If supported by the BSP, call BioSPI_DbCreate with an invalid module handle. ***Expected Results*** Return code other than BioAPI_OK and a DbHandle of BioAPI_DB_INVALID_HANDLE.	
25.3	Yes	**BioSPI_DbCreate_DbProtected**	
		Test Purpose To test BioSPI_DbCreate existing database protection if suported by the BSP. ***Test Scenario*** If supported by the BSP, call BioSPI_DbCreate with valid parameters, with DbName the name of an existing database. ***Expected Results*** Return code of BioAPIERR_BSP_DATABASE_ALREADY_EXISTS and an invalid DbHandle.	
25.4	Yes	**BioSPI_DbCreate_InvalidDBName**	
		Test Purpose To test BioSPI_DbCreate with an invalid DB name if suported by the BSP. ***Test Scenario*** If supported by the BSP, call BioSPI_DbCreate with an invalid DB name. ***Expected Results*** Return code other than BioAPI_OK and a DbHandle of BioAPI_DB_INVALID_HANDLE.	
25.5	Yes	**BioSPI_DbCreate_InvalidRequest**	
		Test Purpose To test BioSPI_DbCreate with an invalid request if suported by the BSP. ***Test Scenario*** If supported by the BSP, call BioSPI_DbCreate with an invalid request. ***Expected Results*** Return code other than BioAPI_OK and a DbHandle of BioAPI_DB_INVALID_HANDLE.	
26		**BioSPI Db Delete**	*3.3.5.4*
26.1	Yes	**BioSPI_DbDelete_ValidParam**	
		Test Purpose To test BioSPI_DbDelete with valid parameters if supported by the BSP. ***Test Scenario*** If supported by the BSP, call BioSPI_DbDelete with valid parameters on an existing, closed database. ***Expected Results*** Return code of BioAPI_OK and the database deleted.	
26.2	Yes	**BioSPI_DbDelete_OpenDbProtected**	
		Test Purpose To test that BioSPI_DbDelete does not delete an open database if supported by the BSP. ***Test Scenario*** If supported by the BSP, open a database, then call BioSPI_DbDelete with a valid ModuleHandle and DbName the name of the open database. ***Expected Results*** Return code BioAPIERR_BSP_DATABASE_IS_OPEN and the database not deleted.	

Feature Num	Addressed	Assertion Name	Reference
26.3	Yes	**BioSPI_DbDelete_InvalidModuleHandle**	
	Test Purpose	To test BioSPI_DbDelete with invalid module handle if supported by the BSP.	
	Test Scenario	If supported by the BSP, call BioSPI_DbDelete with an invalid module handle.	
	Expected Results	Return code other than BioAPI_OK and the database not deleted if a valid database name was passed into BioSPi_DbDelete.	

27 BioSPI Db Set Cursor — 3.3.5.5

27.1	Yes	**BioSPI_DbSetCursor_ValidParam**	
	Test Purpose	To test BioSPI_DbSetCursor with valid parameters if supported by the BSP.	
	Test Scenario	If supported by the BSP, call BioSPI_DbSetCursor with valid parameters.	
	Expected Results	Return code of BioAPI_OK and a valid Cursor.	
27.2	Yes	**BioSPI_DbSetCursor_RecordNotFound**	
	Test Purpose	To test BioSPI_DbSetCursor "record not found" if supported by the BSP.	
	Test Scenario	If supported by the BSP, call BioSPI_DbSetCursor with valid parameters but KeyValue not in the database.	
	Expected Results	Return code of BioAPIERR_BSP_RECORD_NOT_FOUND.	
27.3	Yes	**BioSPI_DbSetCursor_InvalidModuleHandle**	
	Test Purpose	To test BioSPI_DbSetCursor with an invalid module handle if supported by the BSP.	
	Test Scenario	If supported by the BSP, call BioSPI_DbSetCursor with and invalid module handle.	
	Expected Results	Return code other than BioAPI_OK and an invalid Cursor.	
27.4	Yes	**BioSPI_DbSetCursor_InvalidDBHandle**	
	Test Purpose	To test BioSPI_DbSetCursor with an invalid DB handle if supported by the BSP.	
	Test Scenario	If supported by the BSP, call BioSPI_DbSetCursor with and invalid DB handle.	
	Expected Results	Return code other than BioAPI_OK and an invalid Cursor.	

28 BioSPI Db Free Cursor — 3.3.5.6

28.1	Yes	**BioSPI_DbFreeCursor_ValidParam**	
	Test Purpose	To test BioSPI_DbFreeCursor with valid parameters if supported by the BSP.	
	Test Scenario	If supported by the BSP, call BioSPI_DbFreeCursor with valid parameters.	
	Expected Results	Return code BioAPI_OK.	
28.2	Yes	**BioSPI_DbFreeCursor_InvalidModuleHandle**	
	Test Purpose	To test BioSPI_DbFreeCursor with an invalid module handle if supported by the BSP.	
	Test Scenario	If supported by the BSP, call BioSPI_DbFreeCursor with an invalid module handle.	
	Expected Results	Return code other than BioAPI_OK.	

Feature Num	Addressed	Assertion Name	Reference
28.3	Yes	**BioSPI_DbFreeCursor_InvalidCursor**	
	Test Purpose	To test BioSPI_DbFreeCursor "invalid cursor" if supported by the BSP.	
	Test Scenario	If supported by the BSP, call BioSPI_DbFreeCursor with a valid ModuleHandle and an invalid Cursor.	
	Expected Results	Return code BioAPIERR_BSP_CURSOR_IS_INVALID.	

29 BioSPI Db Store BIR — 3.3.5.7

29.1	Yes	**BioSPI_DbStoreBIR_ValidParam**	
	Test Purpose	To test BioSPI_DbStoreBIR with valid parameters if supported by the BSP.	
	Test Scenario	If supported by the BSP, call BioSPI_DbStoreBIR with valid parameters.	
	Expected Results	Return code BioAPI_OK.	
29.2	Yes	**BioSPI_DbStoreBIR_InvalidModuleHandle**	
	Test Purpose	To test BioSPI_DbStoreBIR with ian nvalid module handle if supported by the BSP.	
	Test Scenario	If supported by the BSP, call BioSPI_DbStoreBIR with an invalid module handle.	
	Expected Results	Return code other than BioAPI_OK.	
29.3	Yes	**BioSPI_DbStoreBIR_InvalidDBHandle**	
	Test Purpose	To test BioSPI_DbStoreBIR with ian nvalid DB handle if supported by the BSP.	
	Test Scenario	If supported by the BSP, call BioSPI_DbStoreBIR with an invalid DB handle.	
	Expected Results	Return code other than BioAPI_OK.	

30 BioSPI Db Get BIR — 3.3.5.8

30.1	Yes	**BioSPI_DbGetBIR_ValidParam**	
	Test Purpose	To test BioSPI_DbGetBIR with valid parameters if supported by the BSP.	
	Test Scenario	If supported by the BSP, call BioSPI_DbGetBIR with valid parameters.	
	Expected Results	Return code BioAPI_OK.	
30.2	Yes	**BioSPI_DbGetBIR_RecordNotFound**	
	Test Purpose	To test BioSPI_DbGetBIR "record not found" if supported by the BSP.	
	Test Scenario	If supported by the BSP, call BioSPI_DbGetBIR with valid parameters and KeyValue not in the DbHandle database.	
	Expected Results	Return code BioAPIERR_BSP_RECORD_NOT_FOUND.	
30.3	Yes	**BioSPI_DbGetBIR_InvalidModuleHandle**	
	Test Purpose	To test BioSPI_DbGetBIR with an invalid module handle if supported by the BSP.	
	Test Scenario	If supported by the BSP, call BioSPI_DbGetBIR with an invalid module handle.	
	Expected Results	Return code other than BioAPI_OK.	
30.4	Yes	**BioSPI_DbGetBIR_InvalidDBHandle**	
	Test Purpose	To test BioSPI_DbGetBIR with an invalid DB handle if supported by the BSP.	
	Test Scenario	If supported by the BSP, call BioSPI_DbGetBIR with an invalid DB handle.	
	Expected Results	Return code other than BioAPI_OK.	

Feature Num	Addressed	Assertion Name	Reference
30.5	Yes	**BioSPI_DbGetBIR_InvalidKeyValue**	

Test Purpose To test BioSPI_DbGetBIR with an invalid key value if supported by the BSP.
Test Scenario If supported by the BSP, call BioSPI_DbGetBIR with an invalid key value.
Expected Results Return code other than BioAPI_OK.

31 BioSPI Db Get Next BIR *3.3.5.9*

31.1 Yes **BioSPI_DbGetNextBIR_ValidParam**

Test Purpose To test BioSPI_DbGetNextBIR with valid parameters if supported by the BSP.
Test Scenario If supported by the BSP, call BioSPI_DbGetNextBIR with valid parameters.
Expected Results Return code BioAPI_OK.

31.2 Yes **BioSPI_DbGetNextBIR_InvalidModuleHandle**

Test Purpose To test BioSPI_DbGetNextBIR with an nvalid module handle if supported by the BSP.
Test Scenario If supported by the BSP, call BioSPI_DbGetNextBIR with an invalid module handle.
Expected Results Return code other than BioAPI_OK.

31.3 Yes **BioSPI_DbGetNextBIR_InvalidCursor**

Test Purpose To test BioSPI_DbGetNextBIR with an nvalid cursor if supported by the BSP.
Test Scenario If supported by the BSP, call BioSPI_DbGetNextBIR with an invalid cursor.
Expected Results Return code other than BioAPI_OK.

31.4 Yes **BioSPI_DbGetNextBIR_EndOfDatabase**

Test Purpose To test BioSPI_DbGetNextBIR returns the correct end of database indicator.
Test Scenario Repeatedly call BioSPI_DbGetNextBIR until there are no more records.
Expected Results Return code BioAPIERR_BSP_END_OF_DATABASE.

32 BioSPI Db Query BIR *3.3.5.10*

32.1 Yes **BioSPI_DbQueryBIR_ValidParam**

Test Purpose To test BioSPI_DbQueryBIR with valid parameters if supported by the BSP.
Test Scenario If supported by the BSP, call BioSPI_DbQueryBIR with valid parameters.
Expected Results Return code BioAPI_OK.

32.2 Yes **BioSPI_DbQueryBIR_InvalidModuleHandle.**

Test Purpose To test BioSPI_DbQueryBIR with ian nvalid module handle if supported by the BSP.
Test Scenario If supported by the BSP, call BioSPI_DbQueryBIR with an invalid module handle.
Expected Results Return code other than BioAPI_OK.

32.3 Yes **BioSPI_DbQueryBIR_InvalidDBHandle**

Test Purpose To test BioSPI_DbQueryBIR with ian nvalid DB handle if supported by the BSP.
Test Scenario If supported by the BSP, call BioSPI_DbQueryBIR with an invalid DB handle.
Expected Results Return code other than BioAPI_OK.

Feature Num	Addressed	Assertion Name	Reference
33		**BioSPI Db Delete BIR**	*3.3.5.11*
33.1	Yes	**BioSPI_DbDeleteBIR_ValidParam**	
		Test Purpose To test BioSPI_DbDeleteBIR with valid parameters if supported by the BSP. ***Test Scenario*** If supported by the BSP, call BioSPI_DbDeleteBIR with valid parameters. ***Expected Results*** Return code BioAPI_OK.	
33.2	Yes	**BioSPI_DbDeleteBIR_InvalidModuleHandle**	
		Test Purpose (?OMJ?) If supported by the BSP, call BioSPI_DbQueryBIR withan invalid module handle. ***Test Scenario*** If supported by the BSP, call BioSPI_DbDeleteBIR with an invalid module handle. ***Expected Results*** Return code other than BioAPI_OK.	
33.3	Yes	**BioSPI_DbDeleteBIR_InvalidDBHandle**	
		Test Purpose (?OMJ?) If supported by the BSP, call BioSPI_DbQueryBIR withan invalid DB handle. ***Test Scenario*** If supported by the BSP, call BioSPI_DbDeleteBIR with an invalid DB handle. ***Expected Results*** Return code other than BioAPI_OK.	
101		**BSP must implement all mandatory functions IAW SPI**	*A.2*
101.1	Yes	**BSP_MandatoryFunctionsImplemented**	
		Test Purpose To test that the BSP implements all mandatory functions: BioSPI_ModuleLoad, BioSPI_ModuleUnload, BioSPI_ModuleAttach, BioSPI_ModuleDetach, BioSPI_FreeBIRHandle, BioSPI_GetBIRFromHandle, BioSPI_GetHeaderFromHandle, BioSPI_Enroll and BioSPI_Verify. Identification BSPs must also implement BioSPI_Identify. If the BSP is a Client/Server BSP, these functions can be executed Locally or Remotely: BioSPI_Enroll, BioSPI_Verify, and (if an Identification BSP) BioSPI_Identify. ***Test Scenario*** N.A. ***Expected Results*** The BSP implements all mandatory functions. This assertion cannot be tested by calling BioSPI functions. The test lab will use some other means to determine if a BSP passes this assertion.	
102		**BSP accepts all valid input parameters and returns valid outputs**	*A.2*
102.1	Yes	**BSP_ValidInOut**	
		Test Purpose To test that the BSP accepts all valid inputs to all mandatory functions and implemented optional functions, and returns valid output from all mandatory functions and implemented optional functions. ***Test Scenario*** N.A. ***Expected Results*** The BSP accepts all valid inputs to all mandatory functions and implemented optional functions, and returns valid output from all mandatory functions and implemented optional functions. This assertion cannot be tested by calling BioSPI functions. The test lab will use some other means to determine if a BSP passes this assertion.	

Feature Num	Addressed	Assertion Name	Reference
103	Options implemented must be in accordance to spec (as shown in column 2 & 3 of Table)		A.2
103.1	Yes	**BSP_Options_InSpec**	

Test Purpose To test that all options implemented are correct.
Test Scenario N.A.
Expected Results Each implemented option should succeed according to its Test Cases.

This assertion cannot be tested by calling BioSPI functions. The test lab will use some other means to determine if a BSP passes this assertion.

104	BSP provides all registry entries		A.2
104.1	Yes	**BSP_Registry**	

Test Purpose To test that the BSP provides all registry entries.
Test Scenario N.A.
Expected Results The BSP provides all registry entries.

This assertion cannot be tested by calling BioSPI functions. The test lab will use some other means to determine if a BSP passes this assertion.

105	BSP has UUID according to data definition		A.2
105.1	Yes	**BSP_UUID**	

Test Purpose To test the BSP's UUID.
Test Scenario N.A.
Expected Results The UUID conforms to the specification.

This assertion cannot be tested by calling BioSPI functions. The test lab will use some other means to determine if a BSP passes this assertion.

106	Conformant data structures -- (Biometric data according to 2.1 & 3.2 including the BIR)		A.2
106.1	Yes	**DataConformant**	

Test Purpose To test conformance of data structures.
Test Scenario N.A.
Expected Results All data structures are conformant.

This assertion cannot be tested by calling BioSPI functions. The test lab will use some other means to determine if a BSP passes this assertion.

Feature			*Reference*
Num	*Addressed*	*Assertion Name*	
107	***Registered valid Format Owner and Format Type***		*A.2*
107.1	Yes	**RegisteredFormatOwnerType**	

Test Purpose To test that Format Owner and Format Type are registered and valid.
Test Scenario N.A.
Expected Results Format Owner and Format Type are registered and valid.

This assertion cannot be tested by calling BioSPI functions. The test lab will use some other means to determine if a BSP passes this assertion.

108	***Error handling according to 2.3***		*A.2*
108.1	Yes	**ErrorHandlingIAW_2.3**	

Test Purpose To test that the BSP implements error handling in accordance with 2.3.
Test Scenario N.A.
Expected Results The BSP implements error handling in accordance with 2.3.

This standard includes conformance requirements related to error handling to the extent that the errors are a) generated by the BSP (not the framework), b) explicitly listed as an error return in the function being tested, and c) are feasible to be induced by a test application. Testing of all possible error conditions is beyond the scope of this standard.

This assertion cannot be tested by calling BioSPI functions. The test lab will use some other means to determine if a BSP passes this assertion. If a BSP passes all assertions items 2 through 33 the BSP can be considered to have passed this assertion.

109	***If GUI BSP must provide it***		*A.2*
109.1	Yes	**BSP_GUI_Provided**	

Test Purpose To test that the BSP provides a GUI for the capture portion of the Verify operation.

Test Scenario N.A.
Expected Results The BSP provides a GUI for the capture portion of the Verify operation.

This assertion cannot be tested by calling BioSPI functions. The test lab will use some other means to determine if a BSP passes this assertion.

109.2	Yes	**BSP_GUI_Provided_Cancel**	

Test Purpose To test that if the BSP provides a GUI, an operator abort/cancel mechanism is also provided.
Test Scenario N.A.
Expected Results If the BSP provides a GUI, an operator abort/cancel mechanism is also provided.

This assertion cannot be tested by calling BioSPI functions. The test lab will use some other means to determine if a BSP passes this assertion.

15 Correspondence Between Implementation Options and Test Assertions to be Executed

15.1 This clause establishes which of the test assertions listed in Clause 15 are to be executed to test the BSP's conformance to the BioAPI specification, depending on the BSP type and BioAPI features implemented by the BSP (as listed in Clause 14).

15.2 For each BSP type, the set of assertions associated with this type shall be executed and then all applicable assertions that correspond to the BioAPI features implemented by the BSP under test shall be executed, preferably in the order they are listed in this clause.

15.3 Table 1 includes the assertions specified in this standard for "Verification BSPs". Table 13 includes the assertions for "Identification BSPs". Tables 2-12 and 14-26 include the assertions that correspond to BioAPI features.

15.4 A Verification BSP that implements BioSPI_EnableEvents and BioSPI_VerifyMatch functionality, for example shall execute at a minimum the assertions listed in tables 1 (assertions associated with the Verification type BSP), 2 (assertions for BioSPI_EnableEvents) and 6 (assertions for BioSPI_VerifyMatch).

15.5 If an implementation can read from the registry, it is permissible to execute all assertions. It is preferable the assertions be executed in the order listed.

15.6 Testing BSPs of Category "Verification BSP"

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

Assertion Name	**Package Name**
BioSPI_ModuleLoad_ValidParam	8ba15b00-d441-11d8-9669-0800200c9a66
BioSPI_ModuleLoad_InvalidUUID	41d96a80-d441-11d8-9669-0800200c9a66
BioSPI_ModuleUnload_ValidParam	b8efc140-d442-11d8-9669-0800200c9a66
BioSPI_ModuleUnload_Unmatch	6cc3c910-d442-11d8-9669-0800200c9a66
BioSPI_ModuleUnload_InvalidUUID	d3291f80-d441-11d8-9669-0800200c9a66
BioSPI_ModuleUnload_Confirm	ec706780-02a1-11d9-9669-0800200c9a66
BioSPI_ModuleAttach_ValidParam	379ab8d0-d4dc-11d8-9669-0800200c9a66
BioSPI_ModuleAttach_InvalidModuleHandle	6acd2480-d7f8-11d8-9669-0800200c9a66
BioSPI_ModuleAttach_InvalidUUID	d55a9350-d4e2-11d8-9669-0800200c9a66
BioSPI_ModuleAttach_InvalidVersion	96a0f5c0-d4e5-11d8-9669-0800200c9a66
BioSPI_ModuleDetach_ValidParam	120ec850-d4fe-11d8-9669-0800200c9a66
BioSPI_ModuleDetach_InvalidModuleHandle	efcbc370-d52e-11d8-9669-0800200c9a66
BioSPI_ModuleDetach_Confirm	2c8ce250-0732-11d9-9669-0800200c9a66
BioSPI_FreeBIRHandle_ValidParam	1c25d7d0-d5a2-11d8-9669-0800200c9a66
BioSPI_FreeBIRHandle_InvalidModuleHandle	7c822430-0595-11d9-9669-0800200c9a66
BioSPI_FreeBIRHandle_InvalidBIRHandle	67052160-d5c6-11d8-9669-0800200c9a66
BioSPI_GetBIRFromHandle_ValidParam	d4f3b820-d5c7-11d8-9669-0800200c9a66
BioSPI_GetBIRFromHandle_InvalidModuleHandle	6da9d8a0-d5c9-11d8-9669-0800200c9a66

BioSPI_GetBIRFromHandle_InvalidBIRHandle	24baf820-d5cb-11d8-9669-0800200c9a66
BioSPI_GetHeaderFromHandle_ValidParam	c591a6d0-d806-11d8-9669-0800200c9a66
BioSPI_GetHeaderFromHandle_InvalidModuleHandle	23e0bb00-d80b-11d8-9669-0800200c9a66
BioSPI_GetHeaderFromHandle_InvalidBIRHandle	5bdcfa20-d80e-11d8-9669-0800200c9a66
BioSPI_GetHeaderFromHandle_ BIRHandleNotFreed	0b43f8f0-08e4-11d9-9669-0800200c9a66
BioSPI_Enroll_ValidParam	f502b220-2f3f-11d9-9669-0800200c9a66
BioSPI_Verify_ValidParam	03c5ba70-30c9-11d9-9669-0800200c9a66
BioSPI_Verify_FARUnspecified	018ed978-07e1-1085-8934-0002a5d5fd2e

15.6.2 If an implementation claims to support BioSPI_EnableEvents (see [missing text]), the implementation shall be tested by executing all of the following assertions (preferably in order) to verify the claim:

Assertion Name	**Package Name**
BioSPI_EnableEvents_ValidParam	49a53f50-065a-11d9-9669-0800200c9a66
BioSPI_EnableEvents_InvalidModuleHandle	ccf8f930-0cab-11d9-9669-0800200c9a66

15.6.3 If an implementation claims to support BioSPI_Capture (see [missing text]), the implementation shall be tested by executing all of the following assertions (preferably in order) to verify the claim:

Assertion Name	**Package Name**
BioSPI_Capture_IntermediateProcessedBIR	62dfd2f0-d5cc-11d8-9669-0800200c9a66
BioSPI_Capture_InvalidModuleHandle	5cdc27d0-2835-11d9-9669-0800200c9a66

15.6.4 If an implementation claims to support BioSPI_CreateTemplate (see [missing text]), the implementation shall be tested by executing all of the following assertions (preferably in order) to verify the claim:

Assertion Name	**Package Name**
BioSPI_CreateTemplate_OutputBIRPurpose	01094930-29dc-11d9-9669-0800200c9a66
BioSPI_CreateTemplate_OutputBIRDataType	39ec8660-2c23-11d9-9669-0800200c9a66
BioSPI_CreateTemplate_InputBIRDataType	6d543ea0-2ce9-11d9-9669-0800200c9a66
BioSPI_CreateTemplate_Purpose	25e3d1b0-2cf4-11d9-9669-0800200c9a66

15.6.5 If an implementation claims to support BioSPI_Process (see [missing text]), the implementation shall be tested by executing all of the following assertions (preferably in order) to verify the claim:

Assertion Name	**Package Name**
BioSPI_Process_ValidParam	f3604490-2d01-11d9-9669-0800200c9a66
BioSPI_Process_OutputBIRPurpose	015b1f48-07e1-1085-8918-0002a5d5fd2e
BioSPI_Process_BuildsProcessedBIR	d555ce30-2d0f-11d9-9669-0800200c9a66
BioSPI_ Process_InputBIRDataType	dec4ca20-3a25-11d9-9669-0800200c9a66

15.6.6 If an implementation claims to support BioSPI_VerifyMatch (see [missing text]), the implementation shall be tested by executing all of the following assertions (preferably in order) to verify the claim:

Assertion Name	**Package Name**
BioSPI_VerifyMatch_ValidParam	60f19fb0-dcec-11d8-9669-0800200c9a66
BioSPI_VerifyMatch_UnspecifiedFAR	229f7f90-de39-11d8-9669-0800200c9a66
BioSPI_VerifyMatch_Inconsistent_Purpose	9108ec70-2e9b-11d9-9669-0800200c9a66

15.6.7 If an implementation claims to support return of raw/audit data (see [missing text]), the implementation shall be tested by executing all of the following assertions (preferably in order) to verify the claim:

Assertion Name	**Package Name**
BioSPI_Enroll_AuditData	d4ab3b80-2f6e-11d9-9669-0800200c9a66
BioSPI_Verify_AuditData	51b2a530-31c9-11d9-9669-0800200c9a66

15.6.8 If an implementation claims to support BioSPI_Capture, the implementation shall be tested by executing the following assertion to verify the claim:

Assertion Name	**Package Name**
BioSPI_Capture_AuditData	44450e60-2445-11d9-9669-0800200c9a66

15.6.9 If an implementation claims to support BioSPI_Capture and the return of a quality value (in the BIR header) for intermediate biometric data, the implementation shall be tested by executing the following assertion to verify the claim:

Assertion Name	**Package Name**
BioSPI_Capture_ReturnQuality	ace3ca10-2765-11d9-9669-0800200c9a66

15.6.10 If an implementation claims to support return of a quality value (in the BIR header) for processed biometric data, the implementation shall be tested by executing the following assertion to verify the claim:

Assertion Name	**Package Name**
BioSPI_Enroll_BIRHeaderQuality	0e8932f0-2f72-11d9-9669-0800200c9a66

15.6.11 If an implementation claims to support BioSPI_Capture, the implementation shall be tested by executing the following assertion to verify the claim:

Assertion Name	**Package Name**
BioSPI_Capture_ReturnQuality	ace3ca10-2765-11d9-9669-0800200c9a66

15.6.12 If an implementation claims to support BioSPI_CreateTemplate, the implementation shall be tested by executing the following assertion to verify the claim:

Assertion Name	**Package Name**
BioSPI_CreateTemplate_BIRHeaderQuality	3712a3e0-2c4e-11d9-9669-0800200c9a66

15.6.13 If an implementation claims to support BioSPI_Process, the implementation shall be tested by executing the following assertion to verify the claim:

Assertion Name	**Package Name**
BioSPI_Process_BIRHeaderQuality	97356030-2d0e-11d9-9669-0800200c9a66

15.6.14 If an implementation claims to support payload carry (accepts payload during enroll/process and returns payload upon successful verify), the implementation shall be tested by executing the following assertions (preferably in order) to verify the claim:

Assertion Name	**Package Name**
BioSPI_Enroll_Payload	ebe2c3e0-2f4b-11d9-9669-0800200c9a66
BioSPI_Verify_Payload	12b4a540-31c4-11d9-9669-0800200c9a66

15.6.15 If an implementation claims to support BioSPI_CreateTemplate, the implementation shall be tested by executing the following assertion to verify the claim:

Assertion Name	**Package Name**
BioSPI_CreateTemplate_PayloadSupported	d31ceee0-2c43-11d9-9669-0800200c9a66

15.6.16 If an implementation claims to support BioSPI_VerifyMatch, the implementation shall be tested by executing the following assertion to verify the claim:

Assertion Name	**Package Name**
BioSPI_VerifyMatch_Payload	48d18e60-2e98-11d9-9669-0800200c9a66

15.6.17 If an implementation claims to support the return of actual FRR during matching operations (Verify, VerifyMatch), the implementation shall be tested by executing the following assertion to verify the claim:

Assertion Name	**Package Name**
BioSPI_Verify_FRR	302849f0-31a2-11d9-9669-0800200c9a66

15.6.18 If an implementation claims to support BioSPI_VerifyMatch, the implementation shall be tested by executing the following assertion to verify the claim:

Assertion Name	**Package Name**

BioSPI_VerifyMatch_AchievedFRR	6afc90e0-2e87-11d9-9669-0800200c9a66

15.6.19 If an implementation claims to support BIR Adaptation, the implementation shall be tested by executing the following assertion to verify the claim:

Assertion Name	**Package Name**
BioSPI_Verify_TemplateAdaptation	0de718a0-31b9-11d9-9669-0800200c9a66

15.6.20 If an implementation claims to support BioSPI_CreateTemplate, the implementation shall be tested by executing the following assertion to verify the claim:

Assertion Name	**Package Name**
BioSPI_CreateTemplate_StoredTemplateUnchanged	ac81ba70-2c27-11d9-9669-0800200c9a66

15.6.21 If an implementation claims to support BioSPI_VerifyMatch, the implementation shall be tested by executing the following assertion to verify the claim:

Assertion Name	**Package Name**
BioSPI_VerifyMatch_Adaptation	91bee710-2e7c-11d9-9669-0800200c9a66

15.7 Testing BSPs of Category "Identification BSP"

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

Assertion Name	**Package Name**
BioSPI_ModuleLoad_ValidParam	8ba15b00-d441-11d8-9669-0800200c9a66
BioSPI_ModuleLoad_InvalidUUID	41d96a80-d441-11d8-9669-0800200c9a66
BioSPI_ModuleUnload_ValidParam	b8efc140-d442-11d8-9669-0800200c9a66
BioSPI_ModuleUnload_Unmatch	6cc3c910-d442-11d8-9669-0800200c9a66
BioSPI_ModuleUnload_InvalidUUID	d3291f80-d441-11d8-9669-0800200c9a66
BioSPI_ModuleUnload_Confirm	ec706780-02a1-11d9-9669-0800200c9a66
BioSPI_ModuleAttach_ValidParam	379ab8d0-d4dc-11d8-9669-0800200c9a66
BioSPI_ModuleAttach_InvalidModuleHandle	6acd2480-d7f8-11d8-9669-0800200c9a66
BioSPI_ModuleAttach_InvalidUUID	d55a9350-d4e2-11d8-9669-0800200c9a66
BioSPI_ModuleAttach_InvalidVersion	96a0f5c0-d4e5-11d8-9669-0800200c9a66
BioSPI_ModuleDetach_ValidParam	120ec850-d4fe-11d8-9669-0800200c9a66
BioSPI_ModuleDetach_InvalidModuleHandle	efcbc370-d52e-11d8-9669-0800200c9a66
BioSPI_ModuleDetach_Confirm	2c8ce250-0732-11d9-9669-0800200c9a66
BioSPI_FreeBIRHandle_ValidParam	1c25d7d0-d5a2-11d8-9669-0800200c9a66
BioSPI_FreeBIRHandle_InvalidModuleHandle	7c822430-0595-11d9-9669-0800200c9a66
BioSPI_FreeBIRHandle_InvalidBIRHandle	67052160-d5c6-11d8-9669-0800200c9a66
BioSPI_GetBIRFromHandle_ValidParam	d4f3b820-d5c7-11d8-9669-0800200c9a66
BioSPI_GetBIRFromHandle_InvalidModuleHandle	6da9d8a0-d5c9-11d8-9669-0800200c9a66
BioSPI_GetBIRFromHandle_InvalidBIRHandle	24baf820-d5cb-11d8-9669-0800200c9a66

BioSPI_GetHeaderFromHandle_ValidParam	c591a6d0-d806-11d8-9669-0800200c9a66
BioSPI_GetHeaderFromHandle_InvalidModuleHandle	23e0bb00-d80b-11d8-9669-0800200c9a66
BioSPI_GetHeaderFromHandle_InvalidBIRHandle	5bdcfa20-d80e-11d8-9669-0800200c9a66
BioSPI_GetHeaderFromHandle_ BIRHandleNotFreed	0b43f8f0-08e4-11d9-9669-0800200c9a66
BioSPI_Enroll_ValidParam	f502b220-2f3f-11d9-9669-0800200c9a66
BioSPI_Verify_ValidParam	03c5ba70-30c9-11d9-9669-0800200c9a66
BioSPI_Verify_FARUnspecified	018ed978-07e1-1085-8934-0002a5d5fd2e

15.7.2 If an implementation claims to support BioSPI_EnableEvents (see [missing text]), the implementation shall be tested by executing all of the following assertions (preferably in order) to verify the claim:

Assertion Name	**Package Name**
BioSPI_EnableEvents_ValidParam	49a53f50-065a-11d9-9669-0800200c9a66
BioSPI_EnableEvents_InvalidModuleHandle	ccf8f930-0cab-11d9-9669-0800200c9a66

15.7.3 If an implementation claims to support BioSPI_Capture (see [missing text]), the implementation shall be tested by executing all of the following assertions (preferably in order) to verify the claim:

Assertion Name	**Package Name**
BioSPI_Capture_IntermediateProcessedBIR	62dfd2f0-d5cc-11d8-9669-0800200c9a66
BioSPI_Capture_InvalidModuleHandle	5cdc27d0-2835-11d9-9669-0800200c9a66

15.7.4 If an implementation claims to support BioSPI_CreateTemplate (see [missing text]), the implementation shall be tested by executing all of the following assertions (preferably in order) to verify the claim:

Assertion Name	**Package Name**
BioSPI_CreateTemplate_OutputBIRPurpose	01094930-29dc-11d9-9669-0800200c9a66
BioSPI_CreateTemplate_OutputBIRDataType	39ec8660-2c23-11d9-9669-0800200c9a66
BioSPI_CreateTemplate_InputBIRDataType	6d543ea0-2ce9-11d9-9669-0800200c9a66
BioSPI_CreateTemplate_Purpose	25e3d1b0-2cf4-11d9-9669-0800200c9a66

15.7.5 If an implementation claims to support BioSPI_Process (see [missing text]), the implementation shall be tested by executing all of the following assertions (preferably in order) to verify the claim:

Assertion Name	**Package Name**
BioSPI_Process_ValidParam	f3604490-2d01-11d9-9669-0800200c9a66
BioSPI_Process_OutputBIRPurpose	015b1f48-07e1-1085-8918-0002a5d5fd2e
BioSPI_Process_BuildsProcessedBIR	d555ce30-2d0f-11d9-9669-0800200c9a66
BioSPI_ Process_InputBIRDataType	dec4ca20-3a25-11d9-9669-0800200c9a66

15.7.6 If an implementation claims to support BioSPI_VerifyMatch (see [missing text]), the implementation shall be tested by executing all of the following assertions (preferably in order) to verify the claim:

Assertion Name	**Package Name**
BioSPI_VerifyMatch_ValidParam	60f19fb0-dcec-11d8-9669-0800200c9a66
BioSPI_VerifyMatch_UnspecifiedFAR	229f7f90-de39-11d8-9669-0800200c9a66
BioSPI_VerifyMatch_Inconsistent_Purpose	9108ec70-2e9b-11d9-9669-0800200c9a66

15.7.7 If an implementation claims to support return of raw/audit data (see [missing text]), the implementation shall be tested by executing all of the following assertions (preferably in order) to verify the claim:

Assertion Name	**Package Name**
BioSPI_Enroll_AuditData	d4ab3b80-2f6e-11d9-9669-0800200c9a66
BioSPI_Verify_AuditData	51b2a530-31c9-11d9-9669-0800200c9a66

15.7.8 If an implementation claims to support BioSPI_Capture, the implementation shall be tested by executing the following assertion to verify the claim:

Assertion Name	**Package Name**
BioSPI_Capture_AuditData	44450e60-2445-11d9-9669-0800200c9a66

15.7.9 If an implementation claims to support BioSPI_Capture and the return of a quality value (in the BIR header) for intermediate biometric data, the implementation shall be tested by executing the following assertion to verify the claim:

Assertion Name	**Package Name**
BioSPI_Capture_ReturnQuality	ace3ca10-2765-11d9-9669-0800200c9a66

15.7.10 If an implementation claims to support return of a quality value (in the BIR header) for processed biometric data, the implementation shall be tested by executing the following assertion to verify the claim:

Assertion Name	**Package Name**
BioSPI_Enroll_BIRHeaderQuality	0e8932f0-2f72-11d9-9669-0800200c9a66

15.7.11 If an implementation claims to support BioSPI_Capture, the implementation shall be tested by executing the following assertion to verify the claim:

Assertion Name	**Package Name**
BioSPI_Capture_ReturnQuality	ace3ca10-2765-11d9-9669-0800200c9a66

15.7.12 If an implementation claims to support BioSPI_CreateTemplate, the implementation shall be tested by executing the following assertion to verify the claim:

Assertion Name	**Package Name**
BioSPI_CreateTemplate_BIRHeaderQuality	3712a3e0-2c4e-11d9-9669-0800200c9a66

15.7.13 If an implementation claims to support BioSPI_Process, the implementation shall be tested by executing the following assertion to verify the claim:

Assertion Name	**Package Name**
BioSPI_Process_BIRHeaderQuality	97356030-2d0e-11d9-9669-0800200c9a66

15.7.14 If an implementation claims to support payload carry (accepts payload during enroll/process and returns payload upon successful verify), the implementation shall be tested by executing the following assertions (preferably in order) to verify the claim:

Assertion Name	**Package Name**
BioSPI_Enroll_Payload	ebe2c3e0-2f4b-11d9-9669-0800200c9a66
BioSPI_Verify_Payload	12b4a540-31c4-11d9-9669-0800200c9a66

15.7.15 If an implementation claims to support BioSPI_CreateTemplate, the implementation shall be tested by executing the following assertion to verify the claim:

Assertion Name	**Package Name**
BioSPI_CreateTemplate_PayloadSupported	d31ceee0-2c43-11d9-9669-0800200c9a66

15.7.16 If an implementation claims to support BioSPI_VerifyMatch, the implementation shall be tested by executing the following assertion to verify the claim:

Assertion Name	**Package Name**
BioSPI_VerifyMatch_Payload	48d18e60-2e98-11d9-9669-0800200c9a66

15.7.17 If an implementation claims to support the return of actual FRR during matching operations (Verify, VerifyMatch), the implementation shall be tested by executing the following assertion to verify the claim:

Assertion Name	**Package Name**
BioSPI_Verify_FRR	302849f0-31a2-11d9-9669-0800200c9a66

15.7.18 If an implementation claims to support BioSPI_VerifyMatch, the implementation shall be tested by executing the following assertion to verify the claim:

Assertion Name	**Package Name**

BioSPI_VerifyMatch_AchievedFRR	6afc90e0-2e87-11d9-9669-0800200c9a66

15.7.19 If an implementation claims to support BIR Adaptation, the implementation shall be tested by executing the following assertion to verify the claim:

Assertion Name	**Package Name**
BioSPI_Verify_TemplateAdaptation	0de718a0-31b9-11d9-9669-0800200c9a66

15.7.20 If an implementation claims to support BioSPI_CreateTemplate, the implementation shall be tested by executing the following assertion to verify the claim:

Assertion Name	**Package Name**
BioSPI_CreateTemplate_StoredTemplateUnchanged	ac81ba70-2c27-11d9-9669-0800200c9a66

15.7.21 If an implementation claims to support BioSPI_VerifyMatch, the implementation shall be tested by executing the following assertion to verify the claim:

Assertion Name	**Package Name**
BioSPI_VerifyMatch_Adaptation	91bee710-2e7c-11d9-9669-0800200c9a66

15.7.22 If an implementation claims to support a BSP-controlled database, the implementation shall be tested by executing all of the following assertions (preferably in order) to verify the claim:

Assertion Name	**Package Name**
BioSPI_DbClose_InvalidModuleHandle	0033e140-0907-1085-adb5-0002a5d5fd2e
BioSPI_DbClose_ValidParam	02a56228-0900-1085-80f7-0002a5d5fd2e
BioSPI_DbCreate_DbProtected	00b93610-093b-1085-b648-0002a5d5fd2e
BioSPI_DbCreate_InvalidModuleHandle	019ecb80-093c-1085-9f29-0002a5d5fd2e
BioSPI_DbCreate_ValidParam	055b3358-0908-1085-9db7-0002a5d5fd2e
BioSPI_DbDelete_InvalidModuleHandle	035a1fb0-096e-1085-ba36-0002a5d5fd2e
BioSPI_DbDelete_OpenDbProtected	00581ab0-0970-1085-9505-0002a5d5fd2e
BioSPI_DbDelete_ValidParam	054198a8-093c-1085-8078-0002a5d5fd2e
BioSPI_DbDeleteBIR_InvalidModuleHandle	03aab498-09c6-1085-b751-0002a5d5fd2e
BioSPI_DbDeleteBIR_ValidParam	00890df0-09c5-1085-9d39-0002a5d5fd2e
BioSPI_DbFreeCursor_InvalidCursor	028a3cf0-0980-1085-88df-0002a5d5fd2e
BioSPI_DbFreeCursor_InvalidModuleHandle	0097bfa8-0980-1085-8279-0002a5d5fd2e
BioSPI_DbFreeCursor_ValidParam	05109d98-097f-1085-a9f1-0002a5d5fd2e
BioSPI_DbGetBIR_InvalidModuleHandle	01067768-09a4-1085-8ff8-0002a5d5fd2e
BioSPI_DbGetBIR_RecordNotFound	0370eba0-09ab-1085-8bb8-0002a5d5fd2e
BioSPI_DbGetBIR_ValidParam	04ac0798-09a2-1085-87aa-0002a5d5fd2e
BioSPI_DbGetNextBIR_InvalidModuleHandle	05c91440-09ac-1085-97e9-0002a5d5fd2e
BioSPI_DbGetNextBIR_ValidParam	038f3d58-09ac-1085-a5f6-0002a5d5fd2e
BioSPI_DbOpen_InvalidModuleHandle	0373dd88-08f7-1085-ab39-0002a5d5fd2e
BioSPI_DbOpen_ValidParam	03caee98-08ef-1085-80f2-0002a5d5fd2e

BioSPI_DbQueryBIR_InvalidModuleHandle	014bdd08-09bc-1085-91a2-0002a5d5fd2e
BioSPI_DbQueryBIR_ValidParam	03a18120-09b5-1085-9cbf-0002a5d5fd2e
BioSPI_DbSetCursor_InvalidModuleHandle	058dd448-0978-1085-a946-0002a5d5fd2e
BioSPI_DbSetCursor_RecordNotFound	00ff0cf8-0979-1085-8859-0002a5d5fd2e
BioSPI_DbSetCursor_ValidParam	0362e5c8-0978-1085-899a-0002a5d5fd2e
BioSPI_DbStoreBIR_InvalidModuleHandle	0352a5a0-098a-1085-be57-0002a5d5fd2e
BioSPI_DbStoreBIR_ValidParam	0509bfc8-0988-1085-82a3-0002a5d5fd2e

15.7.23 If an implementation claims to support BioSPI_Import, the implementation shall be tested by executing the following assertions (preferably in order) to verify the claim:

Assertion Name	**Package Name**
BioSPI_Import_InvalidModuleHandle	04c0a4f0-0a35-1085-bf8a-0002a5d5fd2e
BioSPI_Import_ValidParam	00769b48-09e9-1085-8059-0002a5d5fd2e

16 Test Assertions and Test Cases

16.1 Common activities

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

```
<package name="be0a1330-d59e-11d8-9669-0800200c9a66">
  <author>

  </author>
  <description>
  package contains several useful activities that are invoked by
activities in other packages.
  </description>
  <activity name="LoadAndAttach">
    <input name="moduleUuid"/>
    <input name="moduleVersionMajor"/>
    <input name="moduleVersionMinor"/>
    <input name="deviceIDOrNull"/>
    <input name="module"/>
    <input name="eventtimeouttime"/>
    <!-- Initialize the global variable "_deviceIDOrNull" to make it
available to the activity "EventHandler" -->
    <set name="_deviceIDOrNull" var="deviceIDOrNull"/>
    <!-- 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_ModuleLoad. -->
    <invoke function="BioSPI_ModuleLoad">
      <input name="Reserved" value="0"/>
      <input name="BSPUuid" var="moduleUuid"/>
      <input name="BioAPINotifyCallback" value="*"/>
      <input name="BioAPINotifyCallbackCtx" value="*"/>
      <return setvar="return"/>
    </invoke>
    <!-- Issue a conformity response.
      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>
      function BioSPI_ModuleLoad 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.
        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>
        BioAPI_NOTIFY_INSERT event notification has been received within
the specified maximum duration
        </description>
        <not var="eventtimeoutflag"/>
      </assert_condition>
      <!-- Invoke the function BioSPI_ModuleAttach. -->
      <invoke function="BioSPI_ModuleAttach">
        <input name="BSPUuid" var="moduleUuid"/>
        <input name="VersionMajor" var="moduleVersionMajor"/>
        <input name="VersionMinor" var="moduleVersionMinor"/>
        <input name="DeviceID" var="_deviceID"/>
        <input name="Reserved1" value="0"/>
        <input name="Reserved2" value="0"/>
        <input name="ModuleHandle" var="module"/>
        <input name="Reserved3" value="0"/>
        <input name="Reserved4" value="0"/>
        <input name="Reserved5" value="0"/>
        <input name="Reserved6" value="0"/>
        <return setvar="return"/>
      </invoke>
      <!-- Issue a conformity response.
        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>
        function BioSPI_ModuleAttach 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 "_deviceID", "_insert", "_sourcePresent"
and "_eventtype" are set depending on the input parameter values. -->
    <activity name="EventHandler" atomic="true">
      <input name="BSPUuid"/>
      <input name="BioAPINotifyCallbackCtx"/>
      <input name="DeviceID"/>
      <input name="Reserved"/>
      <input name="EventType"/>
```

```
     <output name="return"/>
     <!-- Set the global variable "_deviceID" 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 device, as specified by
the parameter "deviceIDOrNULL" of the "LoadAndAttach" activity
invocation. -->
     <set name="_deviceID" var="DeviceID">
       <only_if>
         <not>
           <existing var="_deviceID"/>
         </not>
         <or>
           <equal_to var1="EventType" var2="__BioAPI_NOTIFY_INSERT"/>
           <equal_to var1="EventType"
var2="__BioAPI_NOTIFY_SOURCE_PRESENT"/>
         </or>
         <or>
           <equal_to var1="_deviceIDOrNull" value2="0"/>
           <equal_to var1="_deviceIDOrNull" var2="DeviceID"/>
         </or>
       </only_if>
     </set>
     <invoke activity="EventHandlerSetGlobalData">
       <only_if>
         <existing var="_deviceID"/>
       </only_if>
       <input name="DeviceID" var="DeviceID"/>
       <input name="EventType" var="EventType"/>
     </invoke>
     <!-- Issue a conformity response.
       the condition specified in the <description> below is false, a
FAIL conformity response is issued, otherwise a PASS conformity
response is issued  (see BioAPI 1.1, 3.2.1). -->
     <assert_condition>
       <description>
       input value of the parameter "Reserved" of the event handler is
0.
       </description>
       <equal_to var1="Reserved" value2="0"/>
     </assert_condition>
     <set name="return" var="__BioAPI_OK"/>
   </activity>
   <activity name="EventHandlerSetGlobalData" atomic="true">
     <input name="DeviceID"/>
     <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 device, as specified by
the parameter "deviceIDOrNULL" of the "LoadAndAttach" activity
invocation. -->
     <set name="_insert" value="true">
       <only_if>
         <equal_to var1="EventType" var2="__BioAPI_NOTIFY_INSERT"/>
         <equal_to var1="_deviceID" var2="DeviceID"/>
       </only_if>
```

```
     </set>
     <!-- Set the global variable "_insert" to "false" if:
          the event notification is BioAPI_NOTIFY_REMOVE; and
          the event is related to the expected device, as specified by
the parameter "deviceIDOrNULL" of the "LoadAndAttach" activity
invocation. -->
     <set name="_insert" value="false">
       <only_if>
         <equal_to var1="EventType" var2="__BioAPI_NOTIFY_REMOVE"/>
         <equal_to var1="_deviceID" var2="DeviceID"/>
       </only_if>
     </set>
     <!-- Set the global variable "_sourcePresent" to "true" if:
          the event notification is BioAPI_NOTIFY_SOURCE_PRESENT; and
          the event is related to the expected device, as specified by
the parameter "deviceIDOrNULL" of the "LoadAndAttach" activity
invocation. -->
     <set name="_sourcePresent" value="true">
       <only_if>
         <equal_to var1="EventType"
var2="__BioAPI_NOTIFY_SOURCE_PRESENT"/>
         <equal_to var1="_deviceID" var2="DeviceID"/>
       </only_if>
     </set>
     <!-- Set the global variable "_sourcePresent" to "false" if:
          the event notification is BioAPI_NOTIFY_SOURCE_REMOVED; and
          the event is related to the expected device, as specified by
the parameter "deviceIDOrNULL" of the "LoadAndAttach" activity
invocation. -->
     <set name="_sourcePresent" value="false">
       <only_if>
         <equal_to var1="EventType"
var2="__BioAPI_NOTIFY_SOURCE_REMOVED"/>
         <equal_to var1="_deviceID" var2="DeviceID"/>
       </only_if>
     </set>
     <!-- Set the global variable "_eventType" -->
     <set name="_eventtype" var="EventType">
       <only_if>
         <equal_to var1="_deviceID" var2="DeviceID"/>
       </only_if>
     </set>
   </activity>
   <!-- This activity invokes the functions BioSPI_ModuleDetach and
BioSPI_ModuleUnload. -->
   <activity name="DetachAndUnload">
     <input name="moduleUuid"/>
     <input name="module"/>
     <!-- Invoke the function BioSPI_ModuleDetach.-->
     <invoke function="BioSPI_ModuleDetach">
       <input name="ModuleHandle" var="module"/>
       <return setvar="return"/>
     </invoke>
     <!-- Issue a conformity response.
       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>
       function BioSPI_ModuleDetach has returned BioAPI_OK.
       </description>
       <equal_to var1="return" var2="__BioAPI_OK"/>
     </assert_condition>
     <!-- Invoke the function BioSPI_ModuleUnload.-->
     <invoke function="BioSPI_ModuleUnload">
       <input name="Reserved" value="0"/>
       <input name="BSPUuid" var="moduleUuid"/>
       <input name="BioAPINotifyCallback" value="*"/>
       <input name="BioAPINotifyCallbackCtx" value="*"/>
       <return setvar="return"/>
     </invoke>
     <!-- Issue a conformity response.
       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>
       function BioSPI_ModuleUnload has returned BioAPI_OK.
       </description>
       <equal_to var1="return" var2="__BioAPI_OK"/>
     </assert_condition>
   </activity>
   <activity name="check_capturedBIR_datatype">
     <input name="ModuleHandle"/>
     <input name="BIRHandle"/>
     <!-- Invoke the function BioSPI_GetHeaderFromHandle on the captured
BIR. -->
     <invoke function="BioSPI_GetHeaderFromHandle">
       <input name="ModuleHandle" var="ModuleHandle"/>
       <input name="Handle" var="BIRHandle"/>
       <output name="ProcessedLevel" setvar="processedLevel"/>
       <return setvar="return"/>
     </invoke>
     <!-- Issue a conformity response.
       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>
         function BioSPI_GetHeaderFromHandle has returned BioAPI_OK
       </description>
       <equal_to var1="return" var2="__BioAPI_OK"/>
     </assert_condition>
     <!-- Issue a conformity response.
       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>
     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="ModuleHandle"/>
     <input name="CapturedBIR_BIRHandle"/>
     <!-- Invoke the function BioSPI_Process. -->
     <invoke function="BioSPI_Process">
       <input name="ModuleHandle" var="ModuleHandle"/>
       <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.
       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>
       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="farAchieved"/>
     <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="farAchieved" var="farAchieved"/>
     </invoke>
   </activity>
   <activity name="payloadNotSupport_checkPayload">
     <input name="outputPayload"/>
     <!-- Issue a conformity response.
```

```
        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>
        payload has been returned.
        </description>
        <same_as var1="outputPayload" value2=""/>
     </assert_condition>
   </activity>
   <activity name="resultFalse_checkPayload">
     <input name="outputPayload"/>
     <!-- Issue a conformity response.
        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>
        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="farAchieved"/>
     <!-- Issue a conformity response.
        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, if the result is true and the achieved
FAR is less than the payload policy value, then the returned payload is
the same as the provided payload, otherwise the output payload is NULL.
        </description>
        <or>
          <and>
            <less_than var1="farAchieved" var2="payloadPolicy"/>
            <same_as var1="outputPayload" var2="inputPayload"/>
          </and>
          <and>
            <greater_than_or_equal_to var1="farAchieved"
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 does not claim to support template adaptation. -->
   <activity name="check_adaptation_supported">
     <input name="adaptedbir_handle"/>
     <assert_condition response_if_false="undecided"
```

```
break_if_false="true">
       <description>
       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>
       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>
       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 FRRAchieved. It
is used if the BSP claims to support FRR. -->
   <activity name="check_FRR_supported">
     <input name="frrAchieved"/>
     <assert_condition response_if_false="undecided"
break_if_false="true">
       <description>
       has a valid value.
       </description>
       <not_equal_to var1="frrAchieved" value2="-1"/>
     </assert_condition>
     <assert_condition break_if_false="true">
       <description>
       is non-negative.
       </description>
       <greater_than_or_equal_to var1="frrAchieved" value2="0"/>
     </assert_condition>
   </activity>
   <!-- This activity checks the value of the returned FRRAchieved. It
is used if the BSP does not claim to support FRR. -->
   <activity name="check_FRR_not_supported">
     <input name="frrAchieved"/>
     <assert_condition break_if_false="true">
       <description>
       equals -2, indicating that the BSP does not support FRR.
       </description>
       <equal_to var1="frrAchieved" 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.
       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>
       has a valid value.
       </description>
       <not_equal_to var1="quality" value2="-1"/>
     </assert_condition>
     <!-- Issue a conformity response.
       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>
       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>
       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="ModuleHandle"/>
     <input name="dbName"/>
     <input name="nosourcepresentsupported"/>
     <output name="biruuid"/>
     <invoke function="BioSPI_DbDelete">
       <input name="ModuleHandle" var="ModuleHandle"/>
       <input name="DbName" var="dbName"/>
       <return setvar="return"/>
     </invoke>
     <invoke function="BioSPI_DbCreate">
       <input name="ModuleHandle" var="ModuleHandle"/>
       <input name="DbName" var="dbName"/>
       <input name="ReadAccessRequest" value="true"/>
       <input name="WriteAccessRequest" value="true"/>
       <output name="DbHandle" setvar="dbHandle"/>
       <return setvar="return"/>
     </invoke>
     <!-- Issue a conformity response.
       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>
       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.
       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 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="ModuleHandle" var="ModuleHandle"/>
       <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.
       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>
       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="ModuleHandle" var="ModuleHandle"/>
        <input name="BIRToStore_Form" var="__BioAPI_BIR_HANDLE_INPUT"/>
        <input name="BIRToStore_BIRHandle" var="newtemplate_handle"/>
        <input name="DbHandle" var="dbHandle"/>
        <input name="no_Uuid" value="false"/>
        <output name="Uuid" setvar="biruuid"/>
        <return setvar="return"/>
      </invoke>
      <!-- Issue a conformity response.
        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>
        function BioSPI_DbStoreBIR 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="ModuleHandle" var="ModuleHandle"/>
        <input name="DbHandle" var="dbHandle"/>
        <return setvar="return"/>
      </invoke>
      <!-- Issue a conformity response.
        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>
        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="ModuleHandle"/>
      <input name="dbName"/>
      <!-- Invoke the function BioSPI_DbDelete valid parameters -->
      <invoke function="BioSPI_DbDelete">
        <input name="ModuleHandle" var="_modulehandle"/>
        <input name="DbName" var="dbName"/>
        <return setvar="return"/>
      </invoke>
      <!-- Issue a conformity response.
        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="pass">
        <description>
        function BioSPI_DbDelete has returned BioAPI_OK
        </description>
        <equal_to var1="return" var2="__BioAPI_OK"/>
```

```
    </assert_condition>
  </activity>
</package>
```

16.2 Feature 1. BioAPI Registry

16.2.1 BioAPI Specification References:
1.11, 2.2, 4.2, 4.2.3

16.2.2 Comments:
See also item 104, "BSP provide all registry entries."

16.2.3 Assertion 1.1 BioAPI_Registry_Installation

16.2.3.1 Test Purpose:
To test entries to the module registry on BSP installation.

16.2.3.2 Test Scenario:
The BSP is not installed. The capabilities of the BSP, from the vendor's documentation, have been entered into a manifest. The BSP is then installed and the contents of the BSP Schema are compared with the manifest.

16.2.3.3 Expected Results:
The BSP Schema matches the manifest.

This assertion cannot be tested by calling BioSPI functions. The test lab will use some other means to determine if a BSP passes this assertion.

16.2.3.4 XML:
N.A.

16.2.4 Assertion 1.2 BioAPI_Registry_OpsSupported

16.2.4.1 Test Purpose:
To test that the BSP posts to the module registry whether it supports Local operation, Distributed operation, or both.

16.2.4.2 Test Scenario:
The BSP is installed.

16.2.4.3 Expected Results:
At least one of these three options of BioAPI_OPTIONS_MASK must be set, BioAPI_LOCAL_BSP, BioAPI_CLIENT_BSP or BioAPI_SERVER_BSP.

This assertion cannot be tested by calling BioSPI functions. The test lab will use some other means to determine if a BSP passes this assertion.

16.2.4.4 XML:
N.A.

16.3 Feature 2. BioSPI Module Load

16.3.1 BioAPI Specification References:
3.3.1.1 (P. 97)

16.3.2 Assertion 2.1 BioSPI_ModuleLoad_ValidParam

16.3.2.1 Test Purpose:
To test BioSPI_ModuleLoad with valid parameters.

16.3.2.2 Test Scenario:
Call BioSPI_ModuleLoad with valid parameters.

16.3.2.3 Expected Results:
Return code BioAPI_OK.

16.3.2.4 XML:

```
<package name="8ba15b00-d441-11d8-9669-0800200c9a66">
   <author>
      author
   </author>

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

   <assertion name="BioSPI_ModuleLoad_ValidParam" model="BSPTesting">
      <description>
         This assertion checks if calling BioSPI_ModuleLoad with valid input
parameters returns BioAPI_OK.
         The relevant text in BioAPI 1.1 is quoted below from subclauses 3.3.1.1
and 4.2.
         ____________________________________________________________
         BioAPI_RETURN BioAPI BioSPI_ModuleLoad
            (const void * Reserved,
            const BioAPI_UUID *BSPUuid,
            BioSPI_ModuleEventHandler BioAPINotifyCallback,
            void* BioAPINotifyCallbackCtx);

         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.
         ____________________________________________________________
         Section 3.3.1.1:
         This function completes the module initialization process between BioAPI
and the biometric service module.
         The BSPUuid identifies the invoked module.
         The BioAPINotifyCallback and BioAPINotifyCallbackCtx define a callback
and callback context respectively.
         The module must retain this information for later use.
         The module should use the callback to notify BioAPI of module events of
type BioAPI_MODULE_EVENT in any ongoing, attached sessions.
         Section 4.2:
         This function must be supported by both Verification and Identification
```

```
BSP.
        ____________________________________________________________

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

           1) Call BioSPI_ModuleLoad 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 fail, an UNDECIDED conformity
response is issued.
     </description>

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

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

     <!-- Context for the event notifications -->
     <input name="_BioAPINotifyCallbackCtx" />

     <!-- Invocation of the primary activity of this assertion with input
parameter values assigned from the assertion's parameters. -->
     <invoke activity="BioSPI_ModuleLoad">
        <input name="moduleUuid" var="_moduleUuid"/>
        <input name="BioAPIdNotifyCallback"
             var="_BioAPINotifyCallback" />
        <input name="BioAPINotifyCallbackCtx"
             var="_BioAPINotifyCallbackCtx" />
     </invoke>
   </assertion>

   <activity name="BioSPI_ModuleLoad">
     <input name="moduleUuid"/>
     <input name="BioAPIdNotifyCallback" />
     <input name="BioAPINotifyCallbackCtx" />

     <!-- Invoke the function BioSPI_ModuleLoad.  -->
     <invoke function="BioSPI_ModuleLoad">
        <input name="Reserved" value="0"/>
        <input name="BSPUuid" var="moduleUuid"/>
        <input name="BioAPINotifyCallback"
             var="BioAPIdNotifyCallback"/>
        <input name="BioAPINotifyCallbackCtx"
             var="BioAPINotifyCallbackCtx"/>
        <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_ModuleLoad has returned BioAPI_OK.
        </description>
        <equal_to var1="return" var2="__BioAPI_OK"/>
     </assert_condition>

     <!-- Invoke the function BioSPI_ModuleUnload. -->
```

```
     <invoke function="BioSPI_ModuleUnload">
        <input name="Reserved" value="0" />
        <input name="BSPUuid" var="moduleUuid"/>
        <input name="BioAPINotifyCallback" value="*"/>
        <input name="BioAPINotifyCallbackCtx" 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, otherwise a PASS conformity response
is issued.-->
     <assert_condition response_if_false="undecided">
        <description>
           The function BioSPI_ModuleUnload has returned BioAPI_OK.
        </description>
        <equal_to var1="return" var2="__BioAPI_OK"/>
     </assert_condition>
   </activity>

</package>
```

16.3.3 Assertion 2.2 BioSPI_ModuleLoad_InvalidUUID

16.3.3.1 Test Purpose:
To test BioSPI_ModuleLoad with an invalid UUID.

16.3.3.2 Test Scenario:
Call BioSPI_ModuleLoad with an invalid input parameter UUID.

16.3.3.3 Expected Results:
Return code BioAPIERR_H_FRAMEWORK_INVALID_UUID.

16.3.3.4 XML:

```
<package name="41d96a80-d441-11d8-9669-0800200c9a66">
   <author>
      author
   </author>

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

   <assertion name="BioSPI_ModuleLoad_InvalidUUID" model="BSPTesting">
      <description>
         This assertion checks if calling the function BioSPI_ModuleLoad with an
invalid input parameter UUID returns BioAPIERR_H_FRAMEWORK_INVALID_UUID.
         The relevant text in BioAPI 1.1 is quoted below from subclauses 3.3.1.1
and 2.4.5.
         ____________________________________________________________
         BioAPI_RETURN BioAPI BioSPI_ModuleLoad(const void * Reserved,
            const BioAPI_UUID *BSPUuid,
            BioSPI_ModuleEventHandler BioAPINotifyCallback,
            void* BioAPINotifyCallbackCtx);

         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.
        ____________________________________________________
        Section 3.3.1.1:
        The BSPUuid identifies the invoked module.
        Section 2.4.5:
        Errors: BioAPIERR_H_FRAMEWORK_INVALID_UUID
        ____________________________________________________

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

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

        If any of the intermediate operations fail, 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_ModuleLoad">
        <input name="moduleUuid" value="ffffffff-ffff-ffff-ffff-ffffffffffff"/>
     </invoke>
   </assertion>

   <activity name="BioSPI_ModuleLoad">
     <input name="moduleUuid"/>

     <!-- Invoke the function BioSPI_ModuleLoad. -->
     <invoke function="BioSPI_ModuleLoad">
        <input name="Reserved" value="0"/>
        <input name="BSPUuid" var="moduleUuid"/>
        <input name="BioAPINotifyCallback" value="*"/>
        <input name="BioAPINotifyCallbackCtx" 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_ModuleLoad has returned
BioAPIERR_H_FRAMEWORK_INVALID_UUID
        </description>
        <equal_to var1="return"
             var2="__BioAPIERR_H_FRAMEWORK_INVALID_UUID"/>
     </assert_condition>
   </activity>

</package>
```

16.4 Feature 3. BioSPI Module Unload

16.4.1 BioAPI Specification References:
3.3.1.2

16.4.2 **Assertion 3.1 BioSPI_ModuleUnload_ValidParam**

16.4.2.1 Test Purpose:
To test BioSPI_ModuleUnload with valid parameters.

16.4.2.2 Test Scenario:
BioSPI_ModuleLoad succeeded.

16.4.2.3 Expected Results:
Return code BioAPI_OK.

16.4.2.4 XML:

```
<package name="b8efc140-d442-11d8-9669-0800200c9a66">
   <author>
      author
   </author>

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

   <assertion name="BioSPI_ModuleUnload_ValidParam" model="BSPTesting">
      <description>
         This assertion checks if calling BioSPI_ModuleUnload with valid input
parameters returns BioAPI_OK.
         The relevant text in BioAPI 1.1 is quoted below from subclauses 3.3.1.2
and 4.2.
         ____________________________________________________________
         BioAPI_RETURN BioAPI BioSPI_ModuleUnload
            (const void * Reserved,
            const BioAPI_UUID *BSPUuid,
            BioSPI_ModuleEventHandler BioAPINotifyCallback,
            void* BioAPINotifyCallbackCtx);

         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_ModuleLoad.

         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.
         ____________________________________________________________
         Section 3.3.1.2:
         This function disables events and de-registers the BioAPI event-
notification function.
         Section 4.2:
         This function must be supported by both Verification and Identification
BSP.
         ____________________________________________________________
         In order to determine conformance with respect to the text above, the
following steps are performed:

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

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

```
     </description>

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

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

   <activity name="BioSPI_ModuleUnload">
     <input name="moduleUuid"/>

     <!-- Invoke the function BioSPI_ModuleLoad.  -->
     <invoke function="BioSPI_ModuleLoad">
        <input name="BSPUuid" var="moduleUuid"/>
        <input name="Reserved" value="0" />
        <input name="BioAPINotifyCallback" value="*"/>
        <input name="BioAPINotifyCallbackCtx" 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_ModuleLoad has returned BioAPI_OK
        </description>
        <equal_to var1="return" var2="__BioAPI_OK"/>
     </assert_condition>

     <!-- Invoke the function BioSPI_ModuleUnload. -->
     <invoke function="BioSPI_ModuleUnload">
        <input name="Reserved" value="0" />
        <input name="BSPUuid" var="moduleUuid"/>
        <input name="BioAPINotifyCallback" value="*"/>
        <input name="BioAPINotifyCallbackCtx" 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_ModuleUnload has returned BioAPI_OK
        </description>
        <equal_to var1="return" var2="__BioAPI_OK"/>
     </assert_condition>
   </activity>
</package>
```

16.4.3 **Assertion 3.2 BioSPI_ModuleUnload_Unmatch**

16.4.3.1 Test Purpose:

To test BioSPI_ModuleUnload unmatched with BioSPI_ModuleLoad.

16.4.3.2 Test Scenario:
Initial state, or calling ModuleUnload twice.

16.4.3.3 Expected Results:
Return code BioAPI_H_FRAMEWORK_MODULE_UNLOAD_FAILED.

16.4.3.4 XML:

```
<package name="6cc3c910-d442-11d8-9669-0800200c9a66">
   <author>
      author
   </author>

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

   <assertion name="BioSPI_ModuleUnload_Unmatch" model="BSPTesting">
      <description>
         This assertion checks if calling BioSPI_ModuleUnload without a matching
call to BioSPI_ModuleLoad returns BioAPIERR_H_FRAMEWORK_MODULE_NOT_LOADED.
         The relevant text in BioAPI 1.1 is quoted below from subclause 2.4.5.
         ____________________________________________________________
         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_ModuleLoad.
         ____________________________________________________________
         Section 2.4.5:
         Calls to BioAPI_ModuleUnload that are not matched with a previous call
to BioAPI_ModuleLoad result in an error.
         NOTE: The BioAPI SPECIFICATION IS AMBIGUOUS.
         The above statement is only for BioAPI_ModuleUnload, BioAPI 1.1 does not
specify the same statement for BioSPI_ModuleUnload.
         ____________________________________________________________

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

            1) Call BioSPI_ModuleLoad with valid input parameters.
            2) Call BioSPI_ModuleUnload with valid input parameters.
            3) Call BioSPI_ModuleUnload with valid input parameters again.
            4) Check the return value.  If the value is
BioAPIERR_H_FRAMEWORK_MODULE_NOT_LOADED, then the test pass, otherwise the test
failed.

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

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

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

   <activity name="BioSPI_ModuleUnload_Unmatch">
```

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

     <!-- Invoke the function BioSPI_ModuleLoad. -->
     <invoke function="BioSPI_ModuleLoad">
        <input name="Reserved" value="0"/>
        <input name="BSPUuid" var="moduleUuid"/>
        <input name="BioAPINotifyCallback" value="*"/>
        <input name="BioAPINotifyCallbackCtx" 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_ModuleLoad has returned BioAPI_OK
        </description>
        <equal_to var1="return" var2="__BioAPI_OK"/>
     </assert_condition>

     <!-- Invoke the function BioSPI_ModuleUnload. -->
     <invoke function="BioSPI_ModuleUnload">
        <input name="Reserved" value="0" />
        <input name="BSPUuid" var="moduleUuid"/>
        <input name="BioAPINotifyCallback" value="*"/>
        <input name="BioAPINotifyCallbackCtx" 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_ModuleUnload has returned BioAPI_OK
        </description>
        <equal_to var1="return" var2="__BioAPI_OK"/>
     </assert_condition>

     <!-- Invoke the function BioSPI_ModuleUnload again. -->
     <invoke function="BioSPI_ModuleUnload">
        <input name="Reserved" value="0" />
        <input name="BSPUuid" var="moduleUuid"/>
        <input name="BioAPINotifyCallback" value="*"/>
        <input name="BioAPINotifyCallbackCtx" 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, otherwise a PASS conformity response is issued.-->
     <assert_condition>
        <description>
          The function BioSPI_ModuleUnload has returned
BioAPIERR_H_FRAMEWORK_MODULE_NOT_LOADED.
        </description>
        <equal_to var1="return"
```

```
            var2="__BioAPIERR_H_FRAMEWORK_MODULE_NOT_LOADED"/>
      </assert_condition>
   </activity>

</package>
```

16.4.4 Assertion 3.3 BioSPI_ModuleUnload_InvalidUUID

16.4.4.1 Test Purpose:
To test BioSPI_ModuleUnload with an invalid UUID.

16.4.4.2 Test Scenario:
BioSPI_ModuleLoad succeeded.

16.4.4.3 Expected Results:
Return code BioAPI_H_FRAMEWORK_INVALID_UUID.

16.4.4.4 XML:

```
<package name="d3291f80-d441-11d8-9669-0800200c9a66">
   <author>
      author
   </author>
   <description>
      This package contains the assertion "BioSPI_ModuleUnload_InvalidUUID" (see
the "description" element of the assertion below).
   </description>
   <assertion name="BioSPI_ModuleUnload_InvalidUUID" model="BSPTesting">
      <description>
         This assertion checks if calling BioSPI_ModuleUnload with an invalid
input parameter UUID returns BioAPIERR_H_FRAMEWORK_INVALID_UUID.
         The relevant text in BioAPI 1.1 is quoted below from subclauses 3.3.1.2
and 2.4.5.
         ____________________________________________________________
         BioAPI_RETURN BioAPI BioSPI_ModuleUnload
            (const void * Reserved,
            const BioAPI_UUID *BSPUuid,
            BioSPI_ModuleEventHandler BioAPINotifyCallback,
            void* BioAPINotifyCallbackCtx);

         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_ModuleLoad.

         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.
         ____________________________________________________________
         Section 3.3.1.2:
         Parameters: BSPUuid (input) - The BioAPI_UUID of the invoked service
provider module.
         Section 2.4.5:
         Errors: BioAPIERR_H_FRAMEWORK_INVALID_UUID
         ____________________________________________________________

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

            1) Call BioSPI_ModuleLoad with valid input parameters.  This call is
expected to succeed.
            2) Call BioSPI_ModuleUnload with an invalid UUID.
```

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

        If any of the intermediate operations fail, an UNDECIDED conformity
response is issued.
      </description>
      <!-- UUID of the BSP under test -->
      <input name="_moduleUuid"/>
      <!-- Invocation of the primary activity of this assertion with input
parameter values assigned from the assertion's parameters. -->
      <invoke activity="BioSPI_ModuleUnload">
         <input name="moduleUuid" var="_moduleUuid"/>
      </invoke>
   </assertion>
   <activity name="BioSPI_ModuleUnload">
      <input name="moduleUuid"/>
      <!-- Invoke the function BioSPI_ModuleLoad. -->
      <invoke function="BioSPI_ModuleLoad">
         <input name="Reserved" value="0"/>
         <input name="BSPUuid" var="moduleUuid"/>
         <input name="BioAPINotifyCallback" value="*"/>
         <input name="BioAPINotifyCallbackCtx" 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_ModuleLoad has returned BioAPI_OK
         </description>
         <equal_to var1="return" var2="__BioAPI_OK"/>
      </assert_condition>
      <!-- Invoke the function BioSPI_ModuleUnload.-->
      <invoke function="BioSPI_ModuleUnload">
         <input name="Reserved" value="0"/>
         <input name="BSPUuid" value="ffffffff-ffff-ffff-ffff-ffffffffffff"/>
         <input name="BioAPINotifyCallback" value="*"/>
         <input name="BioAPINotifyCallbackCtx" 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_ModuleUnload has returned
BioAPIERR_H_FRAMEWORK_INVALID_UUID
         </description>
         <equal_to var1="return" var2="__BioAPIERR_H_FRAMEWORK_INVALID_UUID"/>
      </assert_condition>
      <!-- Invoke the function BioSPI_ModuleUnload.-->
      <invoke function="BioSPI_ModuleUnload">
         <input name="Reserved" value="0"/>
         <input name="BSPUuid" var="_moduleUuid"/>
         <input name="BioAPINotifyCallback" value="*"/>
         <input name="BioAPINotifyCallbackCtx" 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, otherwise a PASS conformity response
is issued.-->
     <assert_condition response_if_false="undecided">
        <description>
           The function BioSPI_ModuleUnload has returned BioAPI_OK.
        </description>
        <equal_to var1="return" var2="__BioAPI_OK"/>
     </assert_condition>
   </activity>
</package>
```

16.4.5 Assertion 3.4 BioSPI_ModuleUnload_Confirm

16.4.5.1 Test Purpose:
To test that the BSP is truly unloaded.

16.4.5.2 Test Scenario:
1) Call BioSPI_ModuleLoad with valid input parameters. The call is expected to succeed.
2) Call BioSPI_ModuleUnload with valid input parameters.
3) Call BioSPI_ModuleAttach with valid input parameters. It is expected to return BioAPIERR_H_FRAMEWORK_MODULE_NOT_LOADED.

16.4.5.3 Expected Results:
Return code BioAPI_OK.

16.4.5.4 XML:

```
<package name="ec706780-02a1-11d9-9669-0800200c9a66">
   <author>
      author
   </author>

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

   <assertion name="BioSPI_ModuleUnload_Confirm" model="BSPTesting">
      <description>
         This assertion checks if calling BioSPI_ModuleUnload truly unloads the
BSP.
         The relevant text in BioAPI 1.1 is quoted below from subclause 3.3.1.2.
         ____________________________________________________________
         BioAPI_RETURN BioAPI BioSPI_ModuleUnload
            (const void * Reserved,
            const BioAPI_UUID *BSPUuid,
            BioSPI_ModuleEventHandler BioAPINotifyCallback,
            void* BioAPINotifyCallbackCtx);

         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_ModuleLoad.

         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.
         ____________________________________________________________

         Section 3.3.1.2:
         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_ModuleLoad.
        ____________________________________________________________

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

           1) Call BioSPI_ModuleLoad with valid input parameters.  The call is
expected to succeed.
           2) Call BioSPI_ModuleUnload with valid input parameters.
           3) Call BioSPI_ModuleAttach with valid input parameters.  It is
expected to return BioAPIERR_H_FRAMEWORK_MODULE_NOT_LOADED.

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

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

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

   <activity name="BioSPI_ModuleUnload">
     <input name="moduleUuid"/>

     <!-- Invoke the function BioSPI_ModuleLoad. -->
     <invoke function="BioSPI_ModuleLoad">
        <input name="BSPUuid" var="moduleUuid"/>
        <input name="Reserved" value="0" />
        <input name="BioAPINotifyCallback" value="*"/>
        <input name="BioAPINotifyCallbackCtx" 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_ModuleLoad has returned BioAPI_OK
        </description>
        <equal_to var1="return" var2="__BioAPI_OK"/>
     </assert_condition>

     <!-- Invoke the function BioSPI_ModuleUnload.-->
        <invoke function="BioSPI_ModuleUnload">
        <input name="Reserved" value="0" />
        <input name="BSPUuid" var="moduleUuid"/>
        <input name="BioAPINotifyCallback" value="*"/>
        <input name="BioAPINotifyCallbackCtx" 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_ModuleUnload has returned BioAPI_OK
         </description>
         <equal_to var1="return" var2="__BioAPI_OK"/>
      </assert_condition>

      <!-- Invoke the function BioSPI_ModuleAttach to check if the BSP is no
longer loaded -->
      <invoke function="BioSPI_ModuleAttach">
         <input name="BSPUuid" var="moduleUuid"/>
         <input name="VersionMajor" value="1"/>
         <input name="VersionMinor" value="10"/>
         <input name="DeviceID" value="1"/>
         <input name="Reserved1" value="0"/>
         <input name="Reserved2" value="0"/>
         <input name="ModuleHandle" value="1"/>
         <input name="Reserved3" value="0"/>
         <input name="Reserved4" value="0"/>
         <input name="Reserved5" value="0"/>
         <input name="Reserved6" 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_ModuleAttach has returned
BioAPIERR_H_FRAMEWORK_MODULE_NOT_LOADED
         </description>
         <equal_to var1="return"
              var2="__BioAPIERR_H_FRAMEWORK_MODULE_NOT_LOADED"/>
      </assert_condition>
   </activity>

</package>
```

16.5 Feature 4. BioSPI Module Attach

16.5.1 BioAPI Specification References:
3.3.1.3

16.5.2 Assertion 4.1 BioSPI_ModuleAttach_ValidParam

16.5.2.1 Test Purpose:
To test BioSPI_ModuleAttach with valid parameters.

16.5.2.2 Test Scenario:
Call BioSPI_ModuleAttach with valid input parameters.

16.5.2.3 Expected Results:
Return code BioAPI_OK.

16.5.2.4 XML:

```
<package name="379ab8d0-d4dc-11d8-9669-0800200c9a66">
   <author>
      author
   </author>

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

   <assertion name="BioSPI_ModuleAttach_ValidParam" model="BSPTesting">
      <description>
         This assertion checks if a call to the function BioSPI_ModuleAttach with
valid input parameters returns BioAPI_OK.
         The relevant text in BioAPI 1.1 is quoted below from subclauses 3.3.1.3
and 4.2.
         ____________________________________________________
         BioAPI_RETURN BioAPI BioSPI_ModuleAttach
            (const BioAPI_UUID *BSPUuid,
            const BioAPI_VERSION *Version,
            BioAPI_DEVICE_ID DeviceID,
            uint32 Reserved1,
            uint32 Reserved2,
            BioAPI_HANDLE ModuleHandle,
            uint32 Reserved3,
            const void * Reserved4,
            const void * Reserved5,
            const void * Reserved6,
            const BioAPI_UPCALLS *Upcalls,
            BioAPI_MODULE_FUNCS_PTR *FuncTbl);

         The service module must verify compatibility with the system version
level specified by Version. If the version is not compatible, then this
function fails. The service module should perform all initializations required
to support the new attached session and should return a function table for the
SPI entry points that can be invoked by BioAPI in response to API invocations.
BioAPI uses this function table to dispatch requests on for the attach session
created by this function. Each attach session has its own function table.
         ____________________________________________________
         Section 3.3.1.3:
         This function is invoked by BioAPI once for each invocation of
BioAPI_ModuleAttach specifying the module identified by BSPUuid.
         Section 4.2:
         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) Check the return value of the function call.
            4) Detach and unload the BSP.

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

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

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

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

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

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

     <!-- Invocation of the primary activity of this assertion with input
parameter values assigned from the assertion's parameters. -->
     <invoke activity="LoadAndAttach" >
        <input name="moduleUuid" var="_moduleUuid"/>
        <input name="moduleVersionMajor" var="_moduleVersionMajor"/>
        <input name="moduleVersionMinor" var="_moduleVersionMinor"/>
        <input name="deviceIDOrNull" value="0"/>
        <input name="module" var="_modulehandle"/>
        <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="be0a1330-d59e-11d8-9669-0800200c9a66"
          function="BioSPI_ModuleEventHandler"/>
   </assertion>

   <activity name="LoadAndAttach">
     <input name="moduleUuid"/>
     <input name="moduleVersionMajor"/>
     <input name="moduleVersionMinor"/>
     <input name="deviceIDOrNull"/>
     <input name="module"/>
     <input name="eventtimeouttime"/>

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

     <!-- 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_ModuleLoad.-->
     <invoke function="BioSPI_ModuleLoad">
        <input name="Reserved" value="0"/>
        <input name="BSPUuid" var="moduleUuid"/>
        <input name="BioAPINotifyCallback" value="*"/>
        <input name="BioAPINotifyCallbackCtx" 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_ModuleLoad 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_ModuleAttach. -->
     <invoke function="BioSPI_ModuleAttach">
        <input name="BSPUuid" var="moduleUuid"/>
        <input name="VersionMajor" var="moduleVersionMajor"/>
        <input name="VersionMinor" var="moduleVersionMinor"/>
        <input name="DeviceID" var="_deviceID"/>
        <input name="Reserved1" value="0"/>
        <input name="Reserved2" value="0"/>
        <input name="ModuleHandle" var="module"/>
        <input name="Reserved3" value="0"/>
        <input name="Reserved4" value="0"/>
        <input name="Reserved5" value="0"/>
        <input name="Reserved6" 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 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_ModuleAttach 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="be0a1330-d59e-11d8-9669-0800200c9a66" >
        <input name="moduleUuid" var="moduleUuid"/>
        <input name="module" var="module"/>
     </invoke>
   </activity>
```

```
</package>
```

16.5.3 Assertion 4.2 BioSPI_ModuleAttach_InvalidUUID

16.5.3.1 Test Purpose:
To test BioSPI_ModuleAttach with an invalid UUID.

16.5.3.2 Test Scenario:
Call BioSPI_ModuleAttach with an invalid UUID.

16.5.3.3 Expected Results:
Return code BioAPI_H_FRAMEWORK_INVALID_UUID.

16.5.3.4 XML:

```
<package name="d55a9350-d4e2-11d8-9669-0800200c9a66">
   <author>
      author
   </author>

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

   <assertion name="BioSPI_ModuleAttach_InvalidUUID" model="BSPTesting">
      <description>
         This assertion checks if a call to the function BioSPI_ModuleAttach
returns BioAPIERR_H_FRAMEWORK_INVALID_UUID when the input UUID is invalid.
         The relevant text in BioAPI 1.1 is quoted below from subclauses 3.3.1.3
and 2.4.6.
         ____________________________________________________________
         BioAPI_RETURN BioAPI BioSPI_ModuleAttach
            (const BioAPI_UUID *BSPUuid,
            const BioAPI_VERSION *Version,
            BioAPI_DEVICE_ID DeviceID,
            uint32 Reserved1,
            uint32 Reserved2,
            BioAPI_HANDLE ModuleHandle,
            uint32 Reserved3,
            const void * Reserved4,
            const void * Reserved5,
            const void * Reserved6,
            const BioAPI_UPCALLS *Upcalls,
            BioAPI_MODULE_FUNCS_PTR *FuncTbl);

         The service module must verify compatibility with the system version
level specified by Version. If the version is not compatible, then this
function fails. The service module should perform all initializations required
to support the new attached session and should return a function table for the
SPI entry points that can be invoked by BioAPI in response to API invocations.
BioAPI uses this function table to dispatch requests on for the attach session
created by this function. Each attach session has its own function table.
         ____________________________________________________________
         Section 3.3.1.3:
         Parameters: BSPUuid (input) - The BioAPI_UUID of the invoked service
provider module.
         Section 2.4.6:
         Errors: BioAPIERR_H_FRAMEWORK_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 fail, an UNDECIDED conformity
response is issued.
     </description>

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

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

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

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

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

     <!-- Invocation of the primary activity of this assertion with input
parameter values assigned from the assertion's parameters. -->
     <invoke activity="LoadAndAttach" >
        <input name="moduleUuid" var="_moduleUuid"/>
        <input name="moduleVersionMajor" var="_moduleVersionMajor"/>
        <input name="moduleVersionMinor" var="_moduleVersionMinor"/>
        <input name="deviceIDOrNull" value="0"/>
        <input name="module" var="_modulehandle"/>
        <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="be0a1330-d59e-11d8-9669-0800200c9a66"
          function="BioSPI_ModuleEventHandler"/>
   </assertion>

   <activity name="LoadAndAttach">
      <input name="moduleUuid"/>
      <input name="moduleVersionMajor"/>
      <input name="moduleVersionMinor"/>
      <input name="deviceIDOrNull"/>
      <input name="module"/>
      <input name="eventtimeouttime"/>

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

      <!-- 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_ModuleLoad. -->
     <invoke function="BioSPI_ModuleLoad">
        <input name="Reserved" value="0"/>
        <input name="BSPUuid" var="moduleUuid"/>
        <input name="BioAPINotifyCallback" value="*"/>
        <input name="BioAPINotifyCallbackCtx" 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_ModuleLoad 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_ModuleAttach passing an invalid UUID. -->
     <invoke function="BioSPI_ModuleAttach">
        <input name="BSPUuid"
             value="ffffffff-ffff-ffff-ffff-ffffffffffff"/>
        <input name="VersionMajor" var="moduleVersionMajor"/>
        <input name="VersionMinor" var="moduleVersionMinor"/>
        <input name="DeviceID" var="_deviceID"/>
        <input name="Reserved1" value="0"/>
        <input name="Reserved2" value="0"/>
        <input name="ModuleHandle" var="module"/>
        <input name="Reserved3" value="0"/>
        <input name="Reserved4" value="0"/>
        <input name="Reserved5" value="0"/>
        <input name="Reserved6" 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_ModuleAttach has returned
BioAPIERR_H_FRAMEWORK_INVALID_UUID.
        </description>
        <equal_to var1="return"
             var2="__BioAPIERR_H_FRAMEWORK_INVALID_UUID"/>
     </assert_condition>

     <!-- Invoke the function BioSPI_ModuleUnload -->
     <invoke function="BioSPI_ModuleUnload" >
        <input name="Reserved" value="0" />
        <input name="BSPUuid" var="moduleUuid" />
        <input name="BioAPINotifyCallback" value="*" />
        <input name="BioAPINotifyCallbackCtx" 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, otherwise a PASS conformity response
is issued.-->
     <assert_condition response_if_false="undecided"
          break_if_false="true">
        <description>
          The function BioSPI_ModuleUnload has returned BioAPI_OK
        </description>
        <equal_to var1="return" var2="__BioAPI_OK"/>
     </assert_condition>
   </activity>

</package>
```

16.5.4 Assertion 4.3 BioSPI_ModuleAttach_InvalidVersion

16.5.4.1 Test Purpose:
To test BioSPI_ModuleAttach with an invalid BioAPI Version.

16.5.4.2 Test Scenario:
Call BioSPI_ModuleAttach with an invalid version number.

16.5.4.3 Expected Results:
Return code other than BioAPI_OK.

16.5.4.4 XML:

```
<package name="96a0f5c0-d4e5-11d8-9669-0800200c9a66">
   <author>
      author
   </author>

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

   <assertion name="BioSPI_ModuleAttach_InvalidVersion" model="BSPTesting">
      <description>
         This assertion checks if the function BioSPI_ModuleAttach returns
BioAPIERR_H_FRAMEWORK_INCOMPATIBLE_VERSION when called with an invalid version
number.
         The relevant text in BioAPI 1.1 is quoted below from subclause 3.3.1.3.
         ____________________________________________________________
         BioAPI_RETURN BioAPI BioSPI_ModuleAttach
```

```
          (const BioAPI_UUID *BSPUuid,
          const BioAPI_VERSION *Version,
          BioAPI_DEVICE_ID DeviceID,
          uint32 Reserved1,
          uint32 Reserved2,
          BioAPI_HANDLE ModuleHandle,
          uint32 Reserved3,
          const void * Reserved4,
          const void * Reserved5,
          const void * Reserved6,
          const BioAPI_UPCALLS *Upcalls,
          BioAPI_MODULE_FUNCS_PTR *FuncTbl);
```

The service module must verify compatibility with the system version level specified by Version. If the version is not compatible, then this function fails. The service module should perform all initializations required to support the new attached session and should return a function table for the SPI entry points that can be invoked by BioAPI in response to API invocations. BioAPI uses this function table to dispatch requests on for the attach session created by this function. Each attach session has its own function table.

Section 3.3.1.3:
This function attaches the service provider module and verifies that the version of the module expected by the application is compatible with the version on the system.

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 fail, an UNDECIDED conformity response is issued.

```
     </description>

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

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

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

     <!-- Invocation of the primary activity of this assertion with input
parameter values assigned from the assertion's parameters. -->
     <invoke activity="LoadAndAttach" >
        <input name="moduleUuid" var="_moduleUuid"/>
        <input name="deviceIDOrNull" value="0"/>
        <input name="module" var="_modulehandle"/>
        <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="be0a1330-d59e-11d8-9669-0800200c9a66"
          function="BioSPI_ModuleEventHandler"/>
   </assertion>
```

```
   <activity name="LoadAndAttach">
      <input name="moduleUuid"/>
      <input name="deviceIDOrNull"/>
      <input name="module"/>
      <input name="eventtimeouttime"/>

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

      <!-- 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_ModuleLoad. -->
      <invoke function="BioSPI_ModuleLoad">
         <input name="Reserved" value="0"/>
         <input name="BSPUuid" var="moduleUuid"/>
         <input name="BioAPINotifyCallback" value="*"/>
         <input name="BioAPINotifyCallbackCtx" 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_ModuleLoad 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_ModuleAttach passing an invalid version
number. -->
         <invoke function="BioSPI_ModuleAttach">
         <input name="BSPUuid" var="moduleUuid"/>
         <input name="VersionMajor" value="2"/>
```

```
        <input name="VersionMinor" value="3"/>
        <input name="DeviceID" var="_deviceID"/>
        <input name="Reserved1" value="0"/>
        <input name="Reserved2" value="0"/>
        <input name="ModuleHandle" var="module"/>
        <input name="Reserved3" value="0"/>
        <input name="Reserved4" value="0"/>
        <input name="Reserved5" value="0"/>
        <input name="Reserved6" 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_ModuleAttach has returned
BioAPIERR_H_FRAMEWORK_INCOMPATIBLE_VERSION
        </description>
        <equal_to var1="return"
             var2="__BioAPIERR_H_FRAMEWORK_INCOMPATIBLE_VERSION"/>
     </assert_condition>

     <!-- Invoke the function BioSPI_ModuleUnload -->
     <invoke function="BioSPI_ModuleUnload" >
        <input name="Reserved" value="0" />
        <input name="BSPUuid" var="moduleUuid" />
        <input name="BioAPINotifyCallback" value="*" />
        <input name="BioAPINotifyCallbackCtx" 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, otherwise a PASS conformity response
is issued.-->
     <assert_condition response_if_false="undecided"
          break_if_false="true">
        <description>
          The function BioSPI_ModuleUnload has returned BioAPI_OK
        </description>
        <equal_to var1="return" var2="__BioAPI_OK"/>
     </assert_condition>
   </activity>

</package>
```

16.5.5 Assertion 4.4 BioSPI_ModuleAttach_InvalidModuleHandle

16.5.5.1 Test Purpose:
To test BioSPI_ModuleAttach with an invalid module handle.

16.5.5.2 Test Scenario:
Call BioSPI_ModuleAttach with an invalid module handle.

16.5.5.3 Expected Results:
Return code BioAPIERR_H_FRAMEWORK_INVALID_MODULE_HANDLE.

16.5.5.4 XML:

```
<package name="6acd2480-d7f8-11d8-9669-0800200c9a66">
   <author>
      author
   </author>
   <description>
      This package contains the assertion
"BioSPI_ModuleAttach_InvalidModuleHandle" (see the "description" element of the
assertion below).
   </description>
   <assertion name="BioSPI_ModuleAttach_InvalidModuleHandle" model="BSPTesting">
      <description>
         This assertion checks if the function BioSPI_ModuleAttach returns an
error when called with an invalid module handle.
         The relevant text in BioAPI 1.1 is quoted below from subclause 3.3.1.3.
         ____________________________________________________________
         BioAPI_RETURN BioAPI BioSPI_ModuleAttach
            (const BioAPI_UUID *BSPUuid,
            const BioAPI_VERSION *Version,
            BioAPI_DEVICE_ID DeviceID,
            uint32 Reserved1,
            uint32 Reserved2,
            BioAPI_HANDLE ModuleHandle,
            uint32 Reserved3,
            const void * Reserved4,
            const void * Reserved5,
            const void * Reserved6,
            const BioAPI_UPCALLS *Upcalls,
            BioAPI_MODULE_FUNCS_PTR *FuncTbl);

         The service module must verify compatibility with the system version
level specified by Version. If the version is not compatible, then this
function fails. The service module should perform all initializations required
to support the new attached session and should return a function table for the
SPI entry points that can be invoked by BioAPI in response to API invocations.
BioAPI uses this function table to dispatch requests on for the attach session
created by this function. Each attach session has its own function table.
         ____________________________________________________________
         Section 3.3.1.3:
         Parameters: ModuleHandle (input) - The BioAPI_HANDLE value assigned by
BioAPI and associated with the attach session being created 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 with a module handle value of 0 (see
note below).
            3) Check the return value of the function call.
            4) Unload the BSP under test.

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

         NOTE - BioAPI does not specify any "invalid" value for the type
BioAPI_HANDLE.  However, many BSPs assume that 0 is an invalid module handle
value.  Based on this common behavior, the assertion
BioSPI_ModuleAttach_InvalidModuleHandle invokes the function
BioSPI_ModuleAttach passing 0 as the module handle value and expects that the
BSP returns BioAPIERR_H_FRAMEWORK_INVALID_MODULE_HANDLE.

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

```
      <input name="_moduleUuid"/>
      <!-- Major version number of the BSP under test -->
      <input name="_moduleVersionMajor"/>
      <!-- Minor version number of the BSP under test -->
      <input name="_moduleVersionMinor"/>
      <!-- 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="moduleUuid" var="_moduleUuid"/>
         <input name="moduleVersionMajor" var="_moduleVersionMajor"/>
         <input name="moduleVersionMinor" var="_moduleVersionMinor"/>
         <input name="deviceIDOrNull" 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="be0a1330-d59e-11d8-9669-
0800200c9a66" function="BioSPI_ModuleEventHandler"/>
   </assertion>
   <activity name="LoadAndAttach">
      <input name="moduleUuid"/>
      <input name="moduleVersionMajor"/>
      <input name="moduleVersionMinor"/>
      <input name="deviceIDOrNull"/>
      <input name="eventtimeouttime"/>
      <!-- Initialize the global variable "_deviceIDOrNull" to make it available
to the activity "EventHandler" -->
      <set name="_deviceIDOrNull" var="deviceIDOrNull"/>
      <!-- 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_ModuleLoad. -->
      <invoke function="BioSPI_ModuleLoad">
         <input name="Reserved" value="0"/>
         <input name="BSPUuid" var="moduleUuid"/>
         <input name="BioAPINotifyCallback" value="*"/>
         <input name="BioAPINotifyCallbackCtx" 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_ModuleLoad 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_ModuleAttach, passing an invalid value for
the input parameter "ModuleHandle". -->
     <invoke function="BioSPI_ModuleAttach">
        <input name="BSPUuid" var="moduleUuid"/>
        <input name="VersionMajor" var="moduleVersionMajor"/>
        <input name="VersionMinor" var="moduleVersionMinor"/>
        <input name="DeviceID" var="_deviceID"/>
        <input name="Reserved1" value="0"/>
        <input name="Reserved2" value="0"/>
        <input name="ModuleHandle" value="0"/>
        <input name="Reserved3" value="0"/>
        <input name="Reserved4" value="0"/>
        <input name="Reserved5" value="0"/>
        <input name="Reserved6" 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_ModuleAttach has returned
BioAPIERR_H_FRAMEWORK_INVALID_MODULE_HANDLE
        </description>
        <equal_to var1="return"
var2="__BioAPIERR_H_FRAMEWORK_INVALID_MODULE_HANDLE"/>
     </assert_condition>
     <!-- Invoke the function BioSPI_ModuleUnload -->
     <invoke function="BioSPI_ModuleUnload">
        <input name="Reserved" value="0"/>
        <input name="BSPUuid" var="moduleUuid"/>
        <input name="BioAPINotifyCallback" value="*"/>
        <input name="BioAPINotifyCallbackCtx" 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, otherwise a PASS conformity response
is issued.-->
     <assert_condition response_if_false="undecided" break_if_false="true">
        <description>
           The function BioSPI_ModuleUnload has returned BioAPI_OK
        </description>
        <equal_to var1="return" var2="__BioAPI_OK"/>
     </assert_condition>
   </activity>
</package>
```

16.6 Feature 5. BioSPI Module Detach

16.6.1 BioAPI Specification References:
3.3.1.4

16.6.2 **Assertion 5.1 BioSPI_ModuleDetach_ValidParam**

16.6.2.1 Test Purpose:
To test BioSPI_ModuleDetach with valid parameters.

16.6.2.2 Test Scenario:
Call BioSPI_ModuleDetach with valid parameters.

16.6.2.3 Expected Results:
Return code BioAPI_OK.

16.6.2.4 XML:

```
<package name="120ec850-d4fe-11d8-9669-0800200c9a66">
   <author>
      author
   </author>

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

   <assertion name="BioSPI_ModuleDetach_ValidParam" model="BSPTesting">
      <description>
         This assertion checks if a call to the function BioSPI_ModuleDetach with
a valid module handle returns BioAPI_OK.
         The relevant text in BioAPI 1.1 is quoted below from subclauses 3.3.1.4
and 4.2.
         ____________________________________________________________
         BioAPI_RETURN BioAPI BioSPI_ModuleDetach
            (BioAPI_HANDLE ModuleHandle);

         This function is invoked by BioAPI once for each invocation of
BioAPI_ModuleDetach specifying the attach-session identified by ModuleHandle.
The function entry point for BioSPI_ModuleDetach is included in the module
function table BioAPI_MODULE_FUNCS returned to BioAPI as output of a successful
BioSPI_ModuleAttach.
         The service module is expected to perform all cleanup operations
associated with the specified attach handle.
         ____________________________________________________________
         Section 3.3.1.4:
         This function is invoked by BioAPI once for each invocation of
BioAPI_ModuleDetach specifying the attach-session identified by ModuleHandle.
         Section 4.2:
         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) Detach the BSP under test using the module handle.
            4) Check the return value.

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

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

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

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

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

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

     <!-- Invocation of the primary activity of this assertion with input
parameter values assigned from the assertion's parameters. -->
     <invoke activity="LoadAndAttachAndDetach" >
        <input name="moduleUuid" var="_moduleUuid"/>
        <input name="moduleVersionMajor" var="_moduleVersionMajor"/>
        <input name="moduleVersionMinor" var="_moduleVersionMinor"/>
        <input name="deviceIDOrNull" value="0"/>
        <input name="module" var="_modulehandle"/>
        <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="be0a1330-d59e-11d8-9669-0800200c9a66"
          function="BioSPI_ModuleEventHandler"/>
   </assertion>

   <activity name="LoadAndAttachAndDetach">
     <input name="moduleUuid"/>
     <input name="moduleVersionMajor"/>
     <input name="moduleVersionMinor"/>
     <input name="deviceIDOrNull"/>
     <input name="module"/>
     <input name="eventtimeouttime"/>

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

     <!-- 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_ModuleLoad.-->
     <invoke function="BioSPI_ModuleLoad">
        <input name="Reserved" value="0"/>
        <input name="BSPUuid" var="moduleUuid"/>
        <input name="BioAPINotifyCallback" value="*"/>
        <input name="BioAPINotifyCallbackCtx" 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_ModuleLoad 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_ModuleAttach. -->
     <invoke function="BioSPI_ModuleAttach">
        <input name="BSPUuid" var="moduleUuid"/>
        <input name="VersionMajor" var="moduleVersionMajor"/>
        <input name="VersionMinor" var="moduleVersionMinor"/>
        <input name="DeviceID" var="_deviceID"/>
        <input name="Reserved1" value="0"/>
        <input name="Reserved2" value="0"/>
        <input name="ModuleHandle" var="module"/>
        <input name="Reserved3" value="0"/>
        <input name="Reserved4" value="0"/>
        <input name="Reserved5" value="0"/>
        <input name="Reserved6" value="0"/>
        <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_ModuleAttach has returned BioAPI_OK.
        </description>
        <equal_to var1="return" var2="__BioAPI_OK"/>
     </assert_condition>

     <!-- Invoke the function BioSPI_ModuleDetach -->
     <invoke function="BioSPI_ModuleDetach">
        <input name="ModuleHandle" var="module"/>
        <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_ModuleDetach has returned BioAPI_OK
        </description>
        <equal_to var1="return" var2="__BioAPI_OK"/>
     </assert_condition>

     <!-- Invoke the function BioSPI_ModuleUnload -->
     <invoke function="BioSPI_ModuleUnload" >
        <input name="Reserved" value="0"/>
        <input name="BSPUuid" var="moduleUuid"/>
        <input name="BioAPINotifyCallback" value="*"/>
        <input name="BioAPINotifyCallbackCtx" 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_ModuleUnload has returned BioAPI_OK.
        </description>
        <equal_to var1="return" var2="__BioAPI_OK"/>
     </assert_condition>
   </activity>

</package>
```

16.6.3 Assertion 5.2 BioSPI_ModuleDetach_InvalidModuleHandle

16.6.3.1 Test Purpose:
To test BioAPI_ModuleDetach with an invalid module handle.

16.6.3.2 Test Scenario:
Call BioSPI_ModuleDetach with an invalid module handle.

16.6.3.3 Expected Results:
Return code BioAPIERR_H_FRAMEWORK_INVALID_MODULE_HANDLE.

16.6.3.4 XML:

```
<package name="efcbc370-d52e-11d8-9669-0800200c9a66">
   <author>
      author
   </author>

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

   <assertion name="BioSPI_ModuleDetach_InvalidModuleHandle" model="BSPTesting">
      <description>
```

```
        This assertion checks if a call to BioSPI_ModuleDetach with an invalid
module handle returns an error.
        The relevant text in BioAPI 1.1 is quoted below from subclause 3.3.1.4.
        ____________________________________________________________
        BioAPI_RETURN BioAPI BioSPI_ModuleDetach
           (BioAPI_HANDLE ModuleHandle);

        This function is invoked by BioAPI once for each invocation of
BioAPI_ModuleDetach specifying the attach-session identified by ModuleHandle.
The function entry point for BioSPI_ModuleDetach is included in the module
function table BioAPI_MODULE_FUNCS returned to BioAPI as output of a successful
BioSPI_ModuleAttach. The service module must perform all cleanup operations
associated with the specified attach handle.
        ____________________________________________________________
        Section 3.3.1.4:
        Parameters: ModuleHandle (input) - The BioAPI_HANDLE value 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 fail, an UNDECIDED conformity
response is issued.
     </description>

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

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

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

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

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

     <!-- Invocation of the primary activity of this assertion with input
parameter values assigned from the assertion's parameters. -->
     <invoke activity="LoadAndAttachAndDetach" >
        <input name="moduleUuid" var="_moduleUuid"/>
        <input name="moduleVersionMajor" var="_moduleVersionMajor"/>
        <input name="moduleVersionMinor" var="_moduleVersionMinor"/>
        <input name="deviceIDOrNull" value="0"/>
        <input name="module" var="_modulehandle"/>
        <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="be0a1330-d59e-11d8-9669-0800200c9a66"
          function="BioSPI_ModuleEventHandler"/>
```

```
   </assertion>

   <activity name="LoadAndAttachAndDetach">
      <input name="moduleUuid"/>
      <input name="moduleVersionMajor"/>
      <input name="moduleVersionMinor"/>
      <input name="deviceIDOrNull"/>
      <input name="module"/>
      <input name="eventtimeouttime"/>

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

      <!-- 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_ModuleLoad. -->
      <invoke function="BioSPI_ModuleLoad">
         <input name="Reserved" value="0"/>
         <input name="BSPUuid" var="moduleUuid"/>
         <input name="BioAPINotifyCallback" value="*"/>
         <input name="BioAPINotifyCallbackCtx" 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_ModuleLoad 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_ModuleAttach. -->
      <invoke function="BioSPI_ModuleAttach">
```

```
        <input name="BSPUuid" var="moduleUuid"/>
        <input name="VersionMajor" var="moduleVersionMajor"/>
        <input name="VersionMinor" var="moduleVersionMinor"/>
        <input name="DeviceID" var="_deviceID"/>
        <input name="Reserved1" value="0"/>
        <input name="Reserved2" value="0"/>
        <input name="ModuleHandle" var="module"/>
        <input name="Reserved3" value="0"/>
        <input name="Reserved4" value="0"/>
        <input name="Reserved5" value="0"/>
        <input name="Reserved6" value="0"/>
        <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_ModuleAttach has returned BioAPI_OK
        </description>
        <equal_to var1="return" var2="__BioAPI_OK"/>
     </assert_condition>

     <!-- Invoke the function BioSPI_ModuleDetach passing an invalid module
handle. -->
     <invoke function="BioSPI_ModuleDetach">
        <input name="ModuleHandle" 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_ModuleDetach has returned
BioAPIERR_H_FRAMEWORK_INVALID_MODULE_HANDLE
        </description>
        <equal_to var1="return"
             var2="__BioAPIERR_H_FRAMEWORK_INVALID_MODULE_HANDLE"/>
     </assert_condition>

     <!-- Invoke the functions BioSPI_ModuleDetach and BioSPI_ModuleUnload -->
     <invoke activity="DetachAndUnload"
          package="be0a1330-d59e-11d8-9669-0800200c9a66" >
        <input name="moduleUuid" var="moduleUuid" />
        <input name="module" var="module" />
     </invoke>
   </activity>

</package>
```

16.6.4 Assertion 5.3 BioSPI_ModuleDetach_Confirm

16.6.4.1 Test Purpose:

To test BioSPI_ModuelDetach truly terminates the attach session.

16.6.4.2 Test 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_Capture. This function is expected to fail with the error BioAPIERR_H_FRAMEWORK_MODULE_NOT_LOADED.

16.6.4.3 Expected Results:
The call to BioSPI_Capture is expected to fail with the error BioAPIERR_H_FRAMEWORK_MODULE_NOT_LOADED.

16.6.4.4 XML:

```
<package name="2c8ce250-0732-11d9-9669-0800200c9a66">
   <author>
      author
   </author>
   <description>
      This package contains the assertion "BioSPI_ModuleDetach_Confirm" (see the
"description" element of the assertion below).
   </description>
   <assertion name="BioSPI_ModuleDetach_Confirm" model="BSPTesting">
      <description>
         This assertion checks if a call to the function BioSPI_ModuleDetach
truly terminates the attach session.
         The relevant text in BioAPI 1.1 is quoted below from subclause 3.3.1.4.
         ____________________________________________________________
         BioAPI_RETURN BioAPI BioSPI_ModuleDetach
            (BioAPI_HANDLE ModuleHandle);

         This function is invoked by BioAPI once for each invocation of
BioAPI_ModuleDetach specifying the attach-session identified by ModuleHandle.
The function entry point for BioSPI_ModuleDetach is included in the module
function table BioAPI_MODULE_FUNCS returned to BioAPI as output of a successful
BioSPI_ModuleAttach.
         The service module must perform all cleanup operations associated with
the specified attach handle.
         ____________________________________________________________
         Section 3.3.1.4:
         The service module must perform all cleanup operations associated with
the specified attach 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_Capture. This function is expected to fail with the
error BioAPIERR_H_FRAMEWORK_MODULE_NOT_LOADED.

         If any of the intermediate operations fail, an UNDECIDED conformity
response is issued.
      </description>
      <!-- UUID of the BSP under test -->
      <input name="_moduleUuid"/>
      <!-- Major version number of the BSP under test -->
      <input name="_moduleVersionMajor"/>
      <!-- Minor version number of the BSP under test -->
```

```
      <input name="_moduleVersionMinor"/>
      <!-- Timeout for the BioAPI_NOTIFY_INSERT event -->
      <input name="_inserttimeout"/>
      <!-- Module handle for attached BSP -->
      <input name="_modulehandle"/>
      <!-- Invocation of the primary activity of this assertion with input
parameter values assigned from the assertion's parameters. -->
      <invoke activity="LoadAndAttachAndDetach">
         <input name="moduleUuid" var="_moduleUuid"/>
         <input name="moduleVersionMajor" var="_moduleVersionMajor"/>
         <input name="moduleVersionMinor" var="_moduleVersionMinor"/>
         <input name="deviceIDOrNull" value="0"/>
         <input name="module" var="_modulehandle"/>
         <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="be0a1330-d59e-11d8-9669-
0800200c9a66" function="BioSPI_ModuleEventHandler"/>
   </assertion>
   <activity name="LoadAndAttachAndDetach">
      <input name="moduleUuid"/>
      <input name="moduleVersionMajor"/>
      <input name="moduleVersionMinor"/>
      <input name="deviceIDOrNull"/>
      <input name="module"/>
      <input name="eventtimeouttime"/>
      <!-- Initialize the global variable "_deviceIDOrNull" to make it available
to the activity "EventHandler" -->
      <set name="_deviceIDOrNull" var="deviceIDOrNull"/>
      <!-- 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_ModuleLoad. -->
      <invoke function="BioSPI_ModuleLoad">
         <input name="Reserved" value="0"/>
         <input name="BSPUuid" var="moduleUuid"/>
         <input name="BioAPINotifyCallback" value="*"/>
         <input name="BioAPINotifyCallbackCtx" 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_ModuleLoad 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_ModuleAttach. -->
     <invoke function="BioSPI_ModuleAttach">
        <input name="BSPUuid" var="moduleUuid"/>
        <input name="VersionMajor" var="moduleVersionMajor"/>
        <input name="VersionMinor" var="moduleVersionMinor"/>
        <input name="DeviceID" var="_deviceID"/>
        <input name="Reserved1" value="0"/>
        <input name="Reserved2" value="0"/>
        <input name="ModuleHandle" var="module"/>
        <input name="Reserved3" value="0"/>
        <input name="Reserved4" value="0"/>
        <input name="Reserved5" value="0"/>
        <input name="Reserved6" value="0"/>
        <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_ModuleAttach has returned BioAPI_OK.
        </description>
        <equal_to var1="return" var2="__BioAPI_OK"/>
     </assert_condition>
     <!-- Invoke the function BioSPI_ModuleDetach. -->
     <invoke function="BioSPI_ModuleDetach">
        <input name="ModuleHandle" var="module"/>
        <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_ModuleDetach 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="ModuleHandle" var="module"/>
        <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, a
FAIL conformity response is issued, otherwise a PASS conformity response is
issued.-->
     <assert_condition>
        <description>
```

```
          Invoking BioSPI_Enroll after BioSPI_ModuleDetach has returned an
error.
        </description>
        <not_equal_to var1="return" var2="__BioAPI_OK"/>
     </assert_condition>
     <!-- Invoke the function BioSPI_ModuleUnload. -->
     <invoke function="BioSPI_ModuleUnload">
        <input name="Reserved" value="0"/>
        <input name="BSPUuid" var="moduleUuid"/>
        <input name="BioAPINotifyCallback" value="*"/>
        <input name="BioAPINotifyCallbackCtx" 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, otherwise a PASS conformity response
is issued.-->
     <assert_condition response_if_false="undecided">
        <description>
          The function BioSPI_ModuleUnload has returned BioAPI_OK.
        </description>
        <equal_to var1="return" var2="__BioAPI_OK"/>
     </assert_condition>
   </activity>
</package>
```

16.7 Feature 6. BioSPI Free BIR Handle

16.7.1 BioAPI Specification References:
3.3.2.1

16.7.2 Assertion 6.1 BioSPI_FreeBIRHandle_InvalidModuleHandle

16.7.2.1 Test Purpose:
To test BioSPI_FreeBIRHandle with an invalid module handle.

16.7.2.2 Test Scenario:
Call BioSPI_FreeBIRHandle with an invalid module handle.

16.7.2.3 Expected Results:
Return code other than BioAPI_OK.

16.7.2.4 XML:

```
<package name="7c822430-0595-11d9-9669-0800200c9a66">
   <author>
      author
   </author>
   <description>
      This package contains the assertion
"BioSPI_FreeBIRHandle_InvalidModuleHandle" (see the "description" element of
the assertion below).
   </description>
   <assertion name="BioSPI_FreeBIRHandle_InvalidModuleHandle"
model="BSPTesting">
      <description>
         This assertion checks if calling BioSPI_FreeBITHandle with an invalid
module handle returns an error.
```

```
        The relevant text in BioAPI 1.1 is quoted below from subclause 3.3.2.1.
        ____________________________________________________________
        BioAPI_RETURN BioAPI BioSPI_FreeBIRHandle
           (BioAPI_HANDLE ModuleHandle,
           BioAPI_BIR_HANDLE Handle);

        Frees the memory and resources associated with the specified BIR Handle.
The associated BIR is no longer referenceable through that handle. If
necessary, the application must make the BIR persistent either in a BSPmanaged
database or an application-managed database before freeing the handle.
        ____________________________________________________________
        Section 3.3.2.1:
        Parameters: ModuleHandle (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) Enroll to obtain a BIR handle
           4) Call BioSPI_FreeBIRHandle with an invalid module handle
           5) Check the return code.

        If any of the intermediate operations fail, an UNDECIDED conformity
response is issued.
      </description>
      <!-- UUID of the BSP under test -->
      <input name="_moduleUuid"/>
      <!-- Major version number of the BSP under test -->
      <input name="_moduleVersionMajor"/>
      <!-- Minor version number of the BSP under test -->
      <input name="_moduleVersionMinor"/>
      <!-- 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="moduleUuid" var="_moduleUuid"/>
         <input name="moduleVersionMajor" var="_moduleVersionMajor"/>
         <input name="moduleVersionMinor" var="_moduleVersionMinor"/>
         <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="be0a1330-d59e-11d8-9669-
0800200c9a66" function="BioSPI_ModuleEventHandler"/>
   </assertion>
   <activity name="BioSPI_FreeBIRHandle">
      <input name="moduleUuid"/>
      <input name="moduleVersionMajor"/>
```

```
     <input name="moduleVersionMinor"/>
     <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="_modulehandle" value="1"/>
     <!-- Invoke the functions BioSPI_ModuleLoad and BioSPI_ModuleAttach
exposed by the BSP under test. -->
     <invoke activity="LoadAndAttach" package="be0a1330-d59e-11d8-9669-
0800200c9a66" break_on_break="true">
        <input name="moduleUuid" var="moduleUuid"/>
        <input name="moduleVersionMajor" var="moduleVersionMajor"/>
        <input name="moduleVersionMinor" var="moduleVersionMinor"/>
        <input name="deviceIDOrNull" value="0"/>
        <input name="module" var="_modulehandle"/>
        <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="ModuleHandle" var="_modulehandle"/>
        <input name="Purpose"
var="__BioAPI_PURPOSE_ENROLL_FOR_VERIFICATION_ONLY"/>
        <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="ModuleHandle" 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_H_FRAMEWORK_INVALID_MODULE_HANDLE
        </description>
        <equal_to var1="return"
var2="__BioAPIERR_H_FRAMEWORK_INVALID_MODULE_HANDLE"/>
      </assert_condition>
      <!-- Invoke the functions BioSPI_ModuleDetach and BioSPI_ModuleUnload -->
      <invoke activity="DetachAndUnload" package="be0a1330-d59e-11d8-9669-
0800200c9a66">
        <input name="moduleUuid" var="moduleUuid"/>
        <input name="module" var="_modulehandle"/>
      </invoke>
   </activity>
</package>
```

16.7.3 Assertion 6.2 BioSPI_FreeBIRHandle_InvalidBIRHandle

16.7.3.1 Test Purpose:
To test BioSPI_FreeBIRHandle with an invalid BIR handle.

16.7.3.2 Test Scenario:
Call BioSPI_FreeBIRHandle with an invalid BIR handle.

16.7.3.3 Expected Results:
Return code other than BioAPI_OK.

16.7.3.4 XML:

```
<package name="67052160-d5c6-11d8-9669-0800200c9a66">
   <author>
      author
   </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 1.1 is quoted below from subclause 3.3.2.1.
         ____________________________________________________________
         BioAPI_RETURN BioAPI BioSPI_FreeBIRHandle
            (BioAPI_HANDLE ModuleHandle,
            BioAPI_BIR_HANDLE Handle);

         Frees the memory and resources associated with the specified BIR Handle.
The associated BIR is no longer referenceable through that handle. If
necessary, the application must make the BIR persistent either in a BSPmanaged
database or an application-managed database before freeing the handle.
         ____________________________________________________________
         Section 3.3.2.1:
```

```
        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 fail, an UNDECIDED conformity
response is issued.
     </description>
     <!-- UUID of the BSP under test -->
     <input name="_moduleUuid"/>
     <!-- Major version number of the BSP under test -->
     <input name="_moduleVersionMajor"/>
     <!-- Minor version number of the BSP under test -->
     <input name="_moduleVersionMinor"/>
     <!-- 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="moduleUuid" var="_moduleUuid"/>
        <input name="moduleVersionMajor" var="_moduleVersionMajor"/>
        <input name="moduleVersionMinor" var="_moduleVersionMinor"/>
        <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="be0a1330-d59e-11d8-9669-
0800200c9a66" function="BioSPI_ModuleEventHandler"/>
   </assertion>
   <activity name="BioSPI_FreeBIRHandle">
     <input name="moduleUuid"/>
     <input name="moduleVersionMajor"/>
     <input name="moduleVersionMinor"/>
     <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="_modulehandle" value="1"/>
     <!-- Invoke the functions BioSPI_ModuleLoad and BioSPI_ModuleAttach
exposed by the BSP under test.  -->
     <invoke activity="LoadAndAttach" package="be0a1330-d59e-11d8-9669-
0800200c9a66" break_on_break="true">
        <input name="moduleUuid" var="moduleUuid"/>
```

```
        <input name="moduleVersionMajor" var="moduleVersionMajor"/>
        <input name="moduleVersionMinor" var="moduleVersionMinor"/>
        <input name="deviceIDOrNull" value="0"/>
        <input name="module" var="_modulehandle"/>
        <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="ModuleHandle" var="_modulehandle"/>
        <input name="Purpose"
var="__BioAPI_PURPOSE_ENROLL_FOR_VERIFICATION_ONLY"/>
        <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="ModuleHandle" var="_modulehandle"/>
        <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 an error code due to
an invalid BIR handle
        </description>
        <not_equal_to var1="return" var2="__BioAPI_OK"/>
```

```
     </assert_condition>
     <!-- Invoke the functions BioSPI_ModuleDetach and BioSPI_ModuleUnload -->
     <invoke activity="DetachAndUnload" package="be0a1330-d59e-11d8-9669-
0800200c9a66">
        <input name="moduleUuid" var="moduleUuid"/>
        <input name="module" var="_modulehandle"/>
     </invoke>
   </activity>
</package>
```

16.7.4 Assertion 6.3 BioSPI_FreeBIRHandle_ValidParam

16.7.4.1 Test Purpose:
To test BioSPI_FreeBIRHandle with valid parameters.

16.7.4.2 Test Scenario:
Call BioSPI_FreeBIRHandle with valid parameters.

16.7.4.3 Expected Results:
Return code of BioAPI_OK.

16.7.4.4 XML:

```
<package name="1c25d7d0-d5a2-11d8-9669-0800200c9a66">
   <author>
      author
   </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 returns BioAPI_OK.
         The relevant text in BioAPI 1.1 is quoted below from subclauses 4.2 and
2.5.1.1.
         ____________________________________________________________
         BioAPI_RETURN BioAPI BioSPI_FreeBIRHandle
            (BioAPI_HANDLE ModuleHandle,
            BioAPI_BIR_HANDLE Handle);

         Frees the memory and resources associated with the specified BIR Handle.
The associated BIR is no longer referenceable through that handle. If
necessary, the application must make the BIR persistent either in a BSPmanaged
database or an application-managed database before freeing the handle.
         ____________________________________________________________
         Section 2.5.1.1:
         The associated BIR is no longer referenceable through that handle.
         Section 4.2:
         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) 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 error
```

```
code.

        If any of the intermediate operations fail, an UNDECIDED conformity
response is issued.
      </description>
      <!-- UUID of the BSP under test -->
      <input name="_moduleUuid"/>
      <!-- Major version number of the BSP under test -->
      <input name="_moduleVersionMajor"/>
      <!-- Minor version number of the BSP under test -->
      <input name="_moduleVersionMinor"/>
      <!-- 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="moduleUuid" var="_moduleUuid"/>
         <input name="moduleVersionMajor" var="_moduleVersionMajor"/>
         <input name="moduleVersionMinor" var="_moduleVersionMinor"/>
         <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="be0a1330-d59e-11d8-9669-
0800200c9a66" function="BioSPI_ModuleEventHandler"/>
   </assertion>
   <activity name="BioSPI_FreeBIRHandle">
      <input name="moduleUuid"/>
      <input name="moduleVersionMajor"/>
      <input name="moduleVersionMinor"/>
      <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="_modulehandle" value="1"/>
      <!-- Invoke the functions BioSPI_ModuleLoad and BioSPI_ModuleAttach
exposed by the BSP under test.-->
      <invoke activity="LoadAndAttach" package="be0a1330-d59e-11d8-9669-
0800200c9a66" break_on_break="true">
         <input name="moduleUuid" var="moduleUuid"/>
         <input name="moduleVersionMajor" var="moduleVersionMajor"/>
         <input name="moduleVersionMinor" var="moduleVersionMinor"/>
         <input name="deviceIDOrNull" value="0"/>
         <input name="module" var="_modulehandle"/>
         <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="ModuleHandle" var="_modulehandle"/>
        <input name="Purpose"
var="__BioAPI_PURPOSE_ENROLL_FOR_VERIFICATION_ONLY"/>
        <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="ModuleHandle" var="_modulehandle"/>
        <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="ModuleHandle" var="_modulehandle"/>
        <input name="Handle" var="newtemplate_handle"/>
        <output name="Length" setvar="length"/>
        <output name="HeaderVersion" setvar="headerversion"/>
        <output name="ProcessedLevel" setvar="processedLevel"/>
        <output name="FormatOwner" setvar="formatowner"/>
        <output name="FormatId" setvar="formatid"/>
```

```
        <output name="Quality" setvar="quality"/>
        <output name="Purpose" setvar="purpose"/>
        <output name="BiometricData" setvar="biometricdata"/>
        <output name="Signature" setvar="signature"/>
        <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 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="be0a1330-d59e-11d8-9669-
0800200c9a66">
        <input name="moduleUuid" var="moduleUuid"/>
        <input name="module" var="_modulehandle"/>
     </invoke>
   </activity>
</package>
```

16.8 Feature 7. BioSPI Get BIR From Handle

16.8.1 BioAPI Specification References:
3.3.2.2

16.8.2 Assertion 7.1 BioSPI_GetBIRFromHandle_ValidParam

16.8.2.1 Test Purpose:
To test BioSPI_GetBIRFromHandle with valid parameters.

16.8.2.2 Test Scenario:
Call BioSPI_GetBIRFromHandle with valid parameters.

16.8.2.3 Expected Results:
1. Return code BioAPI_OK.
2. Valid BIR.

16.8.2.4 XML:

```
<package name="d4f3b820-d5c7-11d8-9669-0800200c9a66">
   <author>
      author
   </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 1.1 is quoted below from subclauses 2.5.1.2
and 4.2.
```

```
        ____________________________________________________________
        BioAPI_RETURN BioAPI BioAPI_GetBIRFromHandle
           (BioAPI_HANDLE ModuleHandle,
           BioAPI_BIR_HANDLE Handle,
           BioAPI_BIR_PTR *BIR);

        Retrieves the BIR associated with a BIR handle. The Handle is
invalidated. The BSP allocates the storage for both the retrieved BIR structure
and its data members using the application's memory allocation callback
function.
        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.
        ____________________________________________________________
        Section 2.5.1.2:
        Retrieves the BIR associated with a BIR handle.
        The Handle is invalidated.
        Section 4.2:
        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) Enroll to obtain a BIR handle
           4) Call BioSPI_GetBIRFromHandle.  The function is expected to return
BioAPI_OK.

        If any of the intermediate operations fail, an UNDECIDED conformity
response is issued.
     </description>
     <!-- UUID of the BSP under test -->
     <input name="_moduleUuid"/>
     <!-- Major version number of the BSP under test -->
     <input name="_moduleVersionMajor"/>
     <!-- Minor version number of the BSP under test -->
     <input name="_moduleVersionMinor"/>
     <!-- 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="moduleUuid" var="_moduleUuid"/>
        <input name="moduleVersionMajor" var="_moduleVersionMajor"/>
        <input name="moduleVersionMinor" var="_moduleVersionMinor"/>
        <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="be0a1330-d59e-11d8-9669-
0800200c9a66" function="BioSPI_ModuleEventHandler"/>
   </assertion>
   <activity name="BioSPI_GetBIRFromHandle">
      <input name="moduleUuid"/>
      <input name="moduleVersionMajor"/>
      <input name="moduleVersionMinor"/>
      <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="_modulehandle" value="1"/>
      <!-- Invoke the functions BioSPI_ModuleLoad and BioSPI_ModuleAttach
exposed by the BSP under test.-->
      <invoke activity="LoadAndAttach" package="be0a1330-d59e-11d8-9669-
0800200c9a66" break_on_break="true">
         <input name="moduleUuid" var="moduleUuid"/>
         <input name="moduleVersionMajor" var="moduleVersionMajor"/>
         <input name="moduleVersionMinor" var="moduleVersionMinor"/>
         <input name="deviceIDOrNull" value="0"/>
         <input name="module" var="_modulehandle"/>
         <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="ModuleHandle" var="_modulehandle"/>
         <input name="Purpose"
var="__BioAPI_PURPOSE_ENROLL_FOR_VERIFICATION_ONLY"/>
         <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="ModuleHandle" var="_modulehandle"/>
        <input name="Handle" var="newtemplate_handle"/>
        <output name="Length" setvar="length"/>
        <output name="HeaderVersion" setvar="headerversion"/>
        <output name="ProcessedLevel" setvar="processedlevel"/>
        <output name="FormatOwner" setvar="formatowner"/>
        <output name="FormatId" setvar="formatid"/>
        <output name="Quality" setvar="quality"/>
        <output name="Purpose" setvar="purpose"/>
        <output name="BiometricData" setvar="biometricdata"/>
        <output name="Signature" setvar="signature"/>
        <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_ModuleDetach and BioSPI_ModuleUnload -->
      <invoke activity="DetachAndUnload" package="be0a1330-d59e-11d8-9669-
0800200c9a66">
        <input name="moduleUuid" var="moduleUuid"/>
        <input name="module" var="_modulehandle"/>
      </invoke>
   </activity>
</package>
```

16.8.3 Assertion 7.2 BioSPI_GetBIRFromHandle_InvalidModuleHandle

16.8.3.1 Test Purpose:

To test BioSPI_GetBIRFromHandle with an invalid module handle.

16.8.3.2 Test Scenario:

Call BioSPI_GetBIRFromHandle with an invalid module handle.

16.8.3.3 Expected Results:

Return code other than BioAPI_OK.

16.8.3.4 XML:

```
<package name="6da9d8a0-d5c9-11d8-9669-0800200c9a66">
   <author>
      author
   </author>
   <description>
      This package contains the assertion
"BioSPI_GetBIRFromHandle_InvalidModuleHandle" (see the "description" element of
the assertion below).
   </description>
   <assertion name="BioSPI_GetBIRFromHandle_InvalidModuleHandle"
model="BSPTesting">
```

```
     <description>
        This assertion checks if calling BioSPI_GetBIRFromHandle with an invalid
module handle returns an error.
        The relevant text in BioAPI 1.1 is quoted below from subclause 3.3.2.2.
        ____________________________________________________________
        BioAPI_RETURN BioAPI BioAPI_GetBIRFromHandle
           (BioAPI_HANDLE ModuleHandle,
           BioAPI_BIR_HANDLE Handle,
           BioAPI_BIR_PTR *BIR);

        Retrieves the BIR associated with a BIR handle. The Handle is
invalidated.  The BSP allocates the storage for both the retrieved BIR
structure and its data members using the application's memory allocation
callback function.

        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.
        ____________________________________________________________
        Section 3.3.2.2:
        Parameters: ModuleHandle (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) Enroll to obtain a BIR handle
           4) Call BioSPI_GetBIRFromHandle with an invalid module handle.  The
function is expected to return an error.

        If any of the intermediate operations fail, an UNDECIDED conformity
response is issued.
     </description>
     <!-- UUID of the BSP under test -->
     <input name="_moduleUuid"/>
     <!-- Major version number of the BSP under test -->
     <input name="_moduleVersionMajor"/>
     <!-- Minor version number of the BSP under test -->
     <input name="_moduleVersionMinor"/>
     <!-- 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="moduleUuid" var="_moduleUuid"/>
        <input name="moduleVersionMajor" var="_moduleVersionMajor"/>
        <input name="moduleVersionMinor" var="_moduleVersionMinor"/>
        <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="be0a1330-d59e-11d8-9669-
0800200c9a66" function="BioSPI_ModuleEventHandler"/>
   </assertion>
   <activity name="BioSPI_GetBIRFromHandle">
      <input name="moduleUuid"/>
      <input name="moduleVersionMajor"/>
      <input name="moduleVersionMinor"/>
      <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="_modulehandle" value="1"/>
      <!-- Invoke the functions BioSPI_ModuleLoad and BioSPI_ModuleAttach
exposed by the BSP under test. -->
      <invoke activity="LoadAndAttach" package="be0a1330-d59e-11d8-9669-
0800200c9a66" break_on_break="true">
         <input name="moduleUuid" var="moduleUuid"/>
         <input name="moduleVersionMajor" var="moduleVersionMajor"/>
         <input name="moduleVersionMinor" var="moduleVersionMinor"/>
         <input name="deviceIDOrNull" value="0"/>
         <input name="module" var="_modulehandle"/>
         <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="ModuleHandle" var="_modulehandle"/>
         <input name="Purpose"
var="__BioAPI_PURPOSE_ENROLL_FOR_VERIFICATION_ONLY"/>
         <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="ModuleHandle" value="0"/>
        <input name="Handle" var="newtemplate_handle"/>
        <output name="Length" setvar="length"/>
        <output name="HeaderVersion" setvar="headerversion"/>
        <output name="ProcessedLevel" setvar="processedlevel"/>
        <output name="FormatOwner" setvar="formatowner"/>
        <output name="FormatId" setvar="formatid"/>
        <output name="Quality" setvar="quality"/>
        <output name="Purpose" setvar="purpose"/>
        <output name="BiometricData" setvar="biometricdata"/>
        <output name="Signature" setvar="signature"/>
        <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_H_FRAMEWORK_INVALID_MODULE_HANDLE
        </description>
        <equal_to var1="return"
var2="__BioAPIERR_H_FRAMEWORK_INVALID_MODULE_HANDLE"/>
     </assert_condition>
     <!-- Invoke the functions BioSPI_ModuleDetach and BioSPI_ModuleUnload -->
     <invoke activity="DetachAndUnload" package="be0a1330-d59e-11d8-9669-
0800200c9a66">
        <input name="moduleUuid" var="moduleUuid"/>
        <input name="module" var="_modulehandle"/>
     </invoke>
   </activity>
</package>
```

16.8.4 Assertion 7.3 BioSPI_GetBIRFromHandle_InvalidBIRHandle

16.8.4.1 Test Purpose:
To test BioSPI_GetBIRFromHandle with an invalid BIR handle.

16.8.4.2 Test Scenario:
Call BioSPI_GetBIRFromHandle with an invalid BIR handle.

16.8.4.3 Expected Results:
Return code other than BioAPI_OK.

16.8.4.4 XML:

```
<package name="24baf820-d5cb-11d8-9669-0800200c9a66">
   <author>
      author
   </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 1.1 is quoted below from subclause 3.3.2.2.
         ____________________________________________________________
         BioAPI_RETURN BioAPI BioAPI_GetBIRFromHandle
            (BioAPI_HANDLE ModuleHandle,
            BioAPI_BIR_HANDLE Handle,
            BioAPI_BIR_PTR *BIR);

         Retrieves the BIR associated with a BIR handle. The Handle is
invalidated. The BSP allocates the storage for both the retrieved BIR structure
and its data members using the application's memory allocation callback
function.

         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.
         ____________________________________________________________
         Section 3.3.2.2:
         Parameters: Handle (input) - the handle of the BIR whose header is to be
retrieved.
         ____________________________________________________________

         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 fail, an UNDECIDED conformity
response is issued.
      </description>
      <!-- UUID of the BSP under test -->
      <input name="_moduleUuid"/>
      <!-- Major version number of the BSP under test -->
      <input name="_moduleVersionMajor"/>
      <!-- Minor version number of the BSP under test -->
      <input name="_moduleVersionMinor"/>
      <!-- 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="moduleUuid" var="_moduleUuid"/>
         <input name="moduleVersionMajor" var="_moduleVersionMajor"/>
         <input name="moduleVersionMinor" var="_moduleVersionMinor"/>
         <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="be0a1330-d59e-11d8-9669-
0800200c9a66" function="BioSPI_ModuleEventHandler"/>
   </assertion>
   <activity name="BioSPI_GetBIRFromHandle">
     <input name="moduleUuid"/>
     <input name="moduleVersionMajor"/>
     <input name="moduleVersionMinor"/>
     <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="_modulehandle" value="1"/>
     <!-- Invoke the functions BioSPI_ModuleLoad and BioSPI_ModuleAttach
exposed by the BSP under test. -->
     <invoke activity="LoadAndAttach" package="be0a1330-d59e-11d8-9669-
0800200c9a66" break_on_break="true">
        <input name="moduleUuid" var="moduleUuid"/>
        <input name="moduleVersionMajor" var="moduleVersionMajor"/>
        <input name="moduleVersionMinor" var="moduleVersionMinor"/>
        <input name="deviceIDOrNull" value="0"/>
        <input name="module" var="_modulehandle"/>
        <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="ModuleHandle" var="_modulehandle"/>
        <input name="Purpose"
var="__BioAPI_PURPOSE_ENROLL_FOR_VERIFICATION_ONLY"/>
        <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="ModuleHandle" var="_modulehandle"/>
        <input name="Handle" value="-1"/>
        <output name="Length" setvar="length"/>
        <output name="HeaderVersion" setvar="headerversion"/>
        <output name="ProcessedLevel" setvar="processedLevel"/>
        <output name="FormatOwner" setvar="formatowner"/>
        <output name="FormatId" setvar="formatid"/>
        <output name="Quality" setvar="quality"/>
        <output name="Purpose" setvar="purpose"/>
        <output name="BiometricData" setvar="biometricdata"/>
        <output name="Signature" setvar="signature"/>
        <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 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="be0a1330-d59e-11d8-9669-
0800200c9a66">
        <input name="moduleUuid" var="moduleUuid"/>
        <input name="module" var="_modulehandle"/>
     </invoke>
   </activity>
</package>
```

16.9 Feature 8. BioSPI Get Header from Handle

16.9.1 BioAPI Specification References:
3.3.2.3

16.9.2 Assertion 8.1 BioSPI_GetHeaderFromHandle_ValidParam

16.9.2.1 Test Purpose:
To test BioSPI_GetHeaderFromHandle with valid parameters.

16.9.2.2 Test Scenario:
Call BioSPI_GetHeaderFromHandle with valid parameters

16.9.2.3 Expected Results:

1. Return code BioAPI_OK.
2. Valid BIR header.

16.9.2.4 XML:

```
<package name="c591a6d0-d806-11d8-9669-0800200c9a66">
   <author>
      author
   </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 1.1 is quoted below from subclauses 2.5.1.3
and 4.2.
         ____________________________________________________________
         BioAPI_RETURN BioAPI BioSPI_GetHeaderFromHandle
            (BioAPI_HANDLE ModuleHandle,
            BioAPI_BIR_HANDLE Handle,
            BioAPI_BIR_HEADER_PTR Header);

         Retrieves the BIR header identified by Handle. The BIR Handle is not
freed by the BSP.
         ____________________________________________________________
         Section 2.5.1.3:
         Retrieves the BIR header identified by Handle.
         Section 4.2:
         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.
            4) Call BioSPI_GetHeaderFromHandle.
            5) Check the return code of the call.
            6) Detach and unload the BSP.

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

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

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

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

      <!-- 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="moduleUuid" var="_moduleUuid"/>
        <input name="moduleVersionMajor" var="_moduleVersionMajor"/>
        <input name="moduleVersionMinor" var="_moduleVersionMinor"/>
        <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="be0a1330-d59e-11d8-9669-0800200c9a66"
          function="BioSPI_ModuleEventHandler"/>
   </assertion>

   <activity name="BioSPI_GetHeaderFromHandle">
     <input name="moduleUuid"/>
     <input name="moduleVersionMajor"/>
     <input name="moduleVersionMinor"/>
     <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="_modulehandle" value="1"/>

     <!-- Invoke the functions BioSPI_ModuleLoad and BioSPI_ModuleAttach
exposed by the BSP under test.
          The input value for the parameter "deviceIDOrNull" is "0", therefore
the assertion will test the BSP's default device. -->
     <invoke activity="LoadAndAttach"
          package="be0a1330-d59e-11d8-9669-0800200c9a66"
          break_on_break="true">
        <input name="moduleUuid" var="moduleUuid"/>
        <input name="moduleVersionMajor" var="moduleVersionMajor"/>
        <input name="moduleVersionMinor" var="moduleVersionMinor"/>
        <input name="deviceIDOrNull" value="0"/>
        <input name="module" var="_modulehandle"/>
        <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="ModuleHandle" var="_modulehandle"/>
        <input name="Purpose"
             var="__BioAPI_PURPOSE_ENROLL_FOR_VERIFICATION_ONLY"/>
        <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="ModuleHandle" var="_modulehandle"/>
        <input name="Handle" var="newtemplate_handle"/>
        <output name="Length" setvar="length"/>
        <output name="HeaderVersion" setvar="headerversion"/>
        <output name="ProcessedLevel" setvar="processedLevel"/>
        <output name="FormatOwner" setvar="formatowner"/>
        <output name="FormatId" setvar="formatid"/>
        <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 BioAPI_OK
         </description>
         <equal_to var1="return" var2="__BioAPI_OK"/>
      </assert_condition>

      <!-- Invoke the functions BioSPI_ModuleDetach and BioSPI_ModuleUnload -->
      <invoke activity="DetachAndUnload"
            package="be0a1330-d59e-11d8-9669-0800200c9a66" >
         <input name="moduleUuid" var="moduleUuid" />
         <input name="module" var="_modulehandle" />
      </invoke>
   </activity>

</package>
```

16.9.3 Assertion 8.2 BioSPI_GetHeaderfromHandle_InvalidModuleHandle

16.9.3.1 Test Purpose:
To test BioSPI_GetHeaderFromHandle with an invalid ModuleHandle.

16.9.3.2 Test Scenario:
Call BioSPI_GetHeaderFromHandle with an invalid module handle.

16.9.3.3 Expected Results:
Return code other than BioAPI_OK.

16.9.3.4 XML:

```
<package name="23e0bb00-d80b-11d8-9669-0800200c9a66">
   <author>
      author
   </author>

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

   <assertion name="BioSPI_GetHeaderfromHandle_InvalidModuleHandle"
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 1.1 is quoted below from subclause 3.3.2.3.
         ____________________________________________________________
         BioAPI_RETURN BioAPI BioSPI_GetHeaderFromHandle
            (BioAPI_HANDLE ModuleHandle,
            BioAPI_BIR_HANDLE Handle,
            BioAPI_BIR_HEADER_PTR Header);

         Retrieves the BIR header identified by Handle. The BIR Handle is not
freed by the BSP.
         ____________________________________________________________
         Section 3.3.2.3:
         Parameters: ModuleHandle (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) Call the function BioSPI_Enroll.
          4) Call BioSPI_GetHeaderFromHandle using an invalid module handle.
          5) Check the return code of the call.
          6) Detach and unload the BSP under test.

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

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

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

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

     <!-- 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="moduleUuid" var="_moduleUuid"/>
        <input name="moduleVersionMajor" var="_moduleVersionMajor"/>
        <input name="moduleVersionMinor" var="_moduleVersionMinor"/>
        <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="be0a1330-d59e-11d8-9669-0800200c9a66"
          function="BioSPI_ModuleEventHandler"/>
   </assertion>

   <activity name="BioSPI_GetHeaderFromHandle">
     <input name="moduleUuid"/>
     <input name="moduleVersionMajor"/>
     <input name="moduleVersionMinor"/>
     <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="_modulehandle" value="1"/>

     <!-- Invoke the functions BioSPI_ModuleLoad and BioSPI_ModuleAttach
exposed by the BSP under test.
          The input value for the parameter "deviceIDOrNull" is "0", therefore
the assertion will test the BSP's default device. -->
     <invoke activity="LoadAndAttach"
          package="be0a1330-d59e-11d8-9669-0800200c9a66"
          break_on_break="true">
        <input name="moduleUuid" var="moduleUuid"/>
        <input name="moduleVersionMajor" var="moduleVersionMajor"/>
        <input name="moduleVersionMinor" var="moduleVersionMinor"/>
        <input name="deviceIDOrNull" value="0"/>
        <input name="module" var="_modulehandle"/>
        <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="ModuleHandle" var="_modulehandle"/>
        <input name="Purpose"
             var="__BioAPI_PURPOSE_ENROLL_FOR_VERIFICATION_ONLY"/>
        <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="ModuleHandle" value="0"/>
        <input name="Handle" var="newtemplate_handle"/>
        <output name="Length" setvar="length"/>
        <output name="HeaderVersion" setvar="headerversion"/>
        <output name="ProcessedLevel" setvar="processedLevel"/>
        <output name="FormatOwner" setvar="formatowner"/>
        <output name="FormatId" setvar="formatid"/>
        <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_H_FRAMEWORK_INVALID_MODULE_HANDLE
        </description>
        <equal_to var1="return"
             var2="__BioAPIERR_H_FRAMEWORK_INVALID_MODULE_HANDLE"/>
     </assert_condition>

     <!-- Invoke the functions BioSPI_ModuleDetach and BioSPI_ModuleUnload -->
     <invoke activity="DetachAndUnload"
          package="be0a1330-d59e-11d8-9669-0800200c9a66" >
        <input name="moduleUuid" var="moduleUuid"/>
        <input name="module" var="_modulehandle"/>
     </invoke>
   </activity>

</package>
```

16.9.4 Assertion 8.3 BioSPI_GetHeaderfromHandle_InvalidBIRHandle

16.9.4.1 Test Purpose:
To test BioSPI_GetHeaderFromHandle with an invalid BIR Handle.

16.9.4.2 Test Scenario:
Call BioSPI_GetHeaderFromHandle with an invalid BIR handle.

16.9.4.3 Expected Results:
Return code other than BioAPI_OK.

16.9.4.4 XML:

```
<package name="5bdcfa20-d80e-11d8-9669-0800200c9a66">
   <author>
      author
   </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 1.1 is quoted below from subclause 3.3.2.3.
         ____________________________________________________________
         BioAPI_RETURN BioAPI BioSPI_GetHeaderFromHandle
            (BioAPI_HANDLE ModuleHandle,
            BioAPI_BIR_HANDLE Handle,
            BioAPI_BIR_HEADER_PTR Header);

         Retrieves the BIR header identified by Handle. The BIR Handle is not
freed by the BSP.
         ____________________________________________________________
         Section 3.3.2.3:
         Parameters: Handle (input) - the handle of the BIR whose header is to be
retrieved.
         ____________________________________________________________

         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 fail, an UNDECIDED conformity
response is issued.
      </description>

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

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

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

      <!-- 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="moduleUuid" var="_moduleUuid"/>
        <input name="moduleVersionMajor" var="_moduleVersionMajor"/>
        <input name="moduleVersionMinor" var="_moduleVersionMinor"/>
        <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="be0a1330-d59e-11d8-9669-0800200c9a66"
          function="BioSPI_ModuleEventHandler"/>
   </assertion>

   <activity name="BioSPI_GetHeaderFromHandle">
     <input name="moduleUuid"/>
     <input name="moduleVersionMajor"/>
     <input name="moduleVersionMinor"/>
     <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="_modulehandle" value="1"/>

     <!-- Invoke the functions BioSPI_ModuleLoad and BioSPI_ModuleAttach
exposed by the BSP under test.
          The input value for the parameter "deviceIDOrNull" is "0", therefore
the assertion will test the BSP's default device. -->
     <invoke activity="LoadAndAttach"
          package="be0a1330-d59e-11d8-9669-0800200c9a66"
          break_on_break="true">
        <input name="moduleUuid" var="moduleUuid"/>
        <input name="moduleVersionMajor" var="moduleVersionMajor"/>
        <input name="moduleVersionMinor" var="moduleVersionMinor"/>
        <input name="deviceIDOrNull" value="0"/>
        <input name="module" var="_modulehandle"/>
        <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="ModuleHandle" var="_modulehandle"/>
        <input name="Purpose"
             var="__BioAPI_PURPOSE_ENROLL_FOR_VERIFICATION_ONLY"/>
        <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="ModuleHandle" var="_modulehandle"/>
        <input name="Handle" value="-1"/>
        <output name="Length" setvar="length"/>
        <output name="HeaderVersion" setvar="headerversion"/>
        <output name="ProcessedLevel" setvar="processedLevel"/>
        <output name="FormatOwner" setvar="formatowner"/>
        <output name="FormatId" setvar="formatid"/>
        <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 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="be0a1330-d59e-11d8-9669-0800200c9a66" >
        <input name="moduleUuid" var="moduleUuid" />
        <input name="module" var="_modulehandle" />
```

```
      </invoke>
   </activity>

</package>
```

16.9.5 Assertion 8.4 BioSPI_GetHeaderFromHandle_ BIRHandleNotFreed

16.9.5.1 Test Purpose:
To test the BIR handle is not freed after call BioSPI_GetHeaderFromHandle.

16.9.5.2 Test 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.

16.9.5.3 Expected Results:
The call to BioSPI_GetBIRFromHandle is expected to return BioAPI_OK.

16.9.5.4 XML:

```
<package name="0b43f8f0-08e4-11d9-9669-0800200c9a66" >
   <author>
      author
   </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 the call of the function
BioSPI_GetHeaderFromHandle, the BIR handle has not been freed.
         The relevant text in BioAPI 1.1 is quoted below from subclause 2.5.1.3
         ____________________________________________________________
         BioAPI_RETURN BioAPI BioSPI_GetHeaderFromHandle
            (BioAPI_HANDLE ModuleHandle,
            BioAPI_BIR_HANDLE Handle,
            BioAPI_BIR_HEADER_PTR Header);

         Retrieves the BIR header identified by Handle. The BIR Handle is not
freed by the BSP.
         ____________________________________________________________
         Section 2.5.1.3:
         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 fail, an UNDECIDED conformity
response is issued.
     </description>

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

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

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

     <!-- 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="moduleUuid" var="_moduleUuid"/>
        <input name="moduleVersionMajor" var="_moduleVersionMajor"/>
        <input name="moduleVersionMinor" var="_moduleVersionMinor"/>
        <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="be0a1330-d59e-11d8-9669-0800200c9a66"
          function="BioSPI_ModuleEventHandler"/>
   </assertion>

   <activity name="BioSPI_GetHeaderFromHandle">
     <input name="moduleUuid"/>
     <input name="moduleVersionMajor"/>
     <input name="moduleVersionMinor"/>
     <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="_modulehandle" value="1"/>

     <!-- Invoke the functions BioSPI_ModuleLoad and BioSPI_ModuleAttach
exposed by the BSP under test.
          The input value for the parameter "deviceIDOrNull" is "0", therefore
the assertion will test the BSP's default device. -->
     <invoke activity="LoadAndAttach"
          package="be0a1330-d59e-11d8-9669-0800200c9a66"
          break_on_break="true">
        <input name="moduleUuid" var="moduleUuid"/>
        <input name="moduleVersionMajor" var="moduleVersionMajor"/>
        <input name="moduleVersionMinor" var="moduleVersionMinor"/>
        <input name="deviceIDOrNull" value="0"/>
        <input name="module" var="_modulehandle"/>
        <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="ModuleHandle" var="_modulehandle"/>
        <input name="Purpose"
             var="__BioAPI_PURPOSE_ENROLL_FOR_VERIFICATION_ONLY"/>
        <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="ModuleHandle" var="_modulehandle"/>
        <input name="Handle" var="newtemplate_handle"/>
        <output name="Length" setvar="length"/>
        <output name="HeaderVersion" setvar="headerversion"/>
        <output name="ProcessedLevel" setvar="processedLevel"/>
        <output name="FormatOwner" setvar="formatowner"/>
        <output name="FormatId" setvar="formatid"/>
        <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, 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="ModuleHandle" var="_modulehandle"/>
        <input name="Handle" var="newtemplate_handle"/>
        <output name="Length" setvar="length"/>
        <output name="HeaderVersion" setvar="headerversion"/>
        <output name="ProcessedLevel" setvar="processedLevel"/>
        <output name="FormatOwner" setvar="formatowner"/>
        <output name="FormatId" setvar="formatid"/>
        <output name="Quality" setvar="quality"/>
        <output name="Purpose" setvar="purpose"/>
        <output name="BiometricData" setvar="biometricdata"/>
        <output name="Signature" setvar="signature"/>
        <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_ModuleDetach and BioSPI_ModuleUnload -->
     <invoke activity="DetachAndUnload"
          package="be0a1330-d59e-11d8-9669-0800200c9a66" >
        <input name="moduleUuid" var="moduleUuid"/>
        <input name="module" var="_modulehandle"/>
     </invoke>
   </activity>

</package>
```

16.10 Feature 9. BioSPI Enable Events

16.10.1 BioAPI Specification References:
3.3.3.1

16.10.2 Assertion 9.1 BioSPI_EnableEvents_ValidParam

16.10.2.1 Test Purpose:
To test BioSPI_EnableEvents with valid parameters.

16.10.2.2 Test Scenario:
Call BioSPI_EnableEvents with valid parameters.

16.10.2.3 Expected Results:
Return code BioAPI_OK.

16.10.2.4 XML:

```
<package name="49a53f50-065a-11d9-9669-0800200c9a66">
   <author>
      author
   </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 1.1 is quoted below from subclause 2.5.2.1.
         ____________________________________________________________
         BioAPI_RETURN BioAPI BioSPI_EnableEvents
            (BioAPI_HANDLE ModuleHandle,
            BioAPI_MODULE_EVENT_MASK *Events);

         This function enables the events (indicated by the Events mask) from the
attached BSP in the current process. All other events from this BSP are
disabled for this process.

         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.
         ____________________________________________________________
         Section 2.5.2.1:
         This function enables the events (indicated by the Events mask) from the
attached BSP in the current process.
         All other events from this BSP are disabled for this process.
         ____________________________________________________________

         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_ModuleEventHandler, 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 fail, an UNDECIDED conformity
response is issued.
     </description>
     <!-- UUID of the BSP under test -->
     <input name="_moduleUuid"/>
     <!-- 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"/>
     <!-- Major version number of the BSP -->
     <input name="_moduleVersionMajor"/>
     <!-- Minor version number of the BSP -->
     <input name="_moduleVersionMinor"/>
     <!-- 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="moduleUuid" var="_moduleUuid"/>
        <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="moduleVersionMajor" var="_moduleVersionMajor"/>
        <input name="moduleVersionMinor" var="_moduleVersionMinor"/>
        <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_ModuleEventHandler"/>
  </assertion>
  <activity name="BioSPI_EnableEvents">
     <input name="moduleUuid"/>
     <input name="eventtimeouttime"/>
     <input name="eventnotifyinsert"/>
     <input name="eventnotifyremove"/>
     <input name="eventnotifyfault"/>
     <input name="eventnotifysourcepresent"/>
     <input name="eventnotifysourceremoved"/>
     <input name="moduleVersionMajor"/>
     <input name="moduleVersionMinor"/>
     <input name="timeout"/>
     <set name="_modulehandle" 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_ModuleLoad. -->
     <invoke function="BioSPI_ModuleLoad">
        <input name="Reserved" value="0"/>
        <input name="BSPUuid" var="moduleUuid"/>
        <input name="BioAPINotifyCallback" value="*"/>
        <input name="BioAPINotifyCallbackCtx" 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_ModuleLoad 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_ModuleAttach. -->
     <invoke function="BioSPI_ModuleAttach">
        <input name="BSPUuid" var="moduleUuid"/>
        <input name="VersionMajor" var="moduleVersionMajor"/>
        <input name="VersionMinor" var="moduleVersionMinor"/>
        <input name="DeviceID" var="_deviceID"/>
        <input name="Reserved1" value="0"/>
        <input name="Reserved2" value="0"/>
        <input name="ModuleHandle" var="_modulehandle"/>
        <input name="Reserved3" value="0"/>
        <input name="Reserved4" value="0"/>
        <input name="Reserved5" value="0"/>
        <input name="Reserved6" value="0"/>
        <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_ModuleAttach 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="ModuleHandle" var="_modulehandle"/>
        <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 response_if_false="undecided" 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_ModuleDetach and BioSPI_ModuleUnload -->
     <invoke activity="DetachAndUnload" package="be0a1330-d59e-11d8-9669-
0800200c9a66">
        <input name="moduleUuid" var="moduleUuid"/>
        <input name="module" var="_modulehandle"/>
     </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 "_deviceID", "_insert", "_sourcePresent" and "_eventtype"
are set depending on the input parameter values. -->
   <activity name="EventHandler" atomic="true">
     <input name="BSPUuid"/>
     <input name="BioAPINotifyCallbackCtx"/>
     <input name="DeviceID"/>
     <input name="Reserved"/>
     <input name="EventType"/>
     <output name="return"/>
     <!-- Check if the received event notification is compatible with the event
mask set by BioSPI_EnableEvents -->
     <set name="_eventFlag" value="true">
        <only_if>
          <not>
             <not var="_enableEventsCalled"/>
          </not>
          <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>
     <!-- Set the global variable "_deviceID" if:
          - it is not set; and
          - the event notification is either BioAPI_NOTIFY_INSERT or
BioAPI_NOTIFY_SOURCE_PRESENT. -->
     <set name="_deviceID" var="DeviceID">
       <only_if>
          <not>
             <existing var="_deviceID"/>
          </not>
          <or>
             <equal_to var1="EventType" var2="__BioAPI_NOTIFY_INSERT"/>
             <equal_to var1="EventType" var2="__BioAPI_NOTIFY_SOURCE_PRESENT"/>
          </or>
       </only_if>
     </set>
     <!-- Set the global variable "_insert" to "true" if the event notification
is BioAPI_NOTIFY_INSERT. -->
     <set name="_insert" value="true">
       <only_if>
          <not var="_enableEventsCalled"/>
          <equal_to var1="EventType" var2="__BioAPI_NOTIFY_INSERT"/>
       </only_if>
     </set>
     <set name="return" var="__BioAPI_OK"/>
   </activity>
</package>
```

16.10.3 Assertion 9.2 BioSPI_EnableEvents_InvalidModuleHandle

16.10.3.1 Test Purpose:
To test BioSPI_EnableEvents with an invalid module handle.

16.10.3.2 Test Scenario:
Call BioSPI_EnableEvents with an invalid module handle.

16.10.3.3 Expected Results:
Return code other than BioAPI_OK.

16.10.3.4 XML:

```
<package name="ccf8f930-0cab-11d9-9669-0800200c9a66">
   <author>
      author
   </author>

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

   <assertion name="BioSPI_EnableEvents_InvalidModuleHandle" model="BSPTesting">
      <description>
         This assertion is to test BioSPI_EnableEvents with an invalid module
handle.
         The relevant text in BioAPI 1.1 is quoted below from subclause 3.3.3.1.
         ____________________________________________________________
         BioAPI_RETURN BioAPI BioSPI_EnableEvents
            (BioAPI_HANDLE ModuleHandle,
            BioAPI_MODULE_EVENT_MASK *Events);

         This function enables the events (indicated by the Events mask) from the
attached BSP in the current process. All other events from this BSP are
disabled for this process.

         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.
         ____________________________________________________________
         Section 3.3.3.1:
         Parameters: ModuleHandle (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) Call BioSPI_EnableEvents with an invalid module handle.  The
function is expected to return an error.
            4) Detach and unload the BSP under test.

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

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

      <!-- 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"/>

     <!-- Major version number of the BSP -->
     <input name="_moduleVersionMajor" />

     <!-- Minor version number of the BSP -->
     <input name="_moduleVersionMinor" />

     <!-- 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="moduleUuid" var="_moduleUuid"/>
        <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="moduleVersionMajor"
             var="_moduleVersionMajor" />
        <input name="moduleVersionMinor"
             var="_moduleVersionMinor" />
        <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="be0a1330-d59e-11d8-9669-0800200c9a66"
          function="BioSPI_ModuleEventHandler"/>
   </assertion>

   <activity name="BioSPI_EnableEvents">
     <input name="moduleUuid"/>
     <input name="eventnotifyinsert" />
     <input name="eventnotifyremove" />
     <input name="eventnotifyfault" />
     <input name="eventnotifysourcepresent" />
     <input name="eventnotifysourceremoved" />
     <input name="moduleVersionMajor" />
     <input name="moduleVersionMinor" />
     <input name="inserttimeouttime" />

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

     <!-- Invoke the functions BioSPI_ModuleLoad and BioSPI_ModuleAttach
exposed by the BSP under test.
          The input value for the parameter "deviceIDOrNull" is "0", therefore
the assertion will test the BSP's default device. -->
     <invoke activity="LoadAndAttach"
          package="be0a1330-d59e-11d8-9669-0800200c9a66"
```

```
          break_on_break="true">
        <input name="moduleUuid" var="moduleUuid"/>
        <input name="moduleVersionMajor" var="moduleVersionMajor"/>
        <input name="moduleVersionMinor" var="moduleVersionMinor"/>
        <input name="deviceIDOrNull" value="0"/>
        <input name="module" var="_modulehandle"/>
        <input name="eventtimeouttime" var="inserttimeouttime"/>
     </invoke>

     <!-- Invoke the function BioSPI_EnableEvents with an invalid module
handle.-->
     <invoke function="BioSPI_EnableEvents" >
        <input name="ModuleHandle" 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_H_FRAMEWORK_INVALID_MODULE_HANDLE
        </description>
        <equal_to var1="return"
             var2="__BioAPIERR_H_FRAMEWORK_INVALID_MODULE_HANDLE"/>
     </assert_condition>

     <!-- Invoke the functions BioSPI_ModuleDetach and BioSPI_ModuleUnload -->
     <invoke activity="DetachAndUnload"
          package="be0a1330-d59e-11d8-9669-0800200c9a66" >
        <input name="moduleUuid" var="moduleUuid" />
        <input name="module" var="_modulehandle" />
     </invoke>
   </activity>

</package>
```

16.11 Feature 13. BioSPI Capture

16.11.1 BioAPI Specification References:
3.3.4.1

16.11.2 Assertion 13.1 BioSPI_Capture_InvalidModuleHandle

16.11.2.1 Test Purpose:
To test BioSPI_Capture with an invalid module handle.

16.11.2.2 Test Scenario:
Call BioSPI_Capture with an invalid module handle.

16.11.2.3 Expected Results:
Return code other than BioAPI_OK.

16.11.2.4 XML:

```
<package name="5cdc27d0-2835-11d9-9669-0800200c9a66">
   <author>
      author
   </author>

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

   <assertion name="BioSPI_Capture_InvalidModuleHandle" model="BSPTesting">
      <description>
         This assertion checks if calling BioSPI_Capture with an invalid module
handle returns an error BioAPIERR_H_FRAMEWORK_INVALID_MODULE_HANDLE.
         The relevant text in BioAPI 1.1 is quoted below from subclause 2.5.3.1.
         ____________________________________________________
         BioAPI_RETURN BioAPI BioSPI_Capture
            (BioAPI_HANDLE ModuleHandle,
            BioAPI_BIR_PURPOSE Purpose,
            BioAPI_BIR_HANDLE_PTR CapturedBIR,
            sint32 Timeout,
            BioAPI_BIR_HANDLE_PTR AuditData)

         ModuleHandle (input) - The handle of the attached BioAPI service
provider.
         ____________________________________________________
         Section 2.5.3.1:
         Parameters: ModuleHandle (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) Call BioSPI_Capture with an invalid module handle.
            4) Check the return code.  It is expected to be
BioAPIERR_H_FRAMEWORK_INVALID_MODULE_HANDLE.
            5) Detach and unload the BSP under test.

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

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

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

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

      <!-- 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_InvalidModuleHdl">
        <input name="moduleUuid" var="_moduleUuid"/>
        <input name="moduleVersionMajor" var="_moduleVersionMajor"/>
        <input name="moduleVersionMinor" var="_moduleVersionMinor"/>
        <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="be0a1330-d59e-11d8-9669-0800200c9a66"
          function="BioSPI_ModuleEventHandler"/>
   </assertion>

   <activity name="BioSPI_Capture_InvalidModuleHdl">
     <input name="moduleUuid"/>
     <input name="moduleVersionMajor"/>
     <input name="moduleVersionMinor"/>
     <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="_modulehandle" value="1"/>

     <!-- Invoke the functions BioSPI_ModuleLoad and BioSPI_ModuleAttach
exposed by the BSP under test.
          The input value for the parameter "deviceIDOrNull" is "0", therefore
the assertion will test the BSP's default device. -->
     <invoke activity="LoadAndAttach"
          package="be0a1330-d59e-11d8-9669-0800200c9a66"
          break_on_break="true">
        <input name="moduleUuid" var="moduleUuid"/>
        <input name="moduleVersionMajor" var="moduleVersionMajor"/>
        <input name="moduleVersionMinor" var="moduleVersionMinor"/>
        <input name="deviceIDOrNull" value="0"/>
        <input name="module" var="_modulehandle"/>
        <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="ModuleHandle" value="0"/>
        <input name="Purpose"
             var="__BioAPI_PURPOSE_ENROLL_FOR_VERIFICATION_ONLY"/>
        <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_H_FRAMEWORK_INVALID_MODULE_HANDLE
        </description>
        <equal_to var1="return"
             var2="__BioAPIERR_H_FRAMEWORK_INVALID_MODULE_HANDLE"/>
     </assert_condition>

     <!-- Invoke the functions BioSPI_ModuleDetach and BioSPI_ModuleUnload -->
     <invoke activity="DetachAndUnload"
          package="be0a1330-d59e-11d8-9669-0800200c9a66" >
        <input name="moduleUuid" var="moduleUuid" />
        <input name="module" var="_modulehandle" />
     </invoke>
   </activity>

</package>
```

16.11.3 Assertion 13.2 BioSPI_Capture_IntermediateProcessedBIR

16.11.3.1 Test Purpose:

To test that either an "intermediate" BIR or a "processed" BIR is returned for a specified process.

16.11.3.2 Test Scenario:
A valid biometric is presented to BioSPI_Capture.

16.11.3.3 Expected Results:
Return code BioAPI_OK and a valid "intermediate" or "processed" BIR.

16.11.3.4 XML:

```
<package name="62dfd2f0-d5cc-11d8-9669-0800200c9a66">
   <author>
      author
   </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 1.1 is quoted below from subclause 2.5.3.1.
         ____________________________________________________________
         BioAPI_RETURN BioAPI BioSPI_Capture
            (BioAPI_HANDLE ModuleHandle,
            BioAPI_BIR_PURPOSE Purpose,
            BioAPI_BIR_HANDLE_PTR CapturedBIR,
            sint32 Timeout,
            BioAPI_BIR_HANDLE_PTR AuditData)

         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).
         ____________________________________________________________
         Section 2.5.3.1:
         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).
         ____________________________________________________________

         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 fail, an UNDECIDED conformity
response is issued.
      </description>

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

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

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

     <!-- 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="moduleUuid" var="_moduleUuid"/>
        <input name="moduleVersionMajor" var="_moduleVersionMajor"/>
        <input name="moduleVersionMinor" var="_moduleVersionMinor"/>
        <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="be0a1330-d59e-11d8-9669-0800200c9a66"
          function="BioSPI_ModuleEventHandler"/>
   </assertion>

   <activity name="CapturedBIR_Datatype">
     <input name="moduleUuid"/>
     <input name="moduleVersionMajor"/>
     <input name="moduleVersionMinor"/>
     <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="_modulehandle" value="1"/>

     <!-- Invoke the functions BioSPI_ModuleLoad and BioSPI_ModuleAttach
exposed by the BSP under test.
          The input value for the parameter "deviceIDOrNull" is "0", therefore
the assertion will test the BSP's default device. -->
     <invoke activity="LoadAndAttach"
          package="be0a1330-d59e-11d8-9669-0800200c9a66"
          break_on_break="true">
        <input name="moduleUuid" var="moduleUuid"/>
        <input name="moduleVersionMajor" var="moduleVersionMajor"/>
        <input name="moduleVersionMinor" var="moduleVersionMinor"/>
        <input name="deviceIDOrNull" value="0"/>
```

```
        <input name="module" var="_modulehandle"/>
        <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="ModuleHandle" var="_modulehandle"/>
        <input name="Purpose"
             var="__BioAPI_PURPOSE_ENROLL_FOR_VERIFICATION_ONLY"/>
        <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="ModuleHandle" var="_modulehandle"/>
        <input name="Handle" var="capturedbir_handle"/>
        <output name="Length" setvar="length"/>
        <output name="HeaderVersion" setvar="headerversion"/>
        <output name="ProcessedLevel" setvar="processedLevel"/>
        <output name="FormatOwner" setvar="formatowner"/>
        <output name="FormatId" setvar="formatid"/>
        <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, 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_ModuleDetach and BioSPI_ModuleUnload -->
     <invoke activity="DetachAndUnload"
          package="be0a1330-d59e-11d8-9669-0800200c9a66" >
        <input name="moduleUuid" var="moduleUuid" />
        <input name="module" var="_modulehandle" />
     </invoke>
   </activity>

</package>
```

16.11.4 Assertion 13.3 BioSPI_Capture_PurposeInHeader

16.11.4.1 Test Purpose:
To test that Purpose is recorded in the header of the CapturedBIR.

16.11.4.2 Test Scenario:
A valid biometric is presented to BioSPI_Capture.

16.11.4.3 Expected Results:
Return code BioAPI_OK and Purpose in the header of the CapturedBIR.

16.11.4.4 XML:

16.11.5 Assertion 13.4 BioSPI_Capture_Serialization

16.11.5.1 Test Purpose:
To test serialization.

16.11.5.2 Test Scenario:
Two or more capture processes are started.

16.11.5.3 Expected Results:
The capture results are expected to be returned to the proper capture process.

16.11.5.4 XML:

16.11.6 Assertion 13.5 BioSPI_Capture_SerializationTimeout

16.11.6.1 Test Purpose:
To test serialization of capture with timeout.

16.11.6.2 Test Scenario:
1. Start test.
2. Prompt "Start 2nd BioAPI_Capture Application (Long Timeout)" and pause test.
3. Start 2nd application, observe GUI.
4. Continue test (Timeout shorter than 2nd application)
5. Wait for timeout.

16.11.6.3 Expected Results:
Return code BioAPIERR_BSP_TIMEOUT_EXPIRED.

16.11.6.4 XML:

16.11.7 Assertion 13.6 BioSPI_Capture_Timeout

16.11.7.1 Test Purpose:
To test that BioAPIERR_BSP_TIMOUT_EXPIRED is returned.

16.11.7.2 Test Scenario:
Call BioSPI_Capture_Timeout and allow it to timeout.

16.11.7.3 Expected Results:
Return code BioAPIERR_BSP_TIMOUT_EXPIRED.

16.11.7.4 XML:

16.12 Feature 13a. Return of raw/audit data

16.12.1 BioAPI Specification References:
3.3.4.1

16.12.2 Assertion 13a.1 BioSPI_Capture_AuditData

16.12.2.1 Test Purpose:
To test the return of "raw" data if the BSP supports return of raw/audit data.

16.12.2.2 Test Scenario:
A valid biometric is presented to BioSPI_Capture.

16.12.2.3 Expected Results:
If the BSP supports AuditData, a handle to a "raw" BIR or a value of BioAPI_INVALID_BIR_HANDLE if no audit data is available. If AuditData is not supported, the BSP may return a handle value of BioAPI_UNSUPPORTED_BIR_HANDLE.

16.12.2.4 XML:

```
<package name="44450e60-2445-11d9-9669-0800200c9a66">
   <author>
      author
   </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 1.1 is quoted below from subclauses 2.5.3.1,
2.2.1.2 and 4.2.4.2.
         ____________________________________________________________
         BioAPI_RETURN BioAPI BioSPI_Capture
            (BioAPI_HANDLE ModuleHandle,
            BioAPI_BIR_PURPOSE Purpose,
            BioAPI_BIR_HANDLE_PTR CapturedBIR,
            sint32 Timeout,
            BioAPI_BIR_HANDLE_PTR AuditData)

         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.
         ____________________________________________________________
         Section 2.5.3.1:
         If AuditData is non-NULL, a BIR of type 'raw' may be returned.
         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.
         Section 2.2.1.2: BioAPI_OPTIONS_MASK
         #define BioAPI_RAW (0x00000001) If set, indicates that the BSP supports
the return of raw/audit data.
         Section 4.2.4.2:
         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="_moduleUuid"/>

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

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

     <!-- 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="moduleUuid" var="_moduleUuid"/>
        <input name="moduleVersionMajor" var="_moduleVersionMajor"/>
        <input name="moduleVersionMinor" var="_moduleVersionMinor"/>
        <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="be0a1330-d59e-11d8-9669-0800200c9a66"
          function="BioSPI_ModuleEventHandler"/>
   </assertion>

   <activity name="CapturedBIR_AuditDataPresent">
```

```
     <input name="moduleUuid"/>
     <input name="moduleVersionMajor"/>
     <input name="moduleVersionMinor"/>
     <input name="inserttimeouttime"/>
     <input name="nosourcepresentsupported" />
     <input name="sourcepresenttimeouttime"/>
     <input name="capturetimeouttime"/>
     <input name="supportAuditData" />

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

     <!-- Invoke the functions BioSPI_ModuleLoad and BioSPI_ModuleAttach
exposed by the BSP under test.
          The input value for the parameter "deviceIDOrNull" is "0", therefore
the assertion will test the BSP's default device. -->
     <invoke activity="LoadAndAttach"
          package="be0a1330-d59e-11d8-9669-0800200c9a66"
          break_on_break="true">
        <input name="moduleUuid" var="moduleUuid"/>
        <input name="moduleVersionMajor" var="moduleVersionMajor"/>
        <input name="moduleVersionMinor" var="moduleVersionMinor"/>
        <input name="deviceIDOrNull" value="0"/>
        <input name="module" var="_modulehandle"/>
        <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="ModuleHandle" var="_modulehandle"/>
        <input name="Purpose"
             var="__BioAPI_PURPOSE_ENROLL_FOR_VERIFICATION_ONLY"/>
        <input name="Timeout" var="capturetimeouttime"/>
        <output name="AuditData" setvar="auditbir_handle"/>
        <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>

     <!-- 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 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 var1="auditbir_handle" value2="0" />
        </only_if>

        <input name="modulehandle" var="_modulehandle"/>
        <input name="auditbirhandle" var="auditbir_handle" />
     </invoke>

     <!-- Invoke the functions BioSPI_ModuleDetach and BioSPI_ModuleUnload -->
     <invoke activity="DetachAndUnload"
          package="be0a1330-d59e-11d8-9669-0800200c9a66" >
        <input name="moduleUuid" var="moduleUuid" />
        <input name="module" var="_modulehandle" />
     </invoke>
   </activity>

   <activity name="check_auditdata_type">
     <input name="modulehandle" />
     <input name="auditbirhandle" />
```

```
     <!-- Invoke the function BioSPI_GetHeaderFromHandle.-->
     <invoke function="BioSPI_GetHeaderFromHandle">
        <input name="ModuleHandle" var="_modulehandle"/>
        <input name="Handle" var="auditbirhandle"/>
        <output name="Length" setvar="length"/>
        <output name="HeaderVersion" setvar="headerversion"/>
        <output name="ProcessedLevel" setvar="processedLevel"/>
        <output name="FormatOwner" setvar="formatowner"/>
        <output name="FormatId" setvar="formatid"/>
        <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, 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>
```

16.13 Feature 13b. Return of quality in the captured BIR header

16.13.1 BioAPI Specification References:
3.3.4.1, (2.1.46, 4.2.4.2)

16.13.2 Assertion 13b.1 BioSPI_Capture_ReturnQuality

16.13.2.1 Test Purpose:
To test that the captured BIR contains a valid quality value (in the range 0-100).

16.13.2.2 Test Scenario:
Call BioSPI_Capture_ReturnQuality with valid parameters.

16.13.2.3 Expected Results:
The captured BIR contains a valid quality value (in the range 0-100).

16.13.2.4 XML:

```
<package name="ace3ca10-2765-11d9-9669-0800200c9a66">
   <author>
      author
   </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
captured BIR contains a valid quality value (in the range 0-100).
         The relevant text in BioAPI 1.1 is quoted below from subclauses 2.2.1.2
and 4.2.4.2.
         ____________________________________________________________
         BioAPI_RETURN BioAPI BioSPI_Capture
            (BioAPI_HANDLE ModuleHandle,
            BioAPI_BIR_PURPOSE Purpose,
            BioAPI_BIR_HANDLE_PTR CapturedBIR,
            sint32 Timeout,
            BioAPI_BIR_HANDLE_PTR AuditData)

         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.
         ____________________________________________________________
         Section 2.2.1.2:
         #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.
         Section 4.2.4.2:
         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) Capture
            4) GetHeaderFromHandle
            5) Check the quality value of the returned BIR.

         If any of the intermediate operations fail, an UNDECIDED conformity
```

```
response is issued.
     </description>

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

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

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

     <!-- 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="moduleUuid" var="_moduleUuid"/>
        <input name="moduleVersionMajor" var="_moduleVersionMajor"/>
        <input name="moduleVersionMinor" var="_moduleVersionMinor"/>
        <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="be0a1330-d59e-11d8-9669-0800200c9a66"
           function="BioSPI_ModuleEventHandler"/>
   </assertion>

   <activity name="BioSPI_Capture_ReturnQuality">
     <input name="moduleUuid"/>
     <input name="moduleVersionMajor"/>
     <input name="moduleVersionMinor"/>
```

```
     <input name="inserttimeouttime"/>
     <input name="nosourcepresentsupported" />
     <input name="sourcepresenttimeouttime"/>
     <input name="capturetimeouttime"/>
     <input name="intermediateQualitySupported" />
     <input name="processedQualitySupported"  />

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

     <!-- Invoke the functions BioSPI_ModuleLoad and BioSPI_ModuleAttach
exposed by the BSP under test.
          The input value for the parameter "deviceIDOrNull" is "0", therefore
the assertion will test the BSP's default device. -->
     <invoke activity="LoadAndAttach"
          package="be0a1330-d59e-11d8-9669-0800200c9a66"
          break_on_break="true">
        <input name="moduleUuid" var="moduleUuid"/>
        <input name="moduleVersionMajor" var="moduleVersionMajor"/>
        <input name="moduleVersionMinor" var="moduleVersionMinor"/>
        <input name="deviceIDOrNull" value="0"/>
        <input name="module" var="_modulehandle"/>
        <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="ModuleHandle" var="_modulehandle"/>
        <input name="Purpose"
             var="__BioAPI_PURPOSE_ENROLL_FOR_VERIFICATION_ONLY"/>
        <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="ModuleHandle" var="_modulehandle"/>
        <input name="Handle" var="capturedbir_handle"/>
        <output name="Length" setvar="length"/>
        <output name="HeaderVersion" setvar="headerversion"/>
        <output name="ProcessedLevel" setvar="processedLevel"/>
        <output name="FormatOwner" setvar="formatowner"/>
        <output name="FormatId" setvar="formatid"/>
        <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, 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="be0a1330-d59e-11d8-9669-0800200c9a66"
          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="be0a1330-d59e-11d8-9669-0800200c9a66"
          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_ModuleDetach and BioSPI_ModuleUnload -->
      <invoke activity="DetachAndUnload"
          package="be0a1330-d59e-11d8-9669-0800200c9a66" >
        <input name="moduleUuid" var="moduleUuid" />
        <input name="module" var="_modulehandle" />
      </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="be0a1330-d59e-11d8-9669-0800200c9a66" >
        <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="be0a1330-d59e-11d8-9669-0800200c9a66" >
        <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="be0a1330-d59e-11d8-9669-0800200c9a66" >
         <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="be0a1330-d59e-11d8-9669-0800200c9a66" >
         <only_if>
            <equal_to var1="processedLevel"
                  var2="__BioAPI_BIR_DATA_TYPE_PROCESSED" />
         </only_if>
         <input name="quality" var="quality" />
      </invoke>
   </activity>

</package>
```

16.14 Feature 13c. BIR signing (by BSP)

16.14.1 BioAPI Specification References:
1.5, 2.1.7

16.14.2 Assertion 13c.1 BioSPI_Capture_BIRSigned

16.14.2.1 Test Purpose:
To test BIR signing if supported by the BSP.

16.14.2.2 Test Scenario:
If the BSP supports signing cal BioSPI_Capture with valid parameters. Present a valid biometric.

16.14.2.3 Expected Results:
Return code BioAPI_OK and a valid CapturedBIR that includes a signature.

16.14.2.4 XML:

16.15 Feature 13d. BIR encryption (by BSP)

16.15.1 BioAPI Specification References:
1.5, 2.1.7

16.15.2 Assertion 13d.1 BioSPI_Capture_BIREncrypted

16.15.2.1 Test Purpose:
To test BIR encryption if supported by the BSP.

16.15.2.2 Test Scenario:
If the BSP supports BIR encryption call BioSPI_Enroll with valid parameters. Present a valid biometric.

16.15.2.3 Expected Results:
Return code BioAPI_OK and a valid CapturedBIR.

16.15.2.4 XML:

16.16 Feature 14. BioSPI Create Template

16.16.1 BioAPI Specification References:
3.3.4.2

16.16.2 Assertion 14.1 BioSPI_CreateTemplate_OutputBIRPurpose

16.16.2.1 Test Purpose:
To test BioSPI_CreateTemplate with valid parameters.

16.16.2.2 Test Scenario:
Call BioSPI_CreateTemplate with valid parameters.

16.16.2.3 Expected Results:
Return code BioAPI_OK.

16.16.2.4 XML:

```
<package name="01094930-29dc-11d9-9669-0800200c9a66">
   <author>
      author
   </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 1.1 is quoted below from subclause 1.6.
         ____________________________________________________________
         BioAPI_RETURN BioAPI BioSPI_CreateTemplate
            (BioAPI_HANDLE ModuleHandle,
            const BioAPI_INPUT_BIR *CapturedBIR,
            const BioAPI_INPUT_BIR *StoredTemplate,
            BioAPI_BIR_HANDLE_PTR NewTemplate,
            const BioAPI_DATA *Payload)

         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.
         ____________________________________________________________
         Section 1.6:
         CreateTemplate always takes an 'intermediate' BIR as input, and
constructs a template with the recorded purpose of either 'enroll_verify'
and/or 'enroll_identify'.
```

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 fail, an UNDECIDED conformity response is issued.

```
     </description>

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

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

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

     <!-- 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="moduleUuid" var="_moduleUuid"/>
        <input name="moduleVersionMajor" var="_moduleVersionMajor"/>
        <input name="moduleVersionMinor" var="_moduleVersionMinor"/>
        <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="be0a1330-d59e-11d8-9669-0800200c9a66"
          function="BioSPI_ModuleEventHandler"/>
   </assertion>

   <activity name="BioSPI_CreateTemplate">
      <input name="moduleUuid"/>
      <input name="moduleVersionMajor"/>
      <input name="moduleVersionMinor"/>
      <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="_modulehandle" value="1"/>

      <!-- Invoke the functions BioSPI_ModuleLoad and BioSPI_ModuleAttach
exposed by the BSP under test.
          The input value of the parameter "deviceIDOrNull" is "0", therefore
the assertion will test the BSP's default device. -->
      <invoke activity="LoadAndAttach"
          package="be0a1330-d59e-11d8-9669-0800200c9a66"
          break_on_break="true">
         <input name="moduleUuid" var="moduleUuid"/>
         <input name="moduleVersionMajor" var="moduleVersionMajor"/>
         <input name="moduleVersionMinor" var="moduleVersionMinor"/>
         <input name="deviceIDOrNull" value="0"/>
         <input name="module" var="_modulehandle"/>
         <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="ModuleHandle" var="_modulehandle"/>
         <input name="Purpose"
              var="__BioAPI_PURPOSE_ENROLL_FOR_VERIFICATION_ONLY"/>
```

```
        <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="be0a1330-d59e-11d8-9669-0800200c9a66"
               break_on_break="true" >
        <input name="ModuleHandle" var="_modulehandle"/>
        <input name="BIRHandle" var="capturedbir_handle"/>
     </invoke>

     <!-- Invoke the function BioSPI_CreateTemplate. -->
     <invoke function="BioSPI_CreateTemplate">
        <input name="ModuleHandle" var="_modulehandle"/>
        <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="ModuleHandle" var="_modulehandle"/>
        <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="ModuleHandle" var="_modulehandle"/>
         <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_ModuleDetach and BioSPI_ModuleUnload -->
      <invoke activity="DetachAndUnload"
           package="be0a1330-d59e-11d8-9669-0800200c9a66" >
         <input name="moduleUuid" var="moduleUuid"/>
         <input name="module" var="_modulehandle"/>
      </invoke>
   </activity>

</package>
```

16.16.3 Assertion 14.2 BioSPI_CreateTemplate_OutputBIRDataType

16.16.3.1 Test Purpose:
To test that BioSPI_CreateTemplate creates a processed BIR.

16.16.3.2 Test 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.

16.16.3.3 Expected Results:
Return code BioAPI_OK.

16.16.3.4 XML:

```
<package name="39ec8660-2c23-11d9-9669-0800200c9a66">
   <author>
      author
   </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 and
without adaptation.  The new template BIR is expected to have the processed
level PROCESSED.
         The relevant text in BioAPI 1.1 is quoted below from subclause 1.6.
         ____________________________________________________________
         BioAPI_RETURN BioAPI BioSPI_CreateTemplate
            (BioAPI_HANDLE ModuleHandle,
            const BioAPI_INPUT_BIR *CapturedBIR,
            const BioAPI_INPUT_BIR *StoredTemplate,
            BioAPI_BIR_HANDLE_PTR NewTemplate,
            const BioAPI_DATA *Payload)

         This function takes a BIR containing raw biometric data for the purpose
of creating a new enrollment emplate. A new BIR is constructed from the
CapturedBIR, and (optionally) it may perform an adaptation based on an existing
StoredTemplate.
         ____________________________________________________________
         Section 1.6:
         CreateTemplate always takes an 'intermediate' BIR as input, and
constructs a template, i.e., a '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 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 fail, an UNDECIDED conformity
```

```
response is issued.
     </description>

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

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

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

     <!-- 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="moduleUuid" var="_moduleUuid"/>
        <input name="moduleVersionMajor" var="_moduleVersionMajor"/>
        <input name="moduleVersionMinor" var="_moduleVersionMinor"/>
        <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="be0a1330-d59e-11d8-9669-0800200c9a66"
          function="BioSPI_ModuleEventHandler"/>
   </assertion>

   <activity name="BioSPI_CreateTemplate">
     <input name="moduleUuid"/>
     <input name="moduleVersionMajor"/>
     <input name="moduleVersionMinor"/>
     <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="_modulehandle" value="1"/>

     <!-- Invoke the functions BioSPI_ModuleLoad and BioSPI_ModuleAttach
exposed by the BSP under test.
          The input value for the parameter "deviceIDOrNull" is "0", therefore
```

```
the assertion will test the BSP's default device. -->
     <invoke activity="LoadAndAttach"
          package="be0a1330-d59e-11d8-9669-0800200c9a66"
          break_on_break="true">
        <input name="moduleUuid" var="moduleUuid"/>
        <input name="moduleVersionMajor" var="moduleVersionMajor"/>
        <input name="moduleVersionMinor" var="moduleVersionMinor"/>
        <input name="deviceIDOrNull" value="0"/>
        <input name="module" var="_modulehandle"/>
        <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="ModuleHandle" var="_modulehandle"/>
        <input name="Purpose"
             var="__BioAPI_PURPOSE_ENROLL_FOR_VERIFICATION_ONLY"/>
        <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="be0a1330-d59e-11d8-9669-0800200c9a66"
          break_on_break="true" >
        <input name="ModuleHandle" var="_modulehandle"/>
        <input name="BIRHandle" var="capturedbir_handle"/>
     </invoke>

     <!-- Invoke the function BioSPI_CreateTemplate. -->
     <invoke function="BioSPI_CreateTemplate">
        <input name="ModuleHandle" var="_modulehandle"/>
        <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="ModuleHandle" var="_modulehandle"/>
        <input name="Handle" var="newTemplate_handle"/>
        <output name="Length" setvar="length"/>
        <output name="HeaderVersion" setvar="headerversion"/>
        <output name="ProcessedLevel" setvar="processedLevel"/>
        <output name="FormatOwner" setvar="formatowner"/>
        <output name="FormatId" setvar="formatid"/>
        <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, 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_ModuleDetach and BioSPI_ModuleUnload -->
     <invoke activity="DetachAndUnload"
           package="be0a1330-d59e-11d8-9669-0800200c9a66" >
        <input name="moduleUuid" var="moduleUuid" />
        <input name="module" var="_modulehandle" />
     </invoke>
   </activity>

</package>
```

16.16.4 Assertion 14.3 BioSPI_CreateTemplate_InputBIRDataType

16.16.4.1 Test Purpose:
To test that BioSPI_CreateTemplate rejects an input BIR of a data type different from BioAPI_BIR_DATA_TYPE_INTERMEDIATE.

16.16.4.2 Test 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) Change the processed level of the BIR to either RAW or 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.

16.16.4.3 Expected Results:
Return code other than BioAPI_OK.

16.16.4.4 XML:

```
<package name="6d543ea0-2ce9-11d9-9669-0800200c9a66">
   <author>
      author
   </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 1.1 is quoted below from subclauses 1.6 and
2.5.3.2.
         ____________________________________________________________
         BioAPI_RETURN BioAPI BioSPI_CreateTemplate
            (BioAPI_HANDLE ModuleHandle,
            const BioAPI_INPUT_BIR *CapturedBIR,
            const BioAPI_INPUT_BIR *StoredTemplate,
```

```
          BioAPI_BIR_HANDLE_PTR NewTemplate,
          const BioAPI_DATA *Payload)

        ____________________________________________________________
        Section 1.6:
        CreateTemplate always takes an 'intermediate' BIR as input, and
constructs a template, i.e., a 'processed' BIR.
        Section 2.5.3.2:
        This function takes a BIR containing raw biometric data 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_GetBIRFromHandle.
          5) Change the processed level of the BIR to either RAW or 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 fail, an UNDECIDED conformity
response is issued.
     </description>

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

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

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

     <!-- 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="moduleUuid" var="_moduleUuid"/>
        <input name="moduleVersionMajor" var="_moduleVersionMajor"/>
        <input name="moduleVersionMinor" var="_moduleVersionMinor"/>
        <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="be0a1330-d59e-11d8-9669-0800200c9a66"
          function="BioSPI_ModuleEventHandler"/>
   </assertion>

   <activity name="BioSPI_CreateTemplate_InputBIRDataType">
      <input name="moduleUuid"/>
      <input name="moduleVersionMajor"/>
      <input name="moduleVersionMinor"/>
      <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="_modulehandle" value="1"/>

     <!-- Invoke the functions BioSPI_ModuleLoad and BioSPI_ModuleAttach
exposed by the BSP under test.
          The input value for the parameter "deviceIDOrNull" is "0", therefore
the assertion will test the BSP's default device. -->
     <invoke activity="LoadAndAttach"
          package="be0a1330-d59e-11d8-9669-0800200c9a66"
          break_on_break="true">
        <input name="moduleUuid" var="moduleUuid"/>
        <input name="moduleVersionMajor" var="moduleVersionMajor"/>
        <input name="moduleVersionMinor" var="moduleVersionMinor"/>
        <input name="deviceIDOrNull" value="0"/>
        <input name="module" var="_modulehandle"/>
        <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="ModuleHandle" var="_modulehandle"/>
        <input name="Purpose"
             var="__BioAPI_PURPOSE_ENROLL_FOR_VERIFICATION_ONLY"/>
        <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="be0a1330-d59e-11d8-9669-0800200c9a66"
          break_on_break="true" >
        <input name="ModuleHandle" var="_modulehandle" />
        <input name="BIRHandle"  var="capturedbir_handle" />
     </invoke>

     <invoke function="BioSPI_GetBIRFromHandle">
        <input name="ModuleHandle" var="_modulehandle"/>
        <input name="Handle" var="capturedbir_handle" />
        <output name="Length" setvar="length"/>
        <output name="HeaderVersion" setvar="headerversion" />
        <output name="ProcessedLevel" setvar="processedLevel" />
        <output name="Encrypted" setvar="encrypted" />
        <output name="Signed" setvar="signed" />
        <output name="FormatOwner" setvar="formatowner" />
        <output name="FormatId" setvar="formatid" />
        <output name="Quality" setvar="quality" />
        <output name="Purpose" setvar="purpose" />
        <output name="FactorMultiple" setvar="factorMultiple"/>
        <output name="FactorFacialFeatures"
             setvar="factorFacialFeatures"/>
        <output name="FactorVoice" setvar="factorVoice"/>
        <output name="FactorFingerprint" setvar="factorFingerprint"/>
        <output name="FactorIris" setvar="factorIris"/>
        <output name="FactorRetina" setvar="factorRetina"/>
        <output name="FactorHandGeometry" setvar="factorHandGeometry"/>
        <output name="FactorSignatureDynamics"
             setvar="factorSignatureDynamics"/>
        <output name="FactorKeystrokeDynamics"
             setvar="factorKeystrokeDynamics"/>
        <output name="FactorLipMovement" setvar="factorLipMovement"/>
        <output name="FactorThermalFaceImage"
             setvar="factorThermalFaceImage"/>
        <output name="FactorThermalHandImage"
             setvar="factorThermalHandImage"/>
```

```
        <output name="FactorGait" setvar="factorGait"/>
        <output name="FactorPassword" setvar="factorPassword"/>
        <output name="BiometricData" setvar="biomtricdata" />
        <output name="Signature" setvar="signature" />
        <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_GetBIRFromHandle has returned BioAPI_OK
        </description>
        <equal_to var1="return" var2="__BioAPI_OK"/>
     </assert_condition>

     <!-- Invoke the function BioSPI_CreateTemplate. -->
     <invoke function="BioSPI_CreateTemplate">
        <input name="ModuleHandle" var="_modulehandle"/>
        <input name="CapturedBIR_Form"
             var="__BioAPI_FULLBIR_INPUT"/>
        <input name="CapturedBIR_Length" var="length"/>
        <input name="CapturedBIR_HeaderVersion"
             var="headerversion" />
        <input name="CapturedBIR_ProcessedLevel"
             var="__BioAPI_BIR_DATA_TYPE_PROCESSED" />
        <input name="CapturedBIR_Encrypted" var="encrypted" />
        <input name="CapturedBIR_Signed" var="signed" />
        <input name="CapturedBIR_FormatOwner" var="formatowner" />
        <input name="CapturedBIR_FormatId" var="formatid" />
        <input name="CapturedBIR_Quality" var="quality" />
        <input name="CapturedBIR_Purpose" var="purpose" />
        <input name="CapturedBIR_FactorMultiple"
             var="factorMultiple"/>
        <input name="CapturedBIR_FactorFacialFeatures"
             var="factorFacialFeatures"/>
        <input name="CapturedBIR_FactorVoice" var="factorVoice"/>
        <input name="CapturedBIR_FactorFingerprint"
             var="factorFingerprint"/>
        <input name="CapturedBIR_FactorIris" var="factorIris"/>
        <input name="CapturedBIR_FactorRetina" var="factorRetina"/>
        <input name="CapturedBIR_FactorHandGeometry"
             var="factorHandGeometry"/>
        <input name="CapturedBIR_FactorSignatureDynamics"
             var="factorSignatureDynamics"/>
        <input name="CapturedBIR_FactorKeystrokeDynamics"
             var="factorKeystrokeDynamics"/>
        <input name="CapturedBIR_FactorLipMovement"
             var="factorLipMovement"/>
        <input name="CapturedBIR_FactorThermalFaceImage"
             var="factorThermalFaceImage"/>
        <input name="CapturedBIR_FactorThermalHandImage"
             var="factorThermalHandImage"/>
        <input name="CapturedBIR_FactorGait" var="factorGait"/>
        <input name="CapturedBIR_FactorPassword"
             var="factorPassword"/>
        <input name="CapturedBIR_BiometricData"
             var="biomtricdata" />
        <input name="CapturedBIR_Signature" var="signature" />
        <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="be0a1330-d59e-11d8-9669-0800200c9a66" >
        <input name="moduleUuid" var="moduleUuid" />
        <input name="module" var="_modulehandle" />
      </invoke>
   </activity>

</package>
```

16.16.5 Assertion 14.4 BioSPI_CreateTemplate_Purpose

16.16.5.1 Test Purpose:
To test that the purpose of the template BIR is either "enroll_verify" and/or "enroll_identify".

16.16.5.2 Test 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) Change the propose to PURPOSE_VERIFY.
6) Call BioSPI_CreateTemplate specifying the input BIR.
7) Check if the return value is equal to __BioAPIERR_BSP_INCONSISTENT_PURPOSE.
8) Detach and unload the BSP under test.

16.16.5.3 Expected Results:
Return code BioAPIERR_BSP_INCONSISTENT_PURPOSE.

16.16.5.4 XML:

```
<package name="25e3d1b0-2cf4-11d9-9669-0800200c9a66">
   <author>
      author
   </author>

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

   <assertion name="BioSPI_CreateTemplate_Purpose" model="BSPTesting">
      <description>
         This assertion invokes the function BioSPI_CreateTemplate with an
invalid input BIR purpose.
         The relevant text in BioAPI 1.1 is quoted below from subclauses 1.6 and
```

2.1.10.

```
        ____________________________________________________________
        BioAPI_RETURN BioAPI BioSPI_CreateTemplate
           (BioAPI_HANDLE ModuleHandle,
           const BioAPI_INPUT_BIR *CapturedBIR,
           const BioAPI_INPUT_BIR *StoredTemplate,
           BioAPI_BIR_HANDLE_PTR NewTemplate,
           const BioAPI_DATA *Payload)

        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.
        ____________________________________________________________
        Section 1.6:
        CreateTemplate is provided to perform the processing of samples for the
construction of an enrollment template.
        CreateTemplate always takes an 'intermediate' BIR as input, and
constructs a template (i.e., a 'processed' BIR with the recorded purpose of
either 'enroll_verify' and/or 'enroll_identify').
        Section 2.1.10:
        g) The Create_Template function may accept as input any intermediate BIR
with a Purpose including Enroll, Enroll_for_Verification_Only, and/or
Enroll_for_Identification, and will output only BIRs with a Purpose including
that of the input BIR.
        (purpose of either 'enroll_verify' and/or 'enroll_identify').
        ____________________________________________________________

        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) Change the propose to PURPOSE_VERIFY.
           6) Call BioSPI_CreateTemplate specifying the input BIR.
           7) Check if the return value is equal to
__BioAPIERR_BSP_INCONSISTENT_PURPOSE.
           8) Detach and unload the BSP under test.

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

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

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

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

     <!-- 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_Purpose">
        <input name="moduleUuid" var="_moduleUuid"/>
        <input name="moduleVersionMajor" var="_moduleVersionMajor"/>
        <input name="moduleVersionMinor" var="_moduleVersionMinor"/>
        <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="be0a1330-d59e-11d8-9669-0800200c9a66"
          function="BioSPI_ModuleEventHandler"/>
   </assertion>

   <activity name="BioSPI_CreateTemplate_Purpose">
     <input name="moduleUuid"/>
     <input name="moduleVersionMajor"/>
     <input name="moduleVersionMinor"/>
     <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="_modulehandle" value="1"/>

     <!-- Invoke the functions BioSPI_ModuleLoad and BioSPI_ModuleAttach
exposed by the BSP under test.
          The input value for the parameter "deviceIDOrNull" is "0", therefore
the assertion will test the BSP's default device. -->
     <invoke activity="LoadAndAttach"
          package="be0a1330-d59e-11d8-9669-0800200c9a66"
          break_on_break="true">
        <input name="moduleUuid" var="moduleUuid"/>
        <input name="moduleVersionMajor" var="moduleVersionMajor"/>
        <input name="moduleVersionMinor" var="moduleVersionMinor"/>
        <input name="deviceIDOrNull" value="0"/>
        <input name="module" var="_modulehandle"/>
        <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="ModuleHandle" var="_modulehandle"/>
     <input name="Purpose"
          var="__BioAPI_PURPOSE_ENROLL_FOR_VERIFICATION_ONLY"/>
     <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="be0a1330-d59e-11d8-9669-0800200c9a66"
          break_on_break="true" >
        <input name="ModuleHandle" var="_modulehandle"/>
        <input name="BIRHandle" var="capturedbir_handle"/>
     </invoke>

     <invoke function="BioSPI_GetBIRFromHandle">
        <input name="ModuleHandle" var="_modulehandle"/>
        <input name="Handle" var="capturedbir_handle"/>
        <output name="Length" setvar="length"/>
        <output name="HeaderVersion" setvar="headerversion"/>
        <output name="ProcessedLevel" setvar="processedLevel"/>
        <output name="Encrypted" setvar="encrypted"/>
        <output name="Signed" setvar="signed"/>
        <output name="FormatOwner" setvar="formatowner"/>
        <output name="FormatId" setvar="formatid"/>
        <output name="Quality" setvar="quality"/>
        <output name="Purpose" setvar="purpose"/>
```

```
        <output name="FactorMultiple" setvar="factorMultiple"/>
        <output name="FactorFacialFeatures"
              setvar="factorFacialFeatures"/>
        <output name="FactorVoice" setvar="factorVoice"/>
        <output name="FactorFingerprint"
              setvar="factorFingerprint"/>
        <output name="FactorIris" setvar="factorIris"/>
        <output name="FactorRetina" setvar="factorRetina"/>
        <output name="FactorHandGeometry"
              setvar="factorHandGeometry"/>
        <output name="FactorSignatureDynamics"
              setvar="factorSignatureDynamics"/>
        <output name="FactorKeystrokeDynamics"
              setvar="factorKeystrokeDynamics"/>
        <output name="FactorLipMovement"
              setvar="factorLipMovement"/>
        <output name="FactorThermalFaceImage"
              setvar="factorThermalFaceImage"/>
        <output name="FactorThermalHandImage"
              setvar="factorThermalHandImage"/>
        <output name="FactorGait" setvar="factorGait"/>
        <output name="FactorPassword" setvar="factorPassword"/>
        <output name="BiometricData" setvar="biomtricdata"/>
        <output name="Signature" setvar="signature"/>
        <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_GetBIRFromHandle has returned BioAPI_OK.
        </description>
        <equal_to var1="return" var2="__BioAPI_OK"/>
     </assert_condition>

     <!-- Invoke the function BioSPI_CreateTemplate. -->
     <invoke function="BioSPI_CreateTemplate">
        <input name="ModuleHandle" var="_modulehandle"/>
        <input name="CapturedBIR_Form"
              var="__BioAPI_FULLBIR_INPUT"/>
        <input name="CapturedBIR_Length" var="length"/>
        <input name="CapturedBIR_HeaderVersion"
              var="headerversion"/>
        <input name="CapturedBIR_ProcessedLevel"
              var="processedLevel"/>
        <input name="CapturedBIR_Encrypted" var="encrypted"/>
        <input name="CapturedBIR_Signed" var="signed"/>
        <input name="CapturedBIR_FormatOwner" var="formatowner"/>
        <input name="CapturedBIR_FormatId" var="formatid"/>
        <input name="CapturedBIR_Quality" var="quality"/>
        <input name="CapturedBIR_Purpose"
              var="__BioAPI_PURPOSE_VERIFY"/>
        <input name="CapturedBIR_FactorMultiple"
              var="factorMultiple"/>
        <input name="CapturedBIR_FactorFacialFeatures"
              var="factorFacialFeatures"/>
        <input name="CapturedBIR_FactorVoice" var="factorVoice"/>
        <input name="CapturedBIR_FactorFingerprint"
              var="factorFingerprint"/>
```

```
        <input name="CapturedBIR_FactorIris" var="factorIris"/>
        <input name="CapturedBIR_FactorRetina" var="factorRetina"/>
        <input name="CapturedBIR_FactorHandGeometry"
             var="factorHandGeometry"/>
        <input name="CapturedBIR_FactorSignatureDynamics"
             var="factorSignatureDynamics"/>
        <input name="CapturedBIR_FactorKeystrokeDynamics"
             var="factorKeystrokeDynamics"/>
        <input name="CapturedBIR_FactorLipMovement"
             var="factorLipMovement"/>
        <input name="CapturedBIR_FactorThermalFaceImage"
             var="factorThermalFaceImage"/>
        <input name="CapturedBIR_FactorThermalHandImage"
             var="factorThermalHandImage"/>
        <input name="CapturedBIR_FactorGait" var="factorGait"/>
        <input name="CapturedBIR_FactorPassword"
             var="factorPassword"/>
        <input name="CapturedBIR_BiometricData"
             var="biomtricdata" />
        <input name="CapturedBIR_Signature" var="signature" />
        <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_BSP_INCONSISTENT_PURPOSE.
        </description>
        <equal_to var1="return"
             var2="__BioAPIERR_BSP_INCONSISTENT_PURPOSE"/>
     </assert_condition>

     <!-- Invoke the functions BioSPI_ModuleDetach and BioSPI_ModuleUnload -->
     <invoke activity="DetachAndUnload"
          package="be0a1330-d59e-11d8-9669-0800200c9a66" >
        <input name="moduleUuid" var="moduleUuid"/>
        <input name="module" var="_modulehandle"/>
     </invoke>
   </activity>
</package>
```

16.17 Feature 14a. Accept input of stored template to return update/adapted template

16.17.1 BioAPI Specification References:
3.3.4.2

16.17.2 **Assertion 14a.1 BioSPI_CreateTemplate_StoredTemplateUnchanged**

16.17.2.1 Test Purpose:
If the BSP supports template adaptation, test that the StoredTemplate remains unchanged.

16.17.2.2 Test Scenario:
BioSPI_Create_Template is passed a valid CapturedBIR and StoredTemplate.

16.17.2.3 Expected Results:
Return code BioAPI_OK and an unchanged StoredTemplate.

16.17.2.4 XML:

```
<package name="ac81ba70-2c27-11d9-9669-0800200c9a66">
   <author>
      author
   </author>
   <description>
      This package contains the assertion
"BioSPI_CreateTemplate_StoredTemplateUnchanged" (see the "description" element
of the assertion below).
   </description>
   <assertion name="BioSPI_CreateTemplate_StoredTemplateUnchanged"
model="BSPTesting">
      <description>
         This assertion tests BioSPI_CreateTemplate with adaptation.
         The relevant text in BioAPI 1.1 is quoted below from subclauses 2.5.3.2,
2.2.1.2 and 4.2.4.2.
         ____________________________________________________________
         BioAPI_RETURN BioAPI BioSPI_CreateTemplate
            (BioAPI_HANDLE ModuleHandle,
            const BioAPI_INPUT_BIR *CapturedBIR,
            const BioAPI_INPUT_BIR *StoredTemplate,
            BioAPI_BIR_HANDLE_PTR NewTemplate,
            const BioAPI_DATA *Payload)

         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. The old StoredTemplate remains unchanged.
         ____________________________________________________________
         Section 2.5.3.2:
         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.
         The old StoredTemplate remains unchanged.
         Section 2.2.1.2:
         #define BioAPI_ADAPTATION (0x00020000)
         If set, BSP supports BIR adaptation (return of Verify or VerifyMatch
operation).
         Section 4.2.4.2:
         The BSP makes the decision as to when and if the adaptation should be
performed (based on such factors as quality, elapsed time, and significant
differences).
         If the BSP does not support adaptation, the returned AdaptedBIR will be
set to -2.
         If the BSP supports adaptation, but for some reason is not able or
chooses not to perform the adaptation, then the return AdaptedBIR is set to -1.
         The BSP must post to the module registry its support for adaptation.
         NOTE: This assertion assumes that the same BioAPI_ADAPTATION option is
used for specifying support of adaptation in the CreateTemplate 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) Call BioSPI_Capture with a purpose of
BioAPI_PURPOSE_ENROLL_FOR_VERIFICATION_ONLY to obtain a BIR for enrollment
          4) Call BioSPI_CreateTemplate with StoredTemplate set to NULL, to
obtain template1
          5) Call BioSPI_Capture with a purpose of
BioAPI_PURPOSE_ENROLL_FOR_VERIFICATION_ONLY to obtain a new BIR for enrollment
          6) Call BioSPI_CreateTemplate with StoredTemplate set to template1 to
obtain template2
          7) Check the return code
          8) Check the StoredTemplate is unchanged after adaptation
          9) Detach and Unload the BSP under test.

        If any of the intermediate operations fail, an UNDECIDED conformity
response is issued.
     </description>
     <!-- UUID of the BSP under test -->
     <input name="_moduleUuid"/>
     <!-- Major version number of the BSP under test -->
     <input name="_moduleVersionMajor"/>
     <!-- Minor version number of the BSP under test -->
     <input name="_moduleVersionMinor"/>
     <!-- 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 template
adaptation -->
     <input name="_supportAdaptation"/>
     <!-- Invocation of the primary activity of this assertion with input
parameter values assigned from the assertion's parameters. -->
     <invoke activity="BioSPI_CreateTemplate_StoredTemplateUnchanged">
        <input name="moduleUuid" var="_moduleUuid"/>
        <input name="moduleVersionMajor" var="_moduleVersionMajor"/>
        <input name="moduleVersionMinor" var="_moduleVersionMinor"/>
        <input name="inserttimeouttime" var="_inserttimeout"/>
        <input name="nosourcepresentsupported" var="_noSourcePresentSupported"/>
        <input name="sourcepresenttimeouttime" var="_sourcepresenttimeout"/>
        <input name="capturetimeouttime" var="_capturetimeout"/>
        <input name="supportAdaptation" var="_supportAdaptation"/>
     </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="be0a1330-d59e-11d8-9669-
0800200c9a66" function="BioSPI_ModuleEventHandler"/>
   </assertion>
   <activity name="BioSPI_CreateTemplate_StoredTemplateUnchanged">
     <input name="moduleUuid"/>
     <input name="moduleVersionMajor"/>
     <input name="moduleVersionMinor"/>
     <input name="inserttimeouttime"/>
     <input name="nosourcepresentsupported"/>
     <input name="sourcepresenttimeouttime"/>
     <input name="capturetimeouttime"/>
     <input name="supportAdaptation"/>
     <!-- This assertion will use ModuleHandle "1" for all BioSPI calls that
```

```
require it -->
     <set name="_modulehandle" value="1"/>
     <!-- Invoke the functions BioSPI_ModuleLoad and BioSPI_ModuleAttach
exposed by the BSP under test.
          The input value for the parameter "deviceIDOrNull" is "0", therefore
the assertion will test the BSP's default device. -->
     <invoke activity="LoadAndAttach" package="be0a1330-d59e-11d8-9669-
0800200c9a66" break_on_break="true">
        <input name="moduleUuid" var="moduleUuid"/>
        <input name="moduleVersionMajor" var="moduleVersionMajor"/>
        <input name="moduleVersionMinor" var="moduleVersionMinor"/>
        <input name="deviceIDOrNull" value="0"/>
        <input name="module" var="_modulehandle"/>
        <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="ModuleHandle" var="_modulehandle"/>
        <input name="Purpose"
var="__BioAPI_PURPOSE_ENROLL_FOR_VERIFICATION_ONLY"/>
        <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="be0a1330-d59e-11d8-
9669-0800200c9a66" break_on_break="true">
        <input name="ModuleHandle" var="_modulehandle"/>
```

```
        <input name="BIRHandle" var="capturedbir_handle"/>
      </invoke>
      <!-- Invoke the function BioSPI_CreateTemplate. -->
      <invoke function="BioSPI_CreateTemplate">
        <input name="ModuleHandle" var="_modulehandle"/>
        <input name="CapturedBIR_Form" var="__BioAPI_BIR_HANDLE_INPUT"/>
        <input name="CapturedBIR_BIRHandle" var="capturedbir_handle"/>
        <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 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 BIR handle
"template_handle1" to compare the header with the header of the adapted BIR
later. -->
      <invoke function="BioSPI_GetHeaderFromHandle">
        <input name="ModuleHandle" var="_modulehandle"/>
        <input name="Handle" var="template_handle1"/>
        <output name="Length" setvar="length1"/>
        <output name="HeaderVersion" setvar="headerversion1"/>
        <output name="ProcessedLevel" setvar="processedLevel1"/>
        <output name="FormatOwner" setvar="formatowner1"/>
        <output name="FormatId" setvar="formatid1"/>
        <output name="Quality" setvar="quality1"/>
        <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_Capture to capture another BIR.-->
      <invoke function="BioSPI_Capture">
        <input name="ModuleHandle" var="_modulehandle"/>
        <input name="Purpose"
var="__BioAPI_PURPOSE_ENROLL_FOR_VERIFICATION_ONLY"/>
        <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_CreateTemplate to test adaptation. -->
     <invoke function="BioSPI_CreateTemplate">
        <input name="ModuleHandle" var="_modulehandle"/>
        <input name="CapturedBIR_Form" var="__BioAPI_BIR_HANDLE_INPUT"/>
        <input name="CapturedBIR_BIRHandle" var="capturedbir_handle"/>
        <input name="StoredTemplate_Form" var="__BioAPI_BIR_HANDLE_INPUT"/>
        <input name="StoredTemplate_BIRHandle" var="template_handle1"/>
        <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>
           The function BioSPI_CreateTemplate has returned either BioAPI_OK (if
the BSP claims to supports adaptation), or an error (if the BSP does not claim
to support adaptation).
        </description>
        <or>
           <and>
              <same_as var1="supportAdaptation" value2="true"/>
              <equal_to var1="return" var2="__BioAPI_OK"/>
           </and>
           <and>
              <same_as var1="supportAdaptation" value2="false"/>
              <not_equal_to var1="return" var2="__BioAPI_OK"/>
           </and>
        </or>
     </assert_condition>
     <!-- Check that the stored template is unchanged after adaptation -->
     <invoke function="BioSPI_GetHeaderFromHandle">
        <input name="ModuleHandle" var="_modulehandle"/>
        <input name="Handle" var="template_handle1"/>
        <output name="Length" setvar="length2"/>
        <output name="HeaderVersion" setvar="headerversion2"/>
        <output name="ProcessedLevel" setvar="processedLevel2"/>
        <output name="FormatOwner" setvar="formatowner2"/>
        <output name="FormatId" setvar="formatid2"/>
        <output name="Quality" setvar="quality2"/>
        <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 stored template has not changed after adaptation.
```

```
        </description>
        <equal_to var1="length1" var2="length2"/>
        <equal_to var1="headerversion1" var2="headerversion2"/>
        <equal_to var1="processedLevel1" var2="processedLevel2"/>
        <equal_to var1="formatowner1" var2="formatowner2"/>
        <equal_to var1="formatid1" var2="formatid2"/>
        <equal_to var1="quality1" var2="quality2"/>
        <equal_to var1="purpose1" var2="purpose2"/>
     </assert_condition>
     <!-- Invoke the functions BioSPI_ModuleDetach and BioSPI_ModuleUnload -->
     <invoke activity="DetachAndUnload" package="be0a1330-d59e-11d8-9669-
0800200c9a66">
        <input name="moduleUuid" var="moduleUuid"/>
        <input name="module" var="_modulehandle"/>
     </invoke>
   </activity>
</package>
```

16.18 Feature 14b. Acceptance of payload for inclusion of enrollment BIR

16.18.1 BioAPI Specification References:
3.3.4.2

16.18.2 Assertion 14b.1 BioSPI_CreateTemplate_PayloadSupported

16.18.2.1 Test Purpose:
To test Payload if supported by the BSP.

16.18.2.2 Test Scenario:
If the BSP supports Payload, call BioSPI_CreateTemplate with valid parameters including Payload.

16.18.2.3 Expected Results:
Return code BioAPI_OK and a valid NewTemplate that includes the Payload.

16.18.2.4 XML:

```
<package name="d31ceee0-2c43-11d9-9669-0800200c9a66">
   <author>
      author
   </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 with valid parameters
and payload.
         The relevant text in BioAPI 1.1 is quoted below from subclauses 1.6,
2.2.1.2 and 2.5.3.2.
         ____________________________________________________________
         BioAPI_RETURN BioAPI BioSPI_CreateTemplate
```

```
          (BioAPI_HANDLE ModuleHandle,
          const BioAPI_INPUT_BIR *CapturedBIR,
          const BioAPI_INPUT_BIR *StoredTemplate,
          BioAPI_BIR_HANDLE_PTR NewTemplate,
          const BioAPI_DATA *Payload)

        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. The old StoredTemplate remains unchanged. If the StoredTemplate
contains a payload, the payload is not copied into the NewTemplate. If the
NewTemplate needs a payload, then that Payload must be presented as an argument
to the function.
        ____________________________________________________________
        Section 1.6:
        The BSP may optionally allow the application to provide a 'payload' to
wrap inside the new template.
        Section 2.2.1.2:
        #define BioAPI_PAYLOAD (0x00001000)
        If set, indicates that the BSP supports payload carry (accepts payload
during enroll/process and returns payroll upon successful verify).
        Section 2.5.3.2:
        If the NewTemplate needs a payload, then that Payload must be presented
as an argument to the 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) 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 fail, an UNDECIDED conformity
response is issued.
     </description>

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

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

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

     <!-- 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 -->
     <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="moduleUuid" var="_moduleUuid"/>
        <input name="moduleVersionMajor" var="_moduleVersionMajor"/>
        <input name="moduleVersionMinor" var="_moduleVersionMinor"/>
        <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="be0a1330-d59e-11d8-9669-0800200c9a66"
          function="BioSPI_ModuleEventHandler"/>
   </assertion>

   <activity name="BioSPI_CreateTemplate_PayloadSupported">
     <input name="moduleUuid"/>
     <input name="moduleVersionMajor"/>
     <input name="moduleVersionMinor"/>
     <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="_modulehandle" value="1"/>

     <!-- Invoke the functions BioSPI_ModuleLoad and BioSPI_ModuleAttach
exposed by the BSP under test.
          The input value for the parameter "deviceIDOrNull" is "0", therefore
the assertion will test the BSP's default device. -->
     <invoke activity="LoadAndAttach"
          package="be0a1330-d59e-11d8-9669-0800200c9a66"
          break_on_break="true">
        <input name="moduleUuid" var="moduleUuid"/>
        <input name="moduleVersionMajor" var="moduleVersionMajor"/>
        <input name="moduleVersionMinor" var="moduleVersionMinor"/>
        <input name="deviceIDOrNull" value="0"/>
        <input name="module" var="_modulehandle"/>
        <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="ModuleHandle" var="_modulehandle"/>
        <input name="Purpose"
             var="__BioAPI_PURPOSE_ENROLL_FOR_VERIFICATION_ONLY"/>
        <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="be0a1330-d59e-11d8-9669-0800200c9a66"
          break_on_break="true">
        <input name="ModuleHandle" var="_modulehandle"/>
        <input name="BIRHandle" var="capturedbir_handle"/>
     </invoke>

     <!-- Invoke the function BioSPI_CreateTemplate. -->
     <invoke function="BioSPI_CreateTemplate">
        <input name="ModuleHandle" var="_modulehandle"/>
        <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_WRAP_PAYLOAD" />
            <same_as var1="supportpayload" value2="false"/>
          </and>
        </or>
     </assert_condition>

     <!-- Invoke the functions BioSPI_ModuleDetach and BioSPI_ModuleUnload -->
     <invoke activity="DetachAndUnload"
          package="be0a1330-d59e-11d8-9669-0800200c9a66" >
        <input name="moduleUuid" var="moduleUuid" />
        <input name="module" var="_modulehandle" />
     </invoke>
   </activity>

</package>
```

16.18.3 Assertion 14b.2 BioSPI_CreateTemplate_PayloadNotCopied

16.18.3.1 Test Purpose:
If the BSP supports template adaptation and a BIR payload, test that the payload is not copied from StoredTemplate to NewTemplate.

16.18.3.2 Test Scenario:
BioSPI_Create_Template is passed a valid CapturedBIR and StoredTemplate. The StoredTemplate must contain a payload. The Payload parameter is NULL.

16.18.3.3 Expected Results:
Return code BioAPI_OK and a valid NewTemplate that does not contain a payload.

16.18.3.4 XML:

16.19 Feature 14c. Return of quality in the processed BIR header

16.19.1 BioAPI Specification References:
3.3.4.2, (2.1.46, 4.2.4.2)

16.19.2 Assertion 14c.1 BioSPI_CreateTemplate_BIRHeaderQuality

16.19.2.1 Test Purpose:
To test the return of quality in the enrollment BIR header if supported by the BSP.

16.19.2.2 Test Scenario:
If the BSP supports return of quality, call BioSPI_CreateTemplate with valid parameters.

16.19.2.3 Expected Results:
Return code BioAPI_OK and a valid NewTemplate that includes the quality.

16.19.2.4 XML:

```
<package name="3712a3e0-2c4e-11d9-9669-0800200c9a66">
   <author>
      author
   </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 1.1 is quoted below from subclauses 2.2.1.2
and 4.2.4.2.
         ____________________________________________________________
         BioAPI_RETURN BioAPI BioSPI_CreateTemplate
            (BioAPI_HANDLE ModuleHandle,
            const BioAPI_INPUT_BIR *CapturedBIR,
            const BioAPI_INPUT_BIR *StoredTemplate,
            BioAPI_BIR_HANDLE_PTR NewTemplate,
            const BioAPI_DATA *Payload)

         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. The old StoredTemplate remains unchanged.
         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.
         ____________________________________________________________
         Section 2.2.1.2:
         #define BioAPI_QUALITY_PROCESSED (0x00000008)
         If set, BSP supports the return of quality value (in the BIR header) for
processed biometric data.
         Section 4.2.4.2:
         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 fail, an UNDECIDED conformity response is issued.

```
      </description>

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

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

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

      <!-- 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="moduleUuid" var="_moduleUuid"/>
         <input name="moduleVersionMajor" var="_moduleVersionMajor"/>
         <input name="moduleVersionMinor" var="_moduleVersionMinor"/>
         <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="be0a1330-d59e-11d8-9669-0800200c9a66"
          function="BioSPI_ModuleEventHandler"/>
   </assertion>

   <activity name="BioSPI_CreateTemplate_ReturnQuality">
      <input name="moduleUuid"/>
      <input name="moduleVersionMajor"/>
      <input name="moduleVersionMinor"/>
      <input name="inserttimeouttime"/>
      <input name="nosourcepresentsupported"/>
      <input name="sourcepresenttimeouttime"/>
      <input name="capturetimeouttime"/>
      <input name="processedQualitySupported"/>

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

      <!-- Invoke the functions BioSPI_ModuleLoad and BioSPI_ModuleAttach
exposed by the BSP under test.
           The input value for the parameter "deviceIDOrNull" is "0", therefore
the assertion will test the BSP's default device. -->
      <invoke activity="LoadAndAttach"
           package="be0a1330-d59e-11d8-9669-0800200c9a66"
           break_on_break="true">
         <input name="moduleUuid" var="moduleUuid"/>
         <input name="moduleVersionMajor" var="moduleVersionMajor"/>
         <input name="moduleVersionMinor" var="moduleVersionMinor"/>
         <input name="deviceIDOrNull" value="0"/>
         <input name="module" var="_modulehandle"/>
         <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="ModuleHandle" var="_modulehandle"/>
```

```
        <input name="Purpose"
             var="__BioAPI_PURPOSE_ENROLL_FOR_VERIFICATION_ONLY"/>
        <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="be0a1330-d59e-11d8-9669-0800200c9a66"
          break_on_break="true" >
        <input name="ModuleHandle" var="_modulehandle"/>
        <input name="BIRHandle"  var="capturedbir_handle"/>
     </invoke>

     <!-- Invoke the function BioSPI_CreateTemplate. -->
     <invoke function="BioSPI_CreateTemplate">
        <input name="ModuleHandle" var="_modulehandle"/>
        <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="ModuleHandle" var="_modulehandle"/>
        <input name="Handle" var="newTemplate_handle"/>
        <output name="Length" setvar="length"/>
        <output name="HeaderVersion" setvar="headerversion"/>
        <output name="ProcessedLevel" setvar="ProcessedLevel"/>
        <output name="FormatOwner" setvar="formatowner"/>
        <output name="FormatId" setvar="formatid"/>
```

```
        <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, 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="be0a1330-d59e-11d8-9669-0800200c9a66"
          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="be0a1330-d59e-11d8-9669-0800200c9a66"
          break_on_break="true">
        <only_if>
          <same_as var1="processedQualitySupported"
               value2="false"/>
        </only_if>
        <input name="quality" var="quality"/>
     </invoke>

     <!-- Invoke the functions BioSPI_ModuleDetach and BioSPI_ModuleUnload -->
     <invoke activity="DetachAndUnload"
          package="be0a1330-d59e-11d8-9669-0800200c9a66" >
        <input name="moduleUuid" var="moduleUuid"/>
        <input name="module" var="_modulehandle"/>
     </invoke>
   </activity>

</package>
```

16.20 Feature 14d. BIR signing (by BSP)

16.20.1 BioAPI Specification References:
1.5, 2.1.7

16.20.2 Assertion 14d.1 BioSPI_CreateTemplate_BIRSigned

16.20.2.1 Test Purpose:
To test BIR signing if supported by the BSP.

16.20.2.2 Test Scenario:
If the BSP supports signing cal BioSPI_CreateTemplate with valid parameters.

16.20.2.3 Expected Results:
Return code BioAPI_OK and a valid NewTemplate that includes a signature.

16.20.2.4 XML:

16.21 Feature 14e. BIR encryption (by BSP)

16.21.1 BioAPI Specification References:
1.5, 2.1.7

16.21.2 Assertion 14e.1 BioSPI_CreateTemplate_BIREncrypted

16.21.2.1 Test Purpose:
To test BIR encryption if supported by the BSP.

16.21.2.2 Test Scenario:
If the BSP supports BIR encryption call BioSPI_CreateTemplatel with valid parameters.

16.21.2.3 Expected Results:
Return code BioAPI_OK and a valid NewTemplate.

16.21.2.4 XML:

16.22 Feature 15. BioSPI Process

16.22.1 BioAPI Specification References:
3.3.4.3

16.22.2 Assertion 15.1 BioSPI_Process_ValidParam

16.22.2.1 Test Purpose:
If the attached BSP has processing capability, test that the BSP builds a ProcessedBIR from an "intermediate" CapturedBIR.

16.22.2.2 Test Scenario:
BioSPI_Process is passed a valid "intermediate" CapturedBIR.

16.22.2.3 Expected Results:
Return code BioAPI_OK and a valid ProcessedBIR.

16.22.2.4 XML:

```
<package name="f3604490-2d01-11d9-9669-0800200c9a66">
   <author>
      author
   </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 1.1 is quoted below from subclause 2.5.3.3.
         ____________________________________________________________
         BioAPI_RETURN BioAPI BioSPI_Process
            (BioAPI_HANDLE ModuleHandle,
            const BioAPI_INPUT_BIR *CapturedBIR,
            BioAPI_BIR_HANDLE_PTR ProcessedBIR);

         This function processes the intermediate data captured via a call to
BioSPI_Capture for the purpose of either verification or identification. If the
processing capability is in the attached BSP, the BSP builds a processed BIR,
otherwise ProcessedBIR is set to NULL.
         ____________________________________________________________
         Section 2.5.3.3:
         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 in the attached BSP, the BSP builds a
processed' BIR, otherwise, ProcessedBIR is set to NULL.
         ____________________________________________________________

         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 fail, an UNDECIDED conformity
response is issued.
      </description>
      <!-- UUID of the BSP under test -->
      <input name="_moduleUuid"/>
      <!-- Major version number of the BSP under test -->
      <input name="_moduleVersionMajor"/>
      <!-- Minor version number of the BSP under test -->
      <input name="_moduleVersionMinor"/>
      <!-- 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="moduleUuid" var="_moduleUuid"/>
         <input name="moduleVersionMajor" var="_moduleVersionMajor"/>
         <input name="moduleVersionMinor" var="_moduleVersionMinor"/>
         <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="be0a1330-d59e-11d8-9669-
0800200c9a66" function="BioSPI_ModuleEventHandler"/>
   </assertion>
   <activity name="BioSPI_Process">
      <input name="moduleUuid"/>
      <input name="moduleVersionMajor"/>
      <input name="moduleVersionMinor"/>
      <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="_modulehandle" value="1"/>
      <!-- Invoke the functions BioSPI_ModuleLoad and BioSPI_ModuleAttach
exposed by the BSP under test.
           The input value for the parameter "deviceIDOrNull" is "0", therefore
the assertion will test the BSP's default device. -->
      <invoke activity="LoadAndAttach" package="be0a1330-d59e-11d8-9669-
0800200c9a66" break_on_break="true">
         <input name="moduleUuid" var="moduleUuid"/>
         <input name="moduleVersionMajor" var="moduleVersionMajor"/>
         <input name="moduleVersionMinor" var="moduleVersionMinor"/>
         <input name="deviceIDOrNull" value="0"/>
         <input name="module" var="_modulehandle"/>
         <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="ModuleHandle" var="_modulehandle"/>
         <input name="Purpose" var="__BioAPI_PURPOSE_VERIFY"/>
         <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="be0a1330-d59e-11d8-
9669-0800200c9a66" break_on_break="true">
        <input name="ModuleHandle" var="_modulehandle"/>
        <input name="BIRHandle" var="capturedbir_handle"/>
     </invoke>
     <!-- Invoke the function BioSPI_Process.-->
     <invoke function="BioSPI_Process">
        <input name="ModuleHandle" var="_modulehandle"/>
        <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="ModuleHandle" var="_modulehandle"/>
        <input name="Handle" var="processedbir_handle"/>
        <output name="Length" setvar="length"/>
        <output name="HeaderVersion" setvar="headerversion"/>
        <output name="ProcessedLevel" setvar="processedLevel"/>
        <output name="FormatOwner" setvar="formatowner"/>
        <output name="FormatId" setvar="formatid"/>
        <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, 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_ModuleDetach and BioSPI_ModuleUnload -->
     <invoke activity="DetachAndUnload" package="be0a1330-d59e-11d8-9669-
0800200c9a66">
        <input name="moduleUuid" var="moduleUuid"/>
        <input name="module" var="_modulehandle"/>
     </invoke>
   </activity>
</package>
```

16.22.3 Assertion 15.2 BioSPI_Process_CannotProcess

16.22.3.1 Test Purpose:
If the attached BSP does not have processing capability, test that the BSP returns a NULL ProcessedBIR from an "intermediate" CapturedBIR.

16.22.3.2 Test Scenario:
BioSPI_Process is passed a valid "intermediate" CapturedBIR.

16.22.3.3 Expected Results:
Return code BioAPI_OK and ProcessedBIR is NULL.

16.22.3.4 XML:

16.22.4 Assertion 15.3 BioSPI_Process_BuildsProcessedBIR

16.22.4.1 Test Purpose:
To test that the BIR returned by BioSPI_Process is processed.

16.22.4.2 Test 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.

16.22.4.3 Expected Results:
If the BSP has processing capability, a processed BIR. Otherwise a NULL processed BIR is returned.

16.22.4.4 XML:

```
<package name="d555ce30-2d0f-11d9-9669-0800200c9a66">
   <author>
      author
   </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 1.1 is quoted below from subclauses 1.6 and
2.5.3.3.
         ____________________________________________________________
         BioAPI_RETURN BioAPI BioSPI_Process
            (BioAPI_HANDLE ModuleHandle,
            const BioAPI_INPUT_BIR *CapturedBIR,
            BioAPI_BIR_HANDLE_PTR ProcessedBIR);

         This function processes the intermediate data captured via a call to
BioSPI_Capture for the purpose of either verification or identification. If the
processing capability is in the attached BSP, the BSP builds a processed BIR,
otherwise, ProcessedBIR is set to NULL.
         ____________________________________________________________
         Section 2.5.3.3:
         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 in the attached BSP, the BSP builds a
'processed' BIR, otherwise, ProcessedBIR is set to NULL.
         Section 1.6:
         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 fail, an UNDECIDED conformity
response is issued.
      </description>

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

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

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

      <!-- 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="moduleUuid" var="_moduleUuid"/>
        <input name="moduleVersionMajor" var="_moduleVersionMajor"/>
        <input name="moduleVersionMinor" var="_moduleVersionMinor"/>
        <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="be0a1330-d59e-11d8-9669-0800200c9a66"
          function="BioSPI_ModuleEventHandler"/>
   </assertion>

   <activity name="BioSPI_Process">
     <input name="moduleUuid"/>
     <input name="moduleVersionMajor"/>
     <input name="moduleVersionMinor"/>
     <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="_modulehandle" value="1"/>

     <!-- Invoke the functions BioSPI_ModuleLoad and BioSPI_ModuleAttach
exposed by the BSP under test.
          The input value for the parameter "deviceIDOrNull" is "0", therefore
the assertion will test the BSP's default device. -->
     <invoke activity="LoadAndAttach"
          package="be0a1330-d59e-11d8-9669-0800200c9a66"
          break_on_break="true">
        <input name="moduleUuid" var="moduleUuid"/>
        <input name="moduleVersionMajor" var="moduleVersionMajor"/>
        <input name="moduleVersionMinor" var="moduleVersionMinor"/>
        <input name="deviceIDOrNull" value="0"/>
        <input name="module" var="_modulehandle"/>
        <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="ModuleHandle" var="_modulehandle"/>
        <input name="Purpose" var="__BioAPI_PURPOSE_VERIFY"/>
        <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="be0a1330-d59e-11d8-9669-0800200c9a66"
          break_on_break="true" >
        <input name="ModuleHandle" var="_modulehandle" />
        <input name="BIRHandle"  var="capturedbir_handle" />
     </invoke>

     <!-- Invoke the function BioSPI_Process.-->
     <invoke function="BioSPI_Process">
        <input name="ModuleHandle" var="_modulehandle"/>
        <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="ModuleHandle" var="_modulehandle"/>
        <input name="Handle" var="processedbir_handle"/>
        <output name="Length" setvar="length"/>
        <output name="HeaderVersion" setvar="headerversion"/>
        <output name="ProcessedLevel" setvar="processedLevel"/>
        <output name="FormatOwner" setvar="formatowner"/>
        <output name="FormatId" setvar="formatid"/>
        <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, 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_ModuleDetach and BioSPI_ModuleUnload -->
     <invoke activity="DetachAndUnload"
          package="be0a1330-d59e-11d8-9669-0800200c9a66" >
        <input name="moduleUuid" var="moduleUuid" />
        <input name="module" var="_modulehandle" />
     </invoke>
   </activity>

</package>
```

16.22.5 Assertion 15.4 BioSPI_Process_InputBIRDataType

16.22.5.1 Test Purpose:

To test the BioSPI_Process with invalid input BIR data type.

16.22.5.2 Test 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) Change the processed level of the BIR to PROCESSED.
6) Call BioSPI_Process specifying the input BIR.
7) Check if the return value is different from BioAPI_OK.
8) Detach and Unload the BSP under test.

16.22.5.3 Expected Results:
Return code BioAPI_OK.

16.22.5.4 XML:

```
<package name="dec4ca20-3a25-11d9-9669-0800200c9a66">
   <author>
      author
   </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 1.1 is quoted below from subclauses 1.6 and
2.5.3.3.
         ____________________________________________________________
         BioAPI_RETURN BioAPI BioSPI_Process
            (BioAPI_HANDLE ModuleHandle,
            const BioAPI_INPUT_BIR *CapturedBIR,
            BioAPI_BIR_HANDLE_PTR ProcessedBIR);

         This function processes the intermediate data captured via a call to
BioSPI_Capture for the purpose of either verification or identification. If the
processing capability is in the attached BSP, the BSP builds a 'processed' BIR,
otherwise, ProcessedBIR is set to NULL.
         ____________________________________________________________
         Section 2.5.3.3:
         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 in the attached BSP, the BSP builds a
'processed' BIR, otherwise, ProcessedBIR is set to NULL.
         Section 1.6:
         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) Change the processed level of the BIR to PROCESSED.
          6) Call BioSPI_Process 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 fail, an UNDECIDED conformity
response is issued.
     </description>

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

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

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

     <!-- 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="moduleUuid" var="_moduleUuid"/>
        <input name="moduleVersionMajor" var="_moduleVersionMajor"/>
        <input name="moduleVersionMinor" var="_moduleVersionMinor"/>
        <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="be0a1330-d59e-11d8-9669-0800200c9a66"
          function="BioSPI_ModuleEventHandler"/>
   </assertion>

   <activity name="BioSPI_Process_InputBIRDataType">
     <input name="moduleUuid"/>
     <input name="moduleVersionMajor"/>
     <input name="moduleVersionMinor"/>
     <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="_modulehandle" value="1"/>

     <!-- Invoke the functions BioSPI_ModuleLoad and BioSPI_ModuleAttach
exposed by the BSP under test.
          The input value for the parameter "deviceIDOrNull" is "0", therefore
the assertion will test the BSP's default device. -->
     <invoke activity="LoadAndAttach"
          package="be0a1330-d59e-11d8-9669-0800200c9a66"
          break_on_break="true">
        <input name="moduleUuid" var="moduleUuid"/>
        <input name="moduleVersionMajor" var="moduleVersionMajor"/>
        <input name="moduleVersionMinor" var="moduleVersionMinor"/>
        <input name="deviceIDOrNull" value="0"/>
        <input name="module" var="_modulehandle"/>
        <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="ModuleHandle" var="_modulehandle"/>
        <input name="Purpose" var="__BioAPI_PURPOSE_VERIFY"/>
        <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="be0a1330-d59e-11d8-9669-0800200c9a66"
           break_on_break="true" >
        <input name="ModuleHandle" var="_modulehandle" />
        <input name="BIRHandle"  var="capturedbir_handle" />
     </invoke>

     <invoke function="BioSPI_GetBIRFromHandle">
        <input name="ModuleHandle" var="_modulehandle"/>
        <input name="Handle" var="capturedbir_handle" />
        <output name="Length" setvar="length"/>
        <output name="HeaderVersion" setvar="headerversion" />
        <output name="ProcessedLevel" setvar="processedLevel" />
        <output name="Encrypted" setvar="encrypted" />
        <output name="Signed" setvar="signed" />
        <output name="FormatOwner" setvar="formatowner" />
        <output name="FormatId" setvar="formatid" />
        <output name="Quality" setvar="quality" />
        <output name="Purpose" setvar="purpose" />
        <output name="FactorMultiple" setvar="factorMultiple"/>
        <output name="FactorFacialFeatures"
              setvar="factorFacialFeatures"/>
        <output name="FactorVoice" setvar="factorVoice"/>
        <output name="FactorFingerprint"
              setvar="factorFingerprint"/>
        <output name="FactorIris" setvar="factorIris"/>
        <output name="FactorRetina" setvar="factorRetina"/>
        <output name="FactorHandGeometry"
              setvar="factorHandGeometry"/>
        <output name="FactorSignatureDynamics"
              setvar="factorSignatureDynamics"/>
        <output name="FactorKeystrokeDynamics"
              setvar="factorKeystrokeDynamics"/>
        <output name="FactorLipMovement"
              setvar="factorLipMovement"/>
        <output name="FactorThermalFaceImage"
              setvar="factorThermalFaceImage"/>
        <output name="FactorThermalHandImage"
              setvar="factorThermalHandImage"/>
        <output name="FactorGait" setvar="factorGait"/>
        <output name="FactorPassword" setvar="factorPassword"/>
        <output name="BiometricData" setvar="biomtricdata" />
        <output name="Signature" setvar="signature" />
        <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_GetBIRFromHandle has returned BioAPI_OK.
```

```
      </description>
      <equal_to var1="return" var2="__BioAPI_OK"/>
   </assert_condition>

   <!-- Invoke the function BioSPI_Process.-->
   <invoke function="BioSPI_Process">
      <input name="ModuleHandle" var="_modulehandle"/>
      <input name="CapturedBIR_Form"
            var="__BioAPI_FULLBIR_INPUT"/>
      <input name="CapturedBIR_Length" var="length"/>
      <input name="CapturedBIR_HeaderVersion"
            var="headerversion" />
      <input name="CapturedBIR_ProcessedLevel"
            var="__BioAPI_BIR_DATA_TYPE_PROCESSED" />
      <input name="CapturedBIR_Encrypted" var="encrypted" />
      <input name="CapturedBIR_Signed" var="signed" />
      <input name="CapturedBIR_FormatOwner" var="formatowner" />
      <input name="CapturedBIR_FormatId" var="formatid" />
      <input name="CapturedBIR_Quality" var="quality" />
      <input name="CapturedBIR_Purpose" var="purpose" />
      <input name="CapturedBIR_FactorMultiple"
            var="factorMultiple"/>
      <input name="CapturedBIR_FactorFacialFeatures"
            var="factorFacialFeatures"/>
      <input name="CapturedBIR_FactorVoice" var="factorVoice"/>
      <input name="CapturedBIR_FactorFingerprint"
            var="factorFingerprint"/>
      <input name="CapturedBIR_FactorIris" var="factorIris"/>
      <input name="CapturedBIR_FactorRetina" var="factorRetina"/>
      <input name="CapturedBIR_FactorHandGeometry"
            var="factorHandGeometry"/>
      <input name="CapturedBIR_FactorSignatureDynamics"
            var="factorSignatureDynamics"/>
      <input name="CapturedBIR_FactorKeystrokeDynamics"
            var="factorKeystrokeDynamics"/>
      <input name="CapturedBIR_FactorLipMovement"
            var="factorLipMovement"/>
      <input name="CapturedBIR_FactorThermalFaceImage"
            var="factorThermalFaceImage"/>
      <input name="CapturedBIR_FactorThermalHandImage"
            var="factorThermalHandImage"/>
      <input name="CapturedBIR_FactorGait" var="factorGait"/>
      <input name="CapturedBIR_FactorPassword"
            var="factorPassword"/>
      <input name="CapturedBIR_BiometricData"
            var="biomtricdata" />
      <input name="CapturedBIR_Signature" var="signature" />
      <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_ModuleDetach and BioSPI_ModuleUnload -->
```

```
      <invoke activity="DetachAndUnload"
           package="be0a1330-d59e-11d8-9669-0800200c9a66" >
         <input name="moduleUuid" var="moduleUuid" />
         <input name="module" var="_modulehandle" />
      </invoke>
   </activity>

</package>
```

16.22.6 Assertion 15.5 BioSPI_Process_OutputBIRPurpose

16.22.6.1 Test Purpose:
To test BioSPI_Process with valid parameters and check that the returned purpose of the processed BIR is the same as the purpose of the captured BIR.

16.22.6.2 Test 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.

16.22.6.3 Expected Results:
Return code BioAPI_OK.

16.22.6.4 XML:

```
<package name="015b1f48-07e1-1085-8918-0002a5d5fd2e">
   <author>
      author
   </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 1.1 is quoted below from subclause 2.1.10.
         ____________________________________________________________
         BioAPI_RETURN BioAPI BioSPI_Process
            (BioAPI_HANDLE ModuleHandle,
            const BioAPI_INPUT_BIR *CapturedBIR,
            BioAPI_BIR_HANDLE_PTR ProcessedBIR);

         This function processes the intermediate data captured via a call to
BioSPI_Capture for the purpose of either verification or identification. If the
processing capability is in the attached BSP, the BSP builds a processed BIR,
otherwise, ProcessedBIR is set to NULL.
         ____________________________________________________________
         Section 2.1.10:
         e) The Process and Create_Template functions do not have Purpose as an
input parameter, but read the Purpose field from the BIR header of the input
```

```
Captured_BIR.
        f) The Process function may accept as input any intermediate BIR with a
Purpose including Verify or Identify, and will output only BIRs with a Purpose
of Verify and/or Identify.
        ____________________________________________________________

        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 fail, an UNDECIDED conformity
response is issued.
     </description>

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

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

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

     <!-- 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="moduleUuid" var="_moduleUuid"/>
        <input name="moduleVersionMajor" var="_moduleVersionMajor"/>
        <input name="moduleVersionMinor" var="_moduleVersionMinor"/>
        <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="be0a1330-d59e-11d8-9669-0800200c9a66"
```

```
          function="BioSPI_ModuleEventHandler"/>
   </assertion>

   <activity name="BioSPI_Process">
      <input name="moduleUuid"/>
      <input name="moduleVersionMajor"/>
      <input name="moduleVersionMinor"/>
      <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="_modulehandle" value="1"/>

      <!-- Invoke the functions BioSPI_ModuleLoad and BioSPI_ModuleAttach
exposed by the BSP under test.
          The input value for the parameter "deviceIDOrNull" is "0", therefore
the assertion will test the BSP's default device. -->
      <invoke activity="LoadAndAttach"
          package="be0a1330-d59e-11d8-9669-0800200c9a66"
          break_on_break="true">
        <input name="moduleUuid" var="moduleUuid"/>
        <input name="moduleVersionMajor" var="moduleVersionMajor"/>
        <input name="moduleVersionMinor" var="moduleVersionMinor"/>
        <input name="deviceIDOrNull" value="0"/>
        <input name="module" var="_modulehandle"/>
        <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="ModuleHandle" var="_modulehandle"/>
        <input name="Purpose" var="__BioAPI_PURPOSE_VERIFY"/>
        <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="be0a1330-d59e-11d8-9669-0800200c9a66"
          break_on_break="true" >
        <input name="ModuleHandle" var="_modulehandle"/>
        <input name="BIRHandle" var="capturedbir_handle"/>
     </invoke>

     <!-- Invoke the function BioSPI_Process.-->
     <invoke function="BioSPI_Process">
        <input name="ModuleHandle" var="_modulehandle"/>
        <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="ModuleHandle" var="_modulehandle"/>
        <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="ModuleHandle" var="_modulehandle"/>
        <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_ModuleDetach and BioSPI_ModuleUnload -->
      <invoke activity="DetachAndUnload"
          package="be0a1330-d59e-11d8-9669-0800200c9a66" >
        <input name="moduleUuid" var="moduleUuid"/>
        <input name="module" var="_modulehandle"/>
      </invoke>
   </activity>

</package>
```

16.23 Feature 15a. Return of quality in the processed BIR header

16.23.1 BioAPI Specification References:
3.3.4.3, (2.1.46, 4.2.4.2)

16.23.2 Assertion 15a.1 BioSPI_Process_BIRHeaderQuality

16.23.2.1 Test Purpose:

To test the return of quality in the ProcessedBIR header if supported by the BSP.

16.23.2.2 Test Scenario:
If the BSP supports return of quality, call BioSPI_Process with valid parameters.

16.23.2.3 Expected Results:
Return code BioAPI_OK and a valid ProcessedBIR that includes the quality in its header.

16.23.2.4 XML:

```
<package name="97356030-2d0e-11d9-9669-0800200c9a66">
   <author>
      author
   </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 1.1 is quoted below from subclauses 2.2.1.2
and 4.2.4.2.
         ____________________________________________________________
         BioAPI_RETURN BioAPI BioSPI_Process
            (BioAPI_HANDLE ModuleHandle,
            const BioAPI_INPUT_BIR *CapturedBIR,
            BioAPI_BIR_HANDLE_PTR ProcessedBIR);

         This function processes the intermediate data captured via a call to
BioSPI_Capture for the purpose of either verification or identification. If the
processing capability is in the attached BSP, the BSP builds a processed BIR,
otherwise, ProcessedBIR is set to NULL.

         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.
         ____________________________________________________________
         Section 2.2.1.2:
         #define BioAPI_QUALITY_PROCESSED (0x00000008)
         If set, BSP supports the return of quality value (in the BIR header) for
processed biometric data.
         Section 4.2.4.2:
         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 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 fail, an UNDECIDED conformity
response is issued.
     </description>
     <!-- UUID of the BSP under test -->
     <input name="_moduleUuid"/>
     <!-- Major version number of the BSP under test -->
     <input name="_moduleVersionMajor"/>
     <!-- Minor version number of the BSP under test -->
     <input name="_moduleVersionMinor"/>
     <!-- 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="moduleUuid" var="_moduleUuid"/>
        <input name="moduleVersionMajor" var="_moduleVersionMajor"/>
        <input name="moduleVersionMinor" var="_moduleVersionMinor"/>
        <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="be0a1330-d59e-11d8-9669-
0800200c9a66" function="BioSPI_ModuleEventHandler"/>
   </assertion>
   <activity name="BioSPI_Process">
     <input name="moduleUuid"/>
     <input name="moduleVersionMajor"/>
     <input name="moduleVersionMinor"/>
     <input name="inserttimeouttime"/>
     <input name="nosourcepresentsupported"/>
     <input name="sourcepresenttimeouttime"/>
     <input name="capturetimeouttime"/>
     <input name="processedQualitySupported"/>
     <!-- This assertion will use ModuleHandle "1" for all BioSPI calls that
require it -->
     <set name="_modulehandle" value="1"/>
     <!-- Invoke the functions BioSPI_ModuleLoad and BioSPI_ModuleAttach
exposed by the BSP under test.
          The input value for the parameter "deviceIDOrNull" is "0", therefore
the assertion will test the BSP's default device. -->
```

```
     <invoke activity="LoadAndAttach" package="be0a1330-d59e-11d8-9669-
0800200c9a66" break_on_break="true">
        <input name="moduleUuid" var="moduleUuid"/>
        <input name="moduleVersionMajor" var="moduleVersionMajor"/>
        <input name="moduleVersionMinor" var="moduleVersionMinor"/>
        <input name="deviceIDOrNull" value="0"/>
        <input name="module" var="_modulehandle"/>
        <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="ModuleHandle" var="_modulehandle"/>
        <input name="Purpose" var="__BioAPI_PURPOSE_VERIFY"/>
        <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="be0a1330-d59e-11d8-
9669-0800200c9a66" break_on_break="true">
        <input name="ModuleHandle" var="_modulehandle"/>
        <input name="BIRHandle" var="capturedbir_handle"/>
     </invoke>
     <!-- Invoke the function BioSPI_Process. -->
     <invoke function="BioSPI_Process">
        <input name="ModuleHandle" var="_modulehandle"/>
        <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="ModuleHandle" var="_modulehandle"/>
        <input name="Handle" var="processedbir_handle"/>
        <output name="Length" setvar="length"/>
        <output name="HeaderVersion" setvar="headerversion"/>
        <output name="ProcessedLevel" setvar="processedLevel"/>
        <output name="FormatOwner" setvar="formatowner"/>
        <output name="FormatId" setvar="formatid"/>
        <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, 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="be0a1330-d59e-11d8-
9669-0800200c9a66" 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="be0a1330-d59e-
11d8-9669-0800200c9a66" break_on_break="true">
        <only_if>
           <same_as var1="processedQualitySupported" value2="false"/>
        </only_if>
        <input name="quality" var="quality"/>
      </invoke>
      <!-- Invoke the functions BioSPI_ModuleDetach and BioSPI_ModuleUnload -->
      <invoke activity="DetachAndUnload" package="be0a1330-d59e-11d8-9669-
0800200c9a66">
        <input name="moduleUuid" var="moduleUuid"/>
        <input name="module" var="_modulehandle"/>
      </invoke>
   </activity>
</package>
```

16.24 Feature 15b. BIR signing (by BSP)

16.24.1 BioAPI Specification References:
1.5, 2.1.7

16.24.2 Assertion 15b.1 BioSPI_Process_BIRSigned

16.24.2.1 Test Purpose:
To test BIR signing if supported by the BSP.

16.24.2.2 Test Scenario:
If the BSP supports signing cal BioSPI_Process with valid parameters.

16.24.2.3 Expected Results:
Return code BioAPI_OK and a valid ProcessedBIR that includes a signature.

16.24.2.4 XML:

16.25 Feature 15c. BIR encryption (by BSP)

16.25.1 BioAPI Specification References:
1.5, 2.1.7

16.25.2 Assertion 15c.1 BioSPI_Process_BIREncrypted

16.25.2.1 Test Purpose:
To test BIR encryption if supported by the BSP.

16.25.2.2 Test Scenario:
If the BSP supports BIR encryption call BioSPI_Process with valid parameters.

16.25.2.3 Expected Results:
Return code BioAPI_OK and a valid ProcessedBIR.

16.25.2.4 XML:

16.26 Feature 16. BioSPI Verify Match

16.26.1 BioAPI Specification References:
3.3.4.4

16.26.2 Assertion 16.1 BioSPI_VerifyMatch_ValidParam

16.26.2.1 Test Purpose:
To test BioSPI_VerifyMatch with valid parameters.

16.26.2.2 Test Scenario:
Valid parameters are passed to BioSPI_VerifyMatch.

16.26.2.3 Expected Results:
Return code BioAPI_OK. Result can be either BioAPI_TRUE or BioAPI_FALSE. FARAchieved must be valid.

16.26.2.4 XML:

```
<package name="60f19fb0-dcec-11d8-9669-0800200c9a66">
   <author>
      author
   </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 1.1 is quoted below from subclause and
2.5.3.4.
         ____________________________________________________________
         BioAPI_RETURN BioAPI BioSPI_VerifyMatch
            (BioAPI_HANDLE ModuleHandle,
            const BioAPI_FAR *MaxFARRequested,
            const BioAPI_FRR *MaxFRRRequested,
            const BioAPI_BOOL *FARPrecedence,
            const BioAPI_INPUT_BIR *ProcessedBIR,
            const BioAPI_INPUT_BIR *StoredTemplate,
            BioAPI_BIR_HANDLE *AdaptedBIR,
            BioAPI_BOOL *Result,
            BioAPI_FAR_PTR FARAchieved,
            BioAPI_FRR_PTR FRRAchieved,
            BioAPI_DATA_PTR *Payload);

         This function performs a verification (1-to-1) match between two BIRs;
the ProcessedBIR and the StoredTemplate.

         The Boolean Result indicates whether verification was successful or not,
and the FARAchieved is a FAR value indicating how closely the BIRs actually
matched.
         ____________________________________________________________
         Section 2.5.3.4:
         This function performs a verification (1-to-1) match between two BIRs;
the ProcessedBIR and the 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 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 fail, an UNDECIDED conformity
response is issued.
```

```
     </description>

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

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

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

     <!-- 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"/>

     <!-- MaxFARRequested for BioSPI_VerifyMatch -->
     <input name="_maxFARRequested" />

     <!-- MaxFRRRequested for BioSPI_VerifyMatch -->
     <input name="_maxFRRRequested" />

     <!-- MaxFARPrecedence for BioSPI_VerifyMatch -->
     <input name="_farprecedence" />

     <!-- 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="moduleUuid" var="_moduleUuid"/>
        <input name="moduleVersionMajor" var="_moduleVersionMajor"/>
        <input name="moduleVersionMinor" var="_moduleVersionMinor"/>
        <input name="inserttimeouttime" var="_inserttimeout"/>
        <input name="nosourcepresentsupported"
              var="_noSourcePresentSupported" />
        <input name="sourcepresenttimeouttime"
              var="_sourcepresenttimeout"/>
        <input name="capturetimeouttime" var="_capturetimeout"/>
        <input name="maxFARRequested" var="_maxFARRequested"/>
        <input name="maxFRRRequested" var="_maxFRRRequested"/>
        <input name="farprecedence" var="_farprecedence" />
     </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="be0a1330-d59e-11d8-9669-0800200c9a66"
           function="BioSPI_ModuleEventHandler"/>
   </assertion>

   <activity name="BioSPI_VerifyMatch_ValidParam">
     <input name="moduleUuid"/>
     <input name="moduleVersionMajor"/>
     <input name="moduleVersionMinor"/>
     <input name="inserttimeouttime"/>
```

```
     <input name="nosourcepresentsupported" />
     <input name="sourcepresenttimeouttime"/>
     <input name="capturetimeouttime"/>
     <input name="maxFARRequested" />
     <input name="maxFRRRequested" />
     <input name="farprecedence" />

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

     <!-- Invoke the functions BioSPI_ModuleLoad and BioSPI_ModuleAttach
exposed by the BSP under test.
          The input value for the parameter "deviceIDOrNull" is "0", therefore
the assertion will test the BSP's default device. -->
     <invoke activity="LoadAndAttach"
          package="be0a1330-d59e-11d8-9669-0800200c9a66"
          break_on_break="true">
        <input name="moduleUuid" var="moduleUuid"/>
        <input name="moduleVersionMajor" var="moduleVersionMajor"/>
        <input name="moduleVersionMinor" var="moduleVersionMinor"/>
        <input name="deviceIDOrNull" value="0"/>
        <input name="module" var="_modulehandle"/>
        <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="ModuleHandle" var="_modulehandle"/>
        <input name="Purpose"
             var="__BioAPI_PURPOSE_ENROLL_FOR_VERIFICATION_ONLY"/>
        <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="ModuleHandle" var="_modulehandle"/>
        <input name="Purpose" var="__BioAPI_PURPOSE_VERIFY"/>
        <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="ModuleHandle" var="_modulehandle"/>
        <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="be0a1330-d59e-11d8-9669-0800200c9a66"
                break_on_break="true" >
        <only_if>
           <not_equal_to var1="processedLevel"
                var2="__BioAPI_BIR_DATA_TYPE_PROCESSED" />
```

```
        </only_if>
        <input name="ModuleHandle" var="_modulehandle"/>
        <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="ModuleHandle" var="_modulehandle" />
        <input name="MaxFARRequested" var="maxFARRequested" />
        <input name="MaxFRRRequested" var="maxFRRRequested" />
        <input name="FARPrecedence"  var="farprecedence" />
        <input name="ProcessedBIR_Form"
             var="__BioAPI_BIR_HANDLE_INPUT" />
        <input name="ProcessedBIR_BIRHandle"
             var="_processedbir_handle" />
        <input name="StoredTemplate_Form"
             var="__BioAPI_BIR_HANDLE_INPUT" />
        <input name="StoredTemplate_BIRHandle"
             var="template_handle" />
        <input name="no_AdaptedBIR" value="true" />
        <input name="no_Payload" value="true" />
        <output name="Result" setvar="result" />
        <output name="FARAchieved" setvar="farAchieved" />
        <output name="FRRAchieved" setvar="frrAchieved" />
        <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_ModuleDetach and BioSPI_ModuleUnload -->
     <invoke activity="DetachAndUnload"
          package="be0a1330-d59e-11d8-9669-0800200c9a66" >
        <input name="moduleUuid" var="moduleUuid" />
        <input name="module" var="_modulehandle" />
     </invoke>
   </activity>

</package>
```

16.26.3 Assertion 16.2 BioSPI_VerifyMatch_UnspecifiedFAR

16.26.3.1 Test Purpose:
To test failure on unspecified FAR.

16.26.3.2 Test Scenario:
BioSPI_VerifyMatch is passed valid parameters except for MaxFARRequested.

16.26.3.3 Expected Results:
Return code other than BioAPI_OK.

16.26.3.4 XML:

```
<package name="229f7f90-de39-11d8-9669-0800200c9a66">
   <author>
      author
   </author>

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

   <assertion name="BioSPI_VerifyMatch_UnspecifiedFAR" model="BSPTesting">
      <description>
         This assertion invokes the function BioSPI_VerifyMatch with
MaxFARRequested equal to NULL.  The function is expected to return BioAPI_OK.
         The relevant text in BioAPI 1.1 is quoted below from subclause 2.5.3.4.
         ____________________________________________________________
         BioAPI_RETURN BioAPI BioSPI_VerifyMatch
            (BioAPI_HANDLE ModuleHandle,
            const BioAPI_FAR *MaxFARRequested,
            const BioAPI_FRR *MaxFRRRequested,
            const BioAPI_BOOL *FARPrecedence,
            const BioAPI_INPUT_BIR *ProcessedBIR,
            const BioAPI_INPUT_BIR *StoredTemplate,
            BioAPI_BIR_HANDLE *AdaptedBIR,
            BioAPI_BOOL *Result,
            BioAPI_FAR_PTR FARAchieved,
            BioAPI_FRR_PTR FRRAchieved,
            BioAPI_DATA_PTR *Payload);

         This function performs a verification (1-to-1) match between two BIRs;
the ProcessedBIR and the StoredTemplate.

         The application must request a maximum FAR value for a successful match,
and may also (optionally) request a maximum FRR for a successful match.
         ____________________________________________________________
         Section 2.5.3.4:
         The application must request a maximum FAR value for a successful match,
and may also (optionally) request a maximum FRR for a successful match.
         ____________________________________________________________

         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 with FAR set to NULL.
            7) Check the return code, which is expected to be an error code.

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

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

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

      <!-- Minor version number of the BSP under test -->
```

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

     <!-- 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"/>

     <!-- MaxFRRRequested for BioSPI_VerifyMatch -->
     <input name="_maxFRRRequested"/>

     <!-- MaxFARPrecedence for BioSPI_VerifyMatch -->
     <input name="_farprecedence"/>

     <!-- Invocation of the primary activity of this assertion with input
parameter values assigned from the assertion's parameters. -->
     <invoke activity="BioSPI_VerifyMatch">
        <input name="moduleUuid" var="_moduleUuid"/>
        <input name="moduleVersionMajor" var="_moduleVersionMajor"/>
        <input name="moduleVersionMinor" var="_moduleVersionMinor"/>
        <input name="inserttimeouttime" var="_inserttimeout"/>
        <input name="nosourcepresentsupported"
             var="_noSourcePresentSupported"/>
        <input name="sourcepresenttimeouttime"
             var="_sourcepresenttimeout"/>
        <input name="capturetimeouttime" var="_capturetimeout"/>
        <input name="maxFRRRequested" var="_maxFRRRequested"/>
        <input name="farprecedence" var="_farprecedence"/>
     </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="be0a1330-d59e-11d8-9669-0800200c9a66"
          function="BioSPI_ModuleEventHandler"/>
   </assertion>

   <activity name="BioSPI_VerifyMatch">
     <input name="moduleUuid"/>
     <input name="moduleVersionMajor"/>
     <input name="moduleVersionMinor"/>
     <input name="inserttimeouttime"/>
     <input name="nosourcepresentsupported"/>
     <input name="sourcepresenttimeouttime"/>
     <input name="capturetimeouttime"/>
     <input name="maxFRRRequested"/>
     <input name="farprecedence"/>

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

     <!-- Invoke the functions BioSPI_ModuleLoad and BioSPI_ModuleAttach
exposed by the BSP under test.
          The input value for the parameter "deviceIDOrNull" is "0", therefore
```

```
the assertion will test the BSP's default device. -->
     <invoke activity="LoadAndAttach"
          package="be0a1330-d59e-11d8-9669-0800200c9a66"
          break_on_break="true">
        <input name="moduleUuid" var="moduleUuid"/>
        <input name="moduleVersionMajor" var="moduleVersionMajor"/>
        <input name="moduleVersionMinor" var="moduleVersionMinor"/>
        <input name="deviceIDOrNull" value="0"/>
        <input name="module" var="_modulehandle"/>
        <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="ModuleHandle" var="_modulehandle"/>
        <input name="Purpose"
             var="__BioAPI_PURPOSE_ENROLL_FOR_VERIFICATION_ONLY"/>
        <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="ModuleHandle" var="_modulehandle"/>
        <input name="Purpose" var="__BioAPI_PURPOSE_VERIFY"/>
        <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="ModuleHandle" var="_modulehandle"/>
        <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="be0a1330-d59e-11d8-9669-0800200c9a66"
          break_on_break="true" >
        <only_if>
          <not_equal_to var1="processedLevel"
               var2="__BioAPI_BIR_DATA_TYPE_PROCESSED"/>
        </only_if>
        <input name="ModuleHandle" var="_modulehandle"/>
        <input name="CapturedBIR_BIRHandle"
             var="capturedbir_handle"/>
     </invoke>

     <!-- Invoke the function BioSPI_VerifyMatch to verify the processed BIR
against the stored template -->
     <invoke function="BioSPI_VerifyMatch" >
        <input name="ModuleHandle" var="_modulehandle"/>
        <input name="FARPrecedence" var="farprecedence"/>
        <input name="ProcessedBIR_Form"
             var="__BioAPI_BIR_HANDLE_INPUT"/>
        <input name="ProcessedBIR_BIRHandle"
```

```
            var="_processedbir_handle"/>
        <input name="StoredTemplate_Form"
            var="__BioAPI_BIR_HANDLE_INPUT"/>
        <input name="StoredTemplate_BIRHandle"
            var="template_handle"/>
        <input name="no_AdaptedBIR" value="true"/>
        <input name="no_Payload" value="true"/>
        <output name="Result" setvar="result"/>
        <output name="FARAchieved" setvar="farAchieved"/>
        <output name="FRRAchieved" setvar="frrAchieved"/>
        <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_BSP_INVALID_INPUT_POINTER.
        </description>
        <equal_to var1="return"
            var2="__BioAPIERR_BSP_INVALID_INPUT_POINTER"/>
     </assert_condition>

     <!-- Invoke the functions BioSPI_ModuleDetach and BioSPI_ModuleUnload -->
     <invoke activity="DetachAndUnload"
          package="be0a1330-d59e-11d8-9669-0800200c9a66" >
        <input name="moduleUuid" var="moduleUuid"/>
        <input name="module" var="_modulehandle"/>
     </invoke>
   </activity>

</package>
```

16.26.4 Assertion 16.3 BioSPI_VerifyMatch_Inconsistent_Purpose

16.26.4.1 Test Purpose:
To test BioSPI_VerifyMatch

16.26.4.2 Test 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.

16.26.4.3 Expected Results:
Return code BioAPIERR_BSP_INCONSISTENT_PURPOSE.

16.26.4.4 XML:

```
<package name="9108ec70-2e9b-11d9-9669-0800200c9a66">
   <author>
      author
```

```
   </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 1.1 is quoted below from subclauses 2.3.4.2
and 2.5.3.4.
         ____________________________________________________________
         BioAPI_RETURN BioAPI BioSPI_VerifyMatch
            (BioAPI_HANDLE ModuleHandle,
            const BioAPI_FAR *MaxFARRequested,
            const BioAPI_FRR *MaxFRRRequested,
            const BioAPI_BOOL *FARPrecedence,
            const BioAPI_INPUT_BIR *ProcessedBIR,
            const BioAPI_INPUT_BIR *StoredTemplate,
            BioAPI_BIR_HANDLE *AdaptedBIR,
            BioAPI_BOOL *Result,
            BioAPI_FAR_PTR FARAchieved,
            BioAPI_FRR_PTR FRRAchieved,
            BioAPI_DATA_PTR *Payload);

         This function performs a verification (1-to-1) match between two BIRs;
the ProcessedBIR and the StoredTemplate.
         ____________________________________________________________
         Section 2.3.4.2:
         BSP Specific Error values: #define BioAPIERR_BSP_INCONSISTENT_PURPOSE
(BioAPI_BSP_BASE_BSP_ERROR+13)
         The purpose recorded in the BIR, and the requested purpose, are
inconsistent with the function being performed.
         Section 2.5.3.4:
         Errors: BioAPIERR_BSP_INCONSISTENT_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_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 fail, an UNDECIDED conformity
response is issued.
      </description>
      <!-- UUID of the BSP under test -->
      <input name="_moduleUuid"/>
      <!-- Major version number of the BSP under test -->
      <input name="_moduleVersionMajor"/>
      <!-- Minor version number of the BSP under test -->
      <input name="_moduleVersionMinor"/>
      <!-- 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="_maxFARRequested"/>
     <!-- MaxFRRRequested for BioSPI_VerifyMatch -->
     <input name="_maxFRRRequested"/>
     <!-- MaxFARPrecedence for BioSPI_VerifyMatch -->
     <input name="_farprecedence"/>
     <!-- Purpose of the BIR -->
     <input name="_purpose"/>
     <!-- Invocation of the primary activity of this assertion with input
parameter values assigned from the assertion's parameters. -->
     <invoke activity="BioSPI_VerifyMatch">
        <input name="moduleUuid" var="_moduleUuid"/>
        <input name="moduleVersionMajor" var="_moduleVersionMajor"/>
        <input name="moduleVersionMinor" var="_moduleVersionMinor"/>
        <input name="inserttimeouttime" var="_inserttimeout"/>
        <input name="nosourcepresentsupported" var="_noSourcePresentSupported"/>
        <input name="sourcepresenttimeouttime" var="_sourcepresenttimeout"/>
        <input name="capturetimeouttime" var="_capturetimeout"/>
        <input name="maxFARRequested" var="_maxFARRequested"/>
        <input name="maxFRRRequested" var="_maxFRRRequested"/>
        <input name="farprecedence" var="_farprecedence"/>
        <input name="purpose" var="_purpose"/>
     </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="be0a1330-d59e-11d8-9669-
0800200c9a66" function="BioSPI_ModuleEventHandler"/>
   </assertion>
   <activity name="BioSPI_VerifyMatch">
     <input name="moduleUuid"/>
     <input name="moduleVersionMajor"/>
     <input name="moduleVersionMinor"/>
     <input name="inserttimeouttime"/>
     <input name="nosourcepresentsupported"/>
     <input name="sourcepresenttimeouttime"/>
     <input name="capturetimeouttime"/>
     <input name="maxFARRequested"/>
     <input name="maxFRRRequested"/>
     <input name="farprecedence"/>
     <input name="purpose"/>
     <!-- This assertion will use ModuleHandle "1" for all BioSPI calls that
require it -->
     <set name="_modulehandle" value="1"/>
     <!-- Invoke the functions BioSPI_ModuleLoad and BioSPI_ModuleAttach
exposed by the BSP under test.
          The input value for the parameter "deviceIDOrNull" is "0", therefore
the assertion will test the BSP's default device. -->
     <invoke activity="LoadAndAttach" package="be0a1330-d59e-11d8-9669-
0800200c9a66" break_on_break="true">
        <input name="moduleUuid" var="moduleUuid"/>
        <input name="moduleVersionMajor" var="moduleVersionMajor"/>
        <input name="moduleVersionMinor" var="moduleVersionMinor"/>
        <input name="deviceIDOrNull" value="0"/>
        <input name="module" var="_modulehandle"/>
        <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="ModuleHandle" var="_modulehandle"/>
         <input name="Purpose"
var="__BioAPI_PURPOSE_ENROLL_FOR_VERIFICATION_ONLY"/>
         <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="ModuleHandle" var="_modulehandle"/>
         <input name="Purpose"
var="__BioAPI_PURPOSE_ENROLL_FOR_VERIFICATION_ONLY"/>
         <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="ModuleHandle" var="_modulehandle"/>
        <input name="MaxFARRequested" var="maxFARRequested"/>
        <input name="MaxFRRRequested" var="maxFRRRequested"/>
        <input name="FARPrecedence" var="farprecedence"/>
        <input name="ProcessedBIR_Form" var="__BioAPI_BIR_HANDLE_INPUT"/>
        <input name="ProcessedBIR_BIRHandle" var="template_handle2"/>
        <input name="StoredTemplate_Form" var="__BioAPI_BIR_HANDLE_INPUT"/>
        <input name="StoredTemplate_BIRHandle" var="template_handle1"/>
        <input name="no_AdaptedBIR" value="true"/>
        <input name="no_Payload" value="true"/>
        <output name="Result" setvar="result"/>
        <output name="FARAchieved" setvar="farAchieved"/>
        <output name="FRRAchieved" setvar="frrAchieved"/>
        <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_BSP_INCONSISTENT_PURPOSE
        </description>
        <equal_to var1="return" var2="__BioAPIERR_BSP_INCONSISTENT_PURPOSE"/>
     </assert_condition>
     <!-- Invoke the functions BioSPI_ModuleDetach and BioSPI_ModuleUnload -->
     <invoke activity="DetachAndUnload" package="be0a1330-d59e-11d8-9669-
0800200c9a66">
        <input name="moduleUuid" var="moduleUuid"/>
        <input name="module" var="_modulehandle"/>
     </invoke>
   </activity>
</package>
```

16.27 Feature 16b. Model/template adaptation

16.27.1 BioAPI Specification References:
3.3.4.4

16.27.2 Assertion 16b.1 BioSPI_VerifyMatch_Adaptation

16.27.2.1 Test Purpose:
To test Model/template adaptation if supported by the BSP.

16.27.2.2 Test Scenario:
If the BSP supports Model/template adaptation, call BioSPI_VerifyMatch with valid parameters.

16.27.2.3 Expected Results:
Return code BioAPI_OK and a valid AdaptedBIR.

16.27.2.4 XML:

```
<package name="91bee710-2e7c-11d9-9669-0800200c9a66">
   <author>
      author
   </author>
```

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

   <assertion name="BioSPI_VerifyMatch_Adaptation" model="BSPTesting">
      <description>
         This assertion checks if calling BioSPI_VerifyMatch with adaptation
returns BioAPI_OK.
         The relevant text in BioAPI 1.1 is quoted below from subclauses 2.2.1.2,
4.2.4.2 and 2.5.3.4.
         ____________________________________________________________
         BioAPI_RETURN BioAPI BioSPI_VerifyMatch
            (BioAPI_HANDLE ModuleHandle,
            const BioAPI_FAR *MaxFARRequested,
            const BioAPI_FRR *MaxFRRRequested,
            const BioAPI_BOOL *FARPrecedence,
            const BioAPI_INPUT_BIR *ProcessedBIR,
            const BioAPI_INPUT_BIR *StoredTemplate,
            BioAPI_BIR_HANDLE *AdaptedBIR,
            BioAPI_BOOL *Result,
            BioAPI_FAR_PTR FARAchieved,
            BioAPI_FRR_PTR FRRAchieved,
            BioAPI_DATA_PTR *Payload);

         This function performs a verification (1-to-1) match between two BIRs;
the ProcessedBIR and the StoredTemplate.

         By setting the AdaptedBIR pointer to a non-NULL value, the application
can request that a BIR be constructed by adapting the StoredTemplate using the
ProcessedBIR. A new handle is returned to the AdaptedBIR.
         ____________________________________________________________
         Section 2.5.3.4:
         By setting the AdaptedBIR pointer to non-NULL, the application can
request that a BIR be constructed by adapting the StoredTemplate using the
ProcessedBIR.
         A new handle is returned to the AdaptedBIR.
         Parameters: AdaptedBIR (output/optional) - a pointer to the handle of
the adapted BIR.
         This parameter can be NULL if an Adapted BIR is not desired.
         Not all BSPs support the adaptation of BIRs.
         The function may return a handle value of BioAPI_UNSUPPORTED_BIR_HANDLE
to indicate that adaptation is not supported or a value of
BioAPI_INVALID_BIR_HANDLE to indicate that adaptation was not possible.
         Section 2.2.1.2:
         #define BioAPI_ADAPTATION (0x00020000)
         If set, BSP supports BIR adaptation (return of Verify or VerifyMatch
operation).
         Section 4.2.4.2:
         The BSP makes the decision as to when and if the adaptation should be
performed (based on such factors as quality, elapsed time, and significant
differences).
         If the BSP does not support adaptation, the returned AdaptedBIR will be
set to -2.
         If the BSP supports adaptation, but for some reason is not able or
chooses not to perform the adaptation, then the return AdaptedBIR is set to -1.
         The BSP must post to the module registry its support for adaptation.
         ____________________________________________________________

         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 with AdaptedBIR set to a non-NULL value.
          7) Check the return code based on whether the BSP support adaptation
or not.
          8) Detach and unload the BSP under test.

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

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

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

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

     <!-- 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="_maxFARRequested" />

     <!-- MaxFRRRequested for BioSPI_VerifyMatch -->
     <input name="_maxFRRRequested" />

     <!-- MaxFARPrecedence for BioSPI_VerifyMatch -->
     <input name="_farprecedence" />

     <!-- Indicates whether the BSP claims support for template adaptation.-->
     <input name="_supportAdaptation" />

     <!-- Invocation of the primary activity of this assertion with input
parameter values assigned from the assertion's parameters. -->
     <invoke activity="BioSPI_VerifyMatch">
        <input name="moduleUuid" var="_moduleUuid"/>
        <input name="moduleVersionMajor" var="_moduleVersionMajor"/>
        <input name="moduleVersionMinor" var="_moduleVersionMinor"/>
        <input name="inserttimeouttime" var="_inserttimeout"/>
        <input name="nosourcepresentsupported"
             var="_noSourcePresentSupported" />
        <input name="sourcepresenttimeouttime"
             var="_sourcepresenttimeout"/>
        <input name="capturetimeouttime" var="_capturetimeout"/>
        <input name="maxFARRequested" var="_maxFARRequested"/>
        <input name="maxFRRRequested" var="_maxFRRRequested"/>
        <input name="farprecedence" var="_farprecedence" />
```

```
        <input name="supportAdaptation" var="_supportAdaptation" />
     </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="be0a1330-d59e-11d8-9669-0800200c9a66"
          function="BioSPI_ModuleEventHandler"/>
   </assertion>

   <activity name="BioSPI_VerifyMatch">
     <input name="moduleUuid"/>
     <input name="moduleVersionMajor"/>
     <input name="moduleVersionMinor"/>
     <input name="inserttimeouttime"/>
     <input name="nosourcepresentsupported" />
     <input name="sourcepresenttimeouttime"/>
     <input name="capturetimeouttime"/>
     <input name="maxFARRequested" />
     <input name="maxFRRRequested" />
     <input name="farprecedence" />
     <input name="supportAdaptation" />

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

     <!-- Invoke the functions BioSPI_ModuleLoad and BioSPI_ModuleAttach
exposed by the BSP under test.
          The input value for the parameter "deviceIDOrNull" is "0", therefore
the assertion will test the BSP's default device. -->
     <invoke activity="LoadAndAttach"
          package="be0a1330-d59e-11d8-9669-0800200c9a66"
          break_on_break="true">
        <input name="moduleUuid" var="moduleUuid"/>
        <input name="moduleVersionMajor" var="moduleVersionMajor"/>
        <input name="moduleVersionMinor" var="moduleVersionMinor"/>
        <input name="deviceIDOrNull" value="0"/>
        <input name="module" var="_modulehandle"/>
        <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="ModuleHandle" var="_modulehandle"/>
        <input name="Purpose"
              var="__BioAPI_PURPOSE_ENROLL_FOR_VERIFICATION_ONLY"/>
        <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="ModuleHandle" var="_modulehandle"/>
        <input name="Purpose" var="__BioAPI_PURPOSE_VERIFY"/>
        <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="ModuleHandle" var="_modulehandle"/>
        <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="be0a1330-d59e-11d8-9669-0800200c9a66"
           break_on_break="true" >
        <only_if>
          <not_equal_to var1="processedLevel"
                var2="__BioAPI_BIR_DATA_TYPE_PROCESSED" />
        </only_if>
        <input name="ModuleHandle" var="_modulehandle"/>
        <input name="CapturedBIR_BIRHandle" var="capturedbir_handle"/>
      </invoke>

      <!-- Invoke the function BioSPI_VerifyMatch to verify the processed BIR
against the stored template -->
      <invoke function="BioSPI_VerifyMatch" >
        <input name="ModuleHandle" var="_modulehandle" />
        <input name="MaxFARRequested" var="maxFARRequested" />
        <input name="MaxFRRRequested" var="maxFRRRequested" />
        <input name="FARPrecedence"  var="farprecedence" />
        <input name="ProcessedBIR_Form"
             var="__BioAPI_BIR_HANDLE_INPUT" />
        <input name="ProcessedBIR_BIRHandle"
             var="_processedbir_handle" />
        <input name="StoredTemplate_Form"
             var="__BioAPI_BIR_HANDLE_INPUT" />
        <input name="StoredTemplate_BIRHandle"
             var="template_handle" />
        <input name="no_Payload" value="true" />
        <output name="AdaptedBIR" setvar="adaptedbir_handle" />
        <output name="Result" setvar="result" />
        <output name="FARAchieved" setvar="farAchieved" />
        <output name="FRRAchieved" setvar="frrAchieved" />
        <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_VerifyMatch has returned BioAPI_OK.
        </description>
        <equal_to var1="return" var2="__BioAPI_OK"/>
      </assert_condition>

      <invoke activity="check_adaptation_supported" package="be0a1330-d59e-11d8-
9669-0800200c9a66" break_on_break="true">
        <only_if>
          <same_as var1="supportAdaptation" value2="true"/>
        </only_if>
        <input name="adaptedbir_handle" var="adaptedbir_handle"/>
```

```
      </invoke>

      <invoke activity="check_adaptation_not_supported" package="be0a1330-d59e-
11d8-9669-0800200c9a66" break_on_break="true" >
         <only_if>
            <same_as var1="supportAdaptation" value2="false"/>
         </only_if>
         <input name="adaptedbir_handle" var="adaptedbir_handle"/>
      </invoke>

      <!-- Invoke the functions BioSPI_ModuleDetach and BioSPI_ModuleUnload -->
      <invoke activity="DetachAndUnload"
            package="be0a1330-d59e-11d8-9669-0800200c9a66" >
         <input name="moduleUuid" var="moduleUuid" />
         <input name="module" var="_modulehandle" />
      </invoke>
   </activity>

</package>
```

16.28 Feature 16d. Return of achieved FRR score

16.28.1 BioAPI Specification References:
3.3.4.4

16.28.2 Assertion 16d.1 BioSPI_VerifyMatch_AchievedFRR

16.28.2.1 Test Purpose:
To test return of achieved FRR score if supported by the BSP.

16.28.2.2 Test Scenario:
If the BSP supports the return of achieved FRR score, call BioSPI_VerifyMatch with valid parameters. ProcessedBIR and Stored template should be a "match."

16.28.2.3 Expected Results:
Return code BioAPI_OK and a valid FRRAchieved.

16.28.2.4 XML:

```
<package name="6afc90e0-2e87-11d9-9669-0800200c9a66">
   <author>
      author
   </author>

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

   <assertion name="BioSPI_VerifyMatch_AchievedFRR" model="BSPTesting">
      <description>
         This assertion checks the function BioSPI_VerifyMatch called using FRR
as matching criteria.
         The relevant text in BioAPI 1.1 is quoted below from subclauses 2.2.1.2
and 2.5.3.4.
         ____________________________________________________________
         BioAPI_RETURN BioAPI BioSPI_VerifyMatch
            (BioAPI_HANDLE ModuleHandle,
```

```
          const BioAPI_FAR *MaxFARRequested,
          const BioAPI_FRR *MaxFRRRequested,
          const BioAPI_BOOL *FARPrecedence,
          const BioAPI_INPUT_BIR *ProcessedBIR,
          const BioAPI_INPUT_BIR *StoredTemplate,
          BioAPI_BIR_HANDLE *AdaptedBIR,
          BioAPI_BOOL *Result,
          BioAPI_FAR_PTR FARAchieved,
          BioAPI_FRR_PTR FRRAchieved,
          BioAPI_DATA_PTR *Payload);
```

This function performs a verification (1-to-1) match between two BIRs; the ProcessedBIR and the StoredTemplate.

The application must request a maximum FAR value for a successful match, and may also (optionally) request a maximum FRR for a successful match. If a maximum FRR value is provided, the application must also indicate via the FARPrecedence parameter, which one takes precedence. The Boolean Result indicates whether verification was successful or not, and the FARAchieved is a FAR value indicating how closely the BIRs actually matched. The BSP implementation may optionally return the corresponding FRR that was achieved through the FRRAchieved return parameter.

Section 2.5.3.4:

If a maximum FRR value is provided, the application must also indicate via the FARPrecedence parameter, which one takes precedence.

The Boolean Result indicates whether verification was successful or not, and the FARAchieved is a FAR value indicating how closely the BIRs actually matched.

The BSP implementation may optionally return the corresponding FRR that was achieved through the FRRAchieved return parameter.

Section 2.2.1.2:

#define BioAPI_FRR (0x00010000)

If set, indicates BSP supports the return of actual FRR during matching operations (Verify, VerifyMatch, Identify, IdentifyMatch).

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 with the specified FRR. Check the output parameter 'FRRAchieved'.
7) Detach and unload the BSP under test.

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

```
     </description>

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

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

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

```
     <!-- 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"/>

     <!-- MaxFRRRequested for BioSPI_VerifyMatch -->
     <input name="_maxFRRRequested" />

     <!-- Indicates whether the BSP under test claims support for FRR -->
     <input name="_supportFRR" />

     <!-- Invocation of the primary activity of this assertion with input
parameter values assigned from the assertion's parameters. -->
     <invoke activity="BioSPI_VerifyMatch">
        <input name="moduleUuid" var="_moduleUuid"/>
        <input name="moduleVersionMajor" var="_moduleVersionMajor"/>
        <input name="moduleVersionMinor" var="_moduleVersionMinor"/>
        <input name="inserttimeouttime" var="_inserttimeout"/>
        <input name="nosourcepresentsupported"
             var="_noSourcePresentSupported" />
        <input name="sourcepresenttimeouttime"
             var="_sourcepresenttimeout"/>
        <input name="capturetimeouttime" var="_capturetimeout"/>
        <input name="maxFRRRequested" var="_maxFRRRequested"/>
        <input name="supportFRR" var="_supportFRR" />
     </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="be0a1330-d59e-11d8-9669-0800200c9a66"
          function="BioSPI_ModuleEventHandler"/>
   </assertion>

   <activity name="BioSPI_VerifyMatch">
     <input name="moduleUuid"/>
     <input name="moduleVersionMajor"/>
     <input name="moduleVersionMinor"/>
     <input name="inserttimeouttime"/>
     <input name="nosourcepresentsupported" />
     <input name="sourcepresenttimeouttime"/>
     <input name="capturetimeouttime"/>
     <input name="maxFRRRequested" />
     <input name="supportFRR" />

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

     <!-- Invoke the functions BioSPI_ModuleLoad and BioSPI_ModuleAttach
exposed by the BSP under test.
          The input value for the parameter "deviceIDOrNull" is "0", therefore
the assertion will test the BSP's default device. -->
     <invoke activity="LoadAndAttach"
```

```
          package="be0a1330-d59e-11d8-9669-0800200c9a66"
          break_on_break="true">
        <input name="moduleUuid" var="moduleUuid"/>
        <input name="moduleVersionMajor" var="moduleVersionMajor"/>
        <input name="moduleVersionMinor" var="moduleVersionMinor"/>
        <input name="deviceIDOrNull" value="0"/>
        <input name="module" var="_modulehandle"/>
        <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="ModuleHandle" var="_modulehandle"/>
        <input name="Purpose"
              var="__BioAPI_PURPOSE_ENROLL_FOR_VERIFICATION_ONLY"/>
        <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="ModuleHandle" var="_modulehandle"/>
        <input name="Purpose" var="__BioAPI_PURPOSE_VERIFY"/>
        <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="ModuleHandle" var="_modulehandle"/>
        <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>

     <!-- Initialized 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="be0a1330-d59e-11d8-9669-0800200c9a66"
          break_on_break="true" >
        <only_if>
          <not_equal_to var1="processedLevel"
               var2="__BioAPI_BIR_DATA_TYPE_PROCESSED" />
        </only_if>
        <input name="ModuleHandle" var="_modulehandle"/>
        <input name="CapturedBIR_BIRHandle" var="capturedbir_handle"/>
     </invoke>

     <!-- Invoke the function BioSPI_VerifyMatch to verify the processed BIR
against the stored template -->
     <invoke function="BioSPI_VerifyMatch" >
        <input name="ModuleHandle" var="_modulehandle" />
        <input name="MaxFARRequested" value="2147483647" />
        <input name="MaxFRRRequested" var="maxFRRRequested" />
        <input name="FARPrecedence" value="false"/>
        <input name="ProcessedBIR_Form" var="__BioAPI_BIR_HANDLE_INPUT" />
        <input name="ProcessedBIR_BIRHandle" var="_processedbir_handle" />
        <input name="StoredTemplate_Form"
             var="__BioAPI_BIR_HANDLE_INPUT" />
```

```
        <input name="StoredTemplate_BIRHandle" var="template_handle" />
        <input name="no_AdaptedBIR" value="true" />
        <input name="no_Payload" value="true" />
        <output name="Result" setvar="result" />
        <output name="FARAchieved" setvar="farAchieved" />
        <output name="FRRAchieved" setvar="frrAchieved" />
        <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_VerifyMatch has returned BioAPI_OK
        </description>
        <equal_to var1="return" var2="__BioAPI_OK"/>
     </assert_condition>

     <invoke activity="check_FRR_supported"
          package="be0a1330-d59e-11d8-9669-0800200c9a66"
          break_on_break="true">
        <only_if>
          <same_as var1="_supportFRR" value2="true" />
        </only_if>
        <input name="frrAchieved" var="frrAchieved" />
     </invoke>

     <invoke activity="check_FRR_not_supported"
          package="be0a1330-d59e-11d8-9669-0800200c9a66"
          break_on_break="true" >
        <only_if>
          <same_as var1="_supportFRR" value2="false" />
        </only_if>
        <input name="frrAchieved" var="frrAchieved" />
     </invoke>

     <!-- Invoke the functions BioSPI_ModuleDetach and BioSPI_ModuleUnload -->
     <invoke activity="DetachAndUnload"
          package="be0a1330-d59e-11d8-9669-0800200c9a66" >
        <input name="moduleUuid" var="moduleUuid" />
        <input name="module" var="_modulehandle" />
     </invoke>
   </activity>

</package>
```

16.29 Feature 16e. Return of payload

16.29.1 BioAPI Specification References:
3.3.4.4

16.29.2 Assertion 16e.1 BioSPI_VerifyMatch_Payload

16.29.2.1 Test Purpose:
To test Return of payload if supported by the BSP and verification succeeds.

16.29.2.2 Test Scenario:
If the BSP supports the return of payload, call BioSPI_VerifyMatch with valid parameters. ProcessedBIR and Stored template should be a "match."

16.29.2.3 Expected Results:
Return code BioAPI_OK, Result of BioAPI_TRUE and a valid Payload.

16.29.2.4 XML:

```
<package name="48d18e60-2e98-11d9-9669-0800200c9a66">
   <author>
      author
   </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 1.1 is quoted below from subclauses 2.2.1.2,
1.8 and 2.5.3.4.
         ____________________________________________________________
         BioAPI_RETURN BioAPI BioSPI_VerifyMatch
            (BioAPI_HANDLE ModuleHandle,
            const BioAPI_FAR *MaxFARRequested,
            const BioAPI_FRR *MaxFRRRequested,
            const BioAPI_BOOL *FARPrecedence,
            const BioAPI_INPUT_BIR *ProcessedBIR,
            const BioAPI_INPUT_BIR *StoredTemplate,
            BioAPI_BIR_HANDLE *AdaptedBIR,
            BioAPI_BOOL *Result,
            BioAPI_FAR_PTR FARAchieved,
            BioAPI_FRR_PTR FRRAchieved,
            BioAPI_DATA_PTR *Payload);

         This function performs a verification (1-to-1) match between two BIRs;
the ProcessedBIR and the StoredTemplate.
         If the StoredTemplate contains a Payload, the Payload may be returned
upon successful verification if the FARAchieved is sufficiently stringent; this
is controlled by the policy of the BSP.
         ____________________________________________________________
         Section 2.2.1.2:
         #define BioAPI_PAYLOAD (0x00001000)
         If set, indicates that the BSP supports payload carry (accepts payload
during enroll/process and returns payroll upon successful verify).
         Section 1.8:
         This 'payload' is released to the application on successful verification
of the template.
         Section 2.5.3.4:
         Parameters: Payload (output/optional) - if the StoredTemplate contains a
payload, it is returned in an allocated BioAPI_DATA structure if the
FARAchieved 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 fail, an UNDECIDED conformity
response is issued.
     </description>

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

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

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

     <!-- 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"/>

     <!-- MaxFARRequested for BioSPI_VerifyMatch -->
     <input name="_maxFARRequested" />

     <!-- MaxFRRRequested for BioSPI_VerifyMatch -->
     <input name="_maxFRRRequested" />

     <!-- MaxFARPrecedence for BioSPI_VerifyMatch -->
     <input name="_farprecedence" />

     <!-- 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="moduleUuid" var="_moduleUuid"/>
        <input name="moduleVersionMajor" var="_moduleVersionMajor"/>
        <input  name="moduleVersionMinor"
             var="_moduleVersionMinor"/>
        <input name="inserttimeouttime" var="_inserttimeout"/>
```

```
        <input name="nosourcepresentsupported"
             var="_noSourcePresentSupported" />
        <input name="sourcepresenttimeouttime"
             var="_sourcepresenttimeout"/>
        <input name="capturetimeouttime" var="_capturetimeout"/>
        <input name="maxFARRequested" var="_maxFARRequested"/>
        <input name="maxFRRRequested" var="_maxFRRRequested"/>
        <input name="farprecedence" var="_farprecedence" />
        <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="be0a1330-d59e-11d8-9669-0800200c9a66"
          function="BioSPI_ModuleEventHandler"/>
   </assertion>

   <activity name="BioSPI_VerifyMatch">
     <input name="moduleUuid"/>
     <input name="moduleVersionMajor"/>
     <input name="moduleVersionMinor"/>
     <input name="inserttimeouttime"/>
     <input name="nosourcepresentsupported" />
     <input name="sourcepresenttimeouttime"/>
     <input name="capturetimeouttime"/>
     <input name="maxFARRequested" />
     <input name="maxFRRRequested" />
     <input name="farprecedence" />
     <input name="payloadSupported"/>
     <input name="payloadPolicy" />
     <input name="payload" />

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

     <!-- Invoke the functions BioSPI_ModuleLoad and BioSPI_ModuleAttach
exposed by the BSP under test.
          The input value for the parameter "deviceIDOrNull" is "0", therefore
the assertion will test the BSP's default device. -->
     <invoke activity="LoadAndAttach"
          package="be0a1330-d59e-11d8-9669-0800200c9a66"
          break_on_break="true">
        <input name="moduleUuid" var="moduleUuid"/>
        <input name="moduleVersionMajor" var="moduleVersionMajor"/>
        <input name="moduleVersionMinor" var="moduleVersionMinor"/>
        <input name="deviceIDOrNull" value="0"/>
        <input name="module" var="_modulehandle"/>
        <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="ModuleHandle" var="_modulehandle"/>
        <input name="Purpose"
             var="__BioAPI_PURPOSE_ENROLL_FOR_VERIFICATION_ONLY"/>
        <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_Capture again to capture another BIR -->
     <invoke function="BioSPI_Capture">
        <input name="ModuleHandle" var="_modulehandle"/>
        <input name="Purpose" var="__BioAPI_PURPOSE_VERIFY"/>
        <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="ModuleHandle" var="_modulehandle"/>
        <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="be0a1330-d59e-11d8-9669-0800200c9a66"
          break_on_break="true" >
        <only_if>
          <not_equal_to var1="processedLevel"
               var2="__BioAPI_BIR_DATA_TYPE_PROCESSED" />
        </only_if>
        <input name="ModuleHandle" var="_modulehandle"/>
        <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="ModuleHandle" var="_modulehandle" />
        <input name="MaxFARRequested" var="maxFARRequested" />
        <input name="MaxFRRRequested" var="maxFRRRequested" />
        <input name="FARPrecedence"  var="farprecedence" />
        <input name="ProcessedBIR_Form"
             var="__BioAPI_BIR_HANDLE_INPUT" />
        <input name="ProcessedBIR_BIRHandle"
             var="_processedbir_handle" />
        <input name="StoredTemplate_Form"
             var="__BioAPI_BIR_HANDLE_INPUT" />
        <input name="StoredTemplate_BIRHandle"
             var="template_handle" />
        <input name="no_AdaptedBIR" value="true" />
        <output name="Payload" setvar="output_payload" />
        <output name="Result" setvar="result" />
        <output name="FARAchieved" setvar="farAchieved" />
        <output name="FRRAchieved" setvar="frrAchieved" />
        <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="be0a1330-d59e-
11d8-9669-0800200c9a66" 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="farAchieved" var="farAchieved"/>
     </invoke>

     <!-- Check the returned payload (BSP not claiming to support the payload)
-->
     <invoke activity="payloadNotSupport_checkPayload" package="be0a1330-d59e-
11d8-9669-0800200c9a66" 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_ModuleDetach and BioSPI_ModuleUnload -->
     <invoke activity="DetachAndUnload"
           package="be0a1330-d59e-11d8-9669-0800200c9a66" >
        <input name="moduleUuid" var="moduleUuid" />
        <input name="module" var="_modulehandle" />
     </invoke>
   </activity>

</package>
```

16.30 Feature 18. BioSPI Enroll

16.30.1 BioAPI Specification References:
3.3.4.6

16.30.2 Assertion 18.1 BioSPI_Enroll_ValidParam

16.30.2.1 Test Purpose:
To test BioSPI_Enroll with valid parameters.

16.30.2.2 Test Scenario:
BioSPI_Enroll is called with valid parameters and a valid biometric is presented.

16.30.2.3 Expected Results:
Return code BioAPI_OK and NewTemplate is valid..

16.30.2.4 XML:

```
<package name="f502b220-2f3f-11d9-9669-0800200c9a66">
```

```
   <author>
      author
   </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 1.1 is quoted below from subclauses 4.2 and
2.5.3.6.
         ____________________________________________________________
         BioAPI_RETURN BioAPI BioSPI_Enroll
            (BioAPI_HANDLE ModuleHandle,
            BioAPI_BIR_PURPOSE Purpose,
            const BioAPI_INPUT_BIR *StoredTemplate,
            BioAPI_BIR_HANDLE_PTR NewTemplate,
            const BioAPI_DATA *Payload,
            sint32 Timeout
            BioAPI_BIR_HANDLE_PTR AuditData);

         This function captures biometric data from the attached device for the
purpose of creating a ProcessedBIR for the purpose of enrollment.
         ____________________________________________________________
         Section 2.5.3.6:
         This function captures biometric data from the attached device for the
purpose of creating a ProcessedBIR for the purpose of enrollment.
         Section 4.2:
         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 fail, an UNDECIDED conformity
response is issued.
      </description>
      <!-- UUID of the BSP under test -->
      <input name="_moduleUuid"/>
      <!-- Major version number of the BSP under test -->
      <input name="_moduleVersionMajor"/>
      <!-- Minor version number of the BSP under test -->
      <input name="_moduleVersionMinor"/>
      <!-- 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="moduleUuid" var="_moduleUuid"/>
        <input name="moduleVersionMajor" var="_moduleVersionMajor"/>
        <input name="moduleVersionMinor" var="_moduleVersionMinor"/>
        <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="be0a1330-d59e-11d8-9669-
0800200c9a66" function="BioSPI_ModuleEventHandler"/>
   </assertion>
   <activity name="BioSPI_Enroll">
     <input name="moduleUuid"/>
     <input name="moduleVersionMajor"/>
     <input name="moduleVersionMinor"/>
     <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="_modulehandle" value="1"/>
     <!-- Invoke the functions BioSPI_ModuleLoad and BioSPI_ModuleAttach
exposed by the BSP under test.
          The input value for the parameter "deviceIDOrNull" is "0", therefore
the assertion will test the BSP's default device. -->
     <invoke activity="LoadAndAttach" package="be0a1330-d59e-11d8-9669-
0800200c9a66" break_on_break="true">
        <input name="moduleUuid" var="moduleUuid"/>
        <input name="moduleVersionMajor" var="moduleVersionMajor"/>
        <input name="moduleVersionMinor" var="moduleVersionMinor"/>
        <input name="deviceIDOrNull" value="0"/>
        <input name="module" var="_modulehandle"/>
        <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="ModuleHandle" var="_modulehandle"/>
```

```
        <input name="Purpose"
var="__BioAPI_PURPOSE_ENROLL_FOR_VERIFICATION_ONLY"/>
        <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="ModuleHandle" var="_modulehandle"/>
        <input name="Handle" var="newtemplate_handle"/>
        <output name="Length" setvar="length"/>
        <output name="HeaderVersion" setvar="headerversion"/>
        <output name="ProcessedLevel" setvar="processedLevel"/>
        <output name="FormatOwner" setvar="formatowner"/>
        <output name="FormatId" setvar="formatid"/>
        <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, 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_ModuleDetach and BioSPI_ModuleUnload -->
     <invoke activity="DetachAndUnload" package="be0a1330-d59e-11d8-9669-
0800200c9a66">
        <input name="moduleUuid" var="moduleUuid"/>
        <input name="module" var="_modulehandle"/>
     </invoke>
   </activity>
</package>
```

16.30.3 Assertion 18.2 BioSPI_Enroll_InvalidPurpose

16.30.3.1 Test Purpose:

To test BioSPI_Enroll with an invalid Purpose.

16.30.3.2 Test Scenario:
BioSPI_Enroll is called with valid parameters, except Purpose, which is invalid.. A valid biometric is presented.

16.30.3.3 Expected Results:
Return code BioAPIERR_BSP_PURPOSE_NOT_SUPPORTED.

16.30.3.4 XML:

16.30.4 Assertion 18.3 BioSPI_Enroll_Timeout

16.30.4.1 Test Purpose:
To test BioSPI_Enroll timeout feature.

16.30.4.2 Test Scenario:
BioSPI_Enroll is called with valid parameters but is not presented with a biometric.

16.30.4.3 Expected Results:
Return code BioAPIERR_BSP_TIMEOUT_EXPIRED.

16.30.4.4 XML:

16.30.5 Assertion 18.4 BioSPI_Enroll_InvalidTimeout

16.30.5.1 Test Purpose:
To test failure on negative Timeout other than the value -1.

16.30.5.2 Test Scenario:
BioSPI_Enroll is passed valid parameters except Timeout, with is a negative number other than -1.

16.30.5.3 Expected Results:
Return code other than BioAPI_OK.

16.30.5.4 XML:

16.30.6 Assertion 18.5 BioSPI_Enroll_Payload

16.30.6.1 Test Purpose:
To test that calling BioSPI_Enroll with a payload returns BioAPI_OK

16.30.6.2 Test 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.

16.30.6.3 Expected Results:
Return code of BioAPI_OK.

16.30.6.4 XML:
```
<package name="ebe2c3e0-2f4b-11d9-9669-0800200c9a66">
```

```
   <author>
      author
   </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 1.1 is quoted below from subclauses 1.6,
2.2.1.2 and 2.5.3.6.
         ____________________________________________________________
         BioAPI_RETURN BioAPI BioSPI_Enroll
            (BioAPI_HANDLE ModuleHandle,
            BioAPI_BIR_PURPOSE Purpose,
            const BioAPI_INPUT_BIR *StoredTemplate,
            BioAPI_BIR_HANDLE_PTR NewTemplate,
            const BioAPI_DATA *Payload,
            sint32 Timeout
            BioAPI_BIR_HANDLE_PTR AuditData);

         This function captures biometric data from the attached device for the
purpose of creating a ProcessedBIR for the purpose of enrollment.
         ____________________________________________________________
         Section 2.5.3.6:
         Parameters: Payload (input/optional) - a pointer to data that will be
wrapped inside the newly created template.
         This parameter is ignored, if NULL.
         Section 1.6:
         The BSP may optionally allow the application to provide a 'payload' to
wrap inside the new template.
         Section 2.2.1.2:
         #define BioAPI_PAYLOAD (0x00001000)
         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_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 fail, an UNDECIDED conformity
response is issued.
      </description>
      <!-- UUID of the BSP under test -->
      <input name="_moduleUuid"/>
      <!-- Major version number of the BSP under test -->
      <input name="_moduleVersionMajor"/>
      <!-- Minor version number of the BSP under test -->
      <input name="_moduleVersionMinor"/>
      <!-- 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 -->
      <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="moduleUuid" var="_moduleUuid"/>
         <input name="moduleVersionMajor" var="_moduleVersionMajor"/>
         <input name="moduleVersionMinor" var="_moduleVersionMinor"/>
         <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="be0a1330-d59e-11d8-9669-
0800200c9a66" function="BioSPI_ModuleEventHandler"/>
   </assertion>
   <activity name="BioSPI_Enroll">
      <input name="moduleUuid"/>
      <input name="moduleVersionMajor"/>
      <input name="moduleVersionMinor"/>
      <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="_modulehandle" value="1"/>
      <!-- Invoke the functions BioSPI_ModuleLoad and BioSPI_ModuleAttach
exposed by the BSP under test.
           The input value for the parameter "deviceIDOrNull" is "0", therefore
the assertion will test the BSP's default device. -->
      <invoke activity="LoadAndAttach" package="be0a1330-d59e-11d8-9669-
0800200c9a66" break_on_break="true">
         <input name="moduleUuid" var="moduleUuid"/>
         <input name="moduleVersionMajor" var="moduleVersionMajor"/>
         <input name="moduleVersionMinor" var="moduleVersionMinor"/>
         <input name="deviceIDOrNull" value="0"/>
         <input name="module" var="_modulehandle"/>
         <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="ModuleHandle" var="_modulehandle"/>
        <input name="Purpose"
var="__BioAPI_PURPOSE_ENROLL_FOR_VERIFICATION_ONLY"/>
        <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_WRAP_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_WRAP_PAYLOAD"/>
              <same_as var1="supportpayload" value2="false"/>
           </and>
        </or>
     </assert_condition>
     <!-- Invoke the functions BioSPI_ModuleDetach and BioSPI_ModuleUnload -->
     <invoke activity="DetachAndUnload" package="be0a1330-d59e-11d8-9669-
0800200c9a66">
        <input name="moduleUuid" var="moduleUuid"/>
        <input name="module" var="_modulehandle"/>
     </invoke>
   </activity>
</package>
```

16.31 Feature 18a. Template update

16.31.1 BioAPI Specification References:
3.3.4.6

16.31.2 Assertion 18a.1 BioSPI_Enroll_TemplateAdaptation

16.31.2.1 Test Purpose:
Test template adaptation if supported by the BSP.

16.31.2.2 Test Scenario:
BioSPI_Enroll is passed valid parameters, including StoredTemplate.. A valid biometric is presented.

16.31.2.3 Expected Results:
Return code BioAPI_OK and a valid NewTemplate.

16.31.2.4 XML:

16.32 Feature 18b. Acceptance of payload for inclusion of enrollment BIR

16.32.1 BioAPI Specification References:
3.3.4.6

16.32.2 Assertion 18b.1 BioSPI_Enroll_PayloadTooLarge

16.32.2.1 Test Purpose:
To test failure on Payload too large if Payload supported by the BSP.

16.32.2.2 Test Scenario:
BioSPI_Enroll is passed valid parameters except Payload, whose size exceeds the maximum payload size specified in the module registry.

16.32.2.3 Expected Results:
Return code other than BioAPI_OK.

16.32.2.4 XML:

16.33 Feature 18c. Return of raw/audit data

16.33.1 BioAPI Specification References:
3.3.4.6

16.33.2 Assertion 18c.1 BioSPI_Enroll_AuditData

16.33.2.1 Test Purpose:
To test AuditData.

16.33.2.2 Test Scenario:
BioAPI_Enroll is passed valid parameters. A valid biometric is presented.

16.33.2.3 Expected Results:
One of the following:
1. Return code BioAPI_OK and valid AuditData.
2. BioAPI_UNSUPPORTED_BIR_HANDLE if the BSP does not support audit data.

3. BioAPI_INVALID_BIR_HANDLE if the BSP supports audit data but no audit data is available.

16.33.2.4 XML:

```
<package name="d4ab3b80-2f6e-11d9-9669-0800200c9a66">
   <author>
      author
   </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 1.1 is quoted below from subclauses 2.2.1.2,
4.2.4.2 and 2.5.3.6.
         ____________________________________________________________
         BioAPI_RETURN BioAPI BioSPI_Enroll
            (BioAPI_HANDLE ModuleHandle,
            BioAPI_BIR_PURPOSE Purpose,
            const BioAPI_INPUT_BIR *StoredTemplate,
            BioAPI_BIR_HANDLE_PTR NewTemplate,
            const BioAPI_DATA *Payload,
            sint32 Timeout
            BioAPI_BIR_HANDLE_PTR AuditData);

         This function captures biometric data from the attached device for the
purpose of creating a ProcessedBIR for the purpose of enrollment.
         ____________________________________________________________
         Section 2.5.3.6:
         Parameters: AuditData (output/optional) - a handle to a BIR containing
biometric audit data.
         This data may be used to provide a 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.
         Section 2.2.1.2: BioAPI_OPTIONS_MASK
         #define BioAPI_RAW (0x00000001)
         If set, indicates that the BSP supports the return of raw/audit data.
         Section 4.2.4.2:
         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.
         NOTE: BioAPI SPECIFICATION AMBIGUOUS.
         If audit data is not supported, Section 2.5.3.6 and Section 4.2.4.2
suggest different return value for 'AuditData'.
         ____________________________________________________________

         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 fail, an UNDECIDED conformity
response is issued.
     </description>
     <!-- UUID of the BSP under test -->
     <input name="_moduleUuid"/>
     <!-- Major version number of the BSP under test -->
     <input name="_moduleVersionMajor"/>
     <!-- Minor version number of the BSP under test -->
     <input name="_moduleVersionMinor"/>
     <!-- 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="moduleUuid" var="_moduleUuid"/>
        <input name="moduleVersionMajor" var="_moduleVersionMajor"/>
        <input name="moduleVersionMinor" var="_moduleVersionMinor"/>
        <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="be0a1330-d59e-11d8-9669-
0800200c9a66" function="BioSPI_ModuleEventHandler"/>
   </assertion>
   <activity name="BioSPI_Enroll">
     <input name="moduleUuid"/>
     <input name="moduleVersionMajor"/>
     <input name="moduleVersionMinor"/>
     <input name="inserttimeouttime"/>
     <input name="nosourcepresentsupported"/>
     <input name="sourcepresenttimeouttime"/>
     <input name="capturetimeouttime"/>
     <input name="supportAuditData"/>
     <!-- This assertion will use ModuleHandle "1" for all BioSPI calls that
require it -->
     <set name="_modulehandle" value="1"/>
     <!-- Invoke the functions BioSPI_ModuleLoad and BioSPI_ModuleAttach
exposed by the BSP under test.
```

```
          The input value of the parameter "deviceIDOrNull" is "0", therefore
the assertion will test the BSP's default device. -->
     <invoke activity="LoadAndAttach" package="be0a1330-d59e-11d8-9669-
0800200c9a66" break_on_break="true">
        <input name="moduleUuid" var="moduleUuid"/>
        <input name="moduleVersionMajor" var="moduleVersionMajor"/>
        <input name="moduleVersionMinor" var="moduleVersionMinor"/>
        <input name="deviceIDOrNull" value="0"/>
        <input name="module" var="_modulehandle"/>
        <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="ModuleHandle" var="_modulehandle"/>
        <input name="Purpose"
var="__BioAPI_PURPOSE_ENROLL_FOR_VERIFICATION_ONLY"/>
        <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 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"/>
        </only_if>
        <input name="ModuleHandle" var="_modulehandle"/>
        <input name="Handle" var="auditbir_handle"/>
     </invoke>
     <!-- Invoke the functions BioSPI_ModuleDetach and BioSPI_ModuleUnload -->
     <invoke activity="DetachAndUnload" package="be0a1330-d59e-11d8-9669-
0800200c9a66">
        <input name="moduleUuid" var="moduleUuid"/>
        <input name="module" var="_modulehandle"/>
     </invoke>
   </activity>
   <activity name="check_audit_data_type">
      <input name="ModuleHandle"/>
      <input name="Handle"/>
      <!-- Invoke the function BioSPI_GetHeaderFromHandle. -->
      <invoke function="BioSPI_GetHeaderFromHandle">
         <input name="ModuleHandle" var="_modulehandle"/>
         <input name="Handle" var="Handle"/>
         <output name="Length" setvar="length"/>
         <output name="HeaderVersion" setvar="headerversion"/>
         <output name="ProcessedLevel" setvar="processedLevel"/>
         <output name="FormatOwner" setvar="formatowner"/>
         <output name="FormatId" setvar="formatid"/>
         <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, 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>
```

16.34 Feature 18d. Return of quality in the enrollment BIR header

16.34.1 BioAPI Specification References:
3.3.4.6, (2.1.46, 4.2.4.2)

16.34.2 Assertion 18d.1 BioSPI_Enroll_BIRHeaderQuality

16.34.2.1 Test Purpose:
To test the return of quality in the enrollment BIR header if supported by the BSP.

16.34.2.2 Test Scenario:
If the BSP supports return of quality, call BioSPI_Enroll with valid parameters. Present a valid biometric.

16.34.2.3 Expected Results:
Return code BioAPI_OK and a valid NewTemplate that includes the quality in its header.

16.34.2.4 XML:

```
<package name="0e8932f0-2f72-11d9-9669-0800200c9a66">
   <author>
      author
   </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 1.1 is quoted below from subclauses 2.2.1.2
and 4.2.4.2.
         ____________________________________________________________
         BioAPI_RETURN BioAPI BioSPI_Enroll
            (BioAPI_HANDLE ModuleHandle,
            BioAPI_BIR_PURPOSE Purpose,
            const BioAPI_INPUT_BIR *StoredTemplate,
            BioAPI_BIR_HANDLE_PTR NewTemplate,
            const BioAPI_DATA *Payload,
            sint32 Timeout
            BioAPI_BIR_HANDLE_PTR AuditData);

         This function captures biometric data from the attached device for the
purpose of creating a ProcessedBIR for the purpose of enrollment.

         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.

Section 2.2.1.2:
#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.
Section 4.2.4.2:
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.
Similarly, when a BIR is processed, another quality calculation may be performed and the quality valueincluded 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.

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 fail, an UNDECIDED conformity response is issued.

```
</description>

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

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

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

<!-- 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="moduleUuid" var="_moduleUuid"/>
        <input name="moduleVersionMajor" var="_moduleVersionMajor"/>
        <input name="moduleVersionMinor" var="_moduleVersionMinor"/>
        <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="be0a1330-d59e-11d8-9669-0800200c9a66"
          function="BioSPI_ModuleEventHandler"/>
   </assertion>

   <activity name="BioSPI_Enroll">
     <input name="moduleUuid"/>
     <input name="moduleVersionMajor"/>
     <input name="moduleVersionMinor"/>
     <input name="inserttimeouttime"/>
     <input name="nosourcepresentsupported"/>
     <input name="sourcepresenttimeouttime"/>
     <input name="capturetimeouttime"/>
     <input name="processedQualitySupported"/>

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

     <!-- Invoke the functions BioSPI_ModuleLoad and BioSPI_ModuleAttach
exposed by the BSP under test.
          The input value for the parameter "deviceIDOrNull" is "0", therefore
the assertion will test the BSP's default device. -->
     <invoke activity="LoadAndAttach"
          package="be0a1330-d59e-11d8-9669-0800200c9a66"
          break_on_break="true">
        <input name="moduleUuid" var="moduleUuid"/>
        <input name="moduleVersionMajor" var="moduleVersionMajor"/>
        <input name="moduleVersionMinor" var="moduleVersionMinor"/>
        <input name="deviceIDOrNull" value="0"/>
        <input name="module" var="_modulehandle"/>
        <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="ModuleHandle" var="_modulehandle"/>
        <input name="Purpose"
             var="__BioAPI_PURPOSE_ENROLL_FOR_VERIFICATION_ONLY"/>
        <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="ModuleHandle" var="_modulehandle"/>
        <input name="Handle" var="newtemplate_handle"/>
        <output name="Length" setvar="length"/>
        <output name="HeaderVersion" setvar="headerversion"/>
        <output name="ProcessedLevel" setvar="processedLevel"/>
        <output name="FormatOwner" setvar="formatowner"/>
        <output name="FormatId" setvar="formatid"/>
        <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, 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="be0a1330-d59e-11d8-9669-0800200c9a66"
            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="be0a1330-d59e-11d8-9669-0800200c9a66"
            break_on_break="true" >
        <only_if>
            <same_as var1="processedQualitySupported"
                  value2="false" />
        </only_if>
        <input name="quality" var="quality" />
      </invoke>

      <!-- Invoke the functions BioSPI_ModuleDetach and BioSPI_ModuleUnload -->
      <invoke activity="DetachAndUnload"
            package="be0a1330-d59e-11d8-9669-0800200c9a66" >
        <input name="moduleUuid" var="moduleUuid"/>
        <input name="module" var="_modulehandle"/>
      </invoke>
    </activity>

</package>
```

16.35 Feature 18f. BIR signing (by BSP)

16.35.1 BioAPI Specification References:
1.5, 2.1.7, 4.2.4.2

16.35.2 Assertion 18f.1 BioSPI_Enroll_BIRSigned

16.35.2.1 Test Purpose:
To test BIR signing if supported by the BSP.

16.35.2.2 Test Scenario:
If the BSP supports signing cal BioSPI_Enroll with valid parameters. Present a valid biometric.

16.35.2.3 Expected Results:
Return code BioAPI_OK and a valid NewTemplate that includes a signature.

16.35.2.4 XML:

16.36 Feature 18g. BIR encryption (by BSP)

16.36.1 BioAPI Specification References:
1.5, 2.1.7

16.36.2 Assertion 18g.1 BioSPI_Enroll_BIREncrypted

16.36.2.1 Test Purpose:
To test BIR encryption if supported by the BSP.

16.36.2.2 Test Scenario:
If the BSP supports BIR encryption call BioSPI_Enroll with valid parameters. Present a valid biometric.

16.36.2.3 Expected Results:
Return code BioAPI_OK and a valid NewTemplate.

16.36.2.4 XML:

16.37 Feature 19. BioSPI Verify

16.37.1 BioAPI Specification References:
3.3.4.7

16.37.2 Assertion 19.1 BioSPI_Verify_ValidParam

16.37.2.1 Test Purpose:
To test BioSPI_Verify with valid parameters.

16.37.2.2 Test Scenario:
Valid parameters are passed to BioSPI_Verify.

16.37.2.3 Expected Results:
Return code BioAPI_OK. Result can be either BioAPI_TRUE or BioAPI_FALSE. FARAchieved must be valid.

16.37.2.4 XML:

```
<package name="03c5ba70-30c9-11d9-9669-0800200c9a66">
   <author>
      author
   </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 1.1 is quoted below from subclauses 4.2 and
2.5.3.7.
         ____________________________________________________________
         BioAPI_RETURN BioAPI BioAPI_Verify
            (BioAPI_HANDLE ModuleHandle,
            const BioAPI_FAR *MaxFARRequested,
            const BioAPI_FRR *MaxFRRRequested,
```

```
          const BioAPI_BOOL *FARPrecedence,
          const BioAPI_INPUT_BIR *StoredTemplate,
          BioAPI_BIR_HANDLE_PTR AdaptedBIR,
          BioAPI_BOOL *Result,
          BioAPI_FAR_PTR FARAchieved,
          BioAPI_FRR_PTR FRRAchieved,
          BioAPI_DATA_PTR *Payload,
          sint32 Timeout,
          BioAPI_BIR_HANDLE_PTR AuditData);
```

This function captures biometric data from the attached device, and compares it against the StoredTemplate.

Boolean Result indicates whether verification was successful or not, and the FARAchieved is a FAR value indicating how closely the BIRs actually matched.

Section 2.5.3.7:
This function captures biometric data from the attached device, and compares it against the StoredTemplate.
Section 4.2:
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 fail, an UNDECIDED conformity response is issued.

```
     </description>

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

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

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

     <!-- 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"/>

     <!-- MaxFARRequested for BioSPI_VerifyMatch -->
     <input name="_maxFARRequested"/>
```

```
     <!-- MaxFRRRequested for BioSPI_VerifyMatch -->
     <input name="_maxFRRRequested"/>

     <!-- MaxFARPrecedence for BioSPI_VerifyMatch -->
     <input name="_farprecedence"/>

     <!-- 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="moduleUuid" var="_moduleUuid"/>
        <input name="moduleVersionMajor" var="_moduleVersionMajor"/>
        <input name="moduleVersionMinor" var="_moduleVersionMinor"/>
        <input name="inserttimeouttime" var="_inserttimeout"/>
        <input name="nosourcepresentsupported"
             var="_noSourcePresentSupported"/>
        <input name="sourcepresenttimeouttime"
             var="_sourcepresenttimeout"/>
        <input name="capturetimeouttime" var="_capturetimeout"/>
        <input name="maxFARRequested" var="_maxFARRequested"/>
        <input name="maxFRRRequested" var="_maxFRRRequested"/>
        <input name="farprecedence" var="_farprecedence"/>
        <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="be0a1330-d59e-11d8-9669-0800200c9a66"
          function="BioSPI_ModuleEventHandler"/>
   </assertion>

   <activity name="BioSPI_Verify">
     <input name="moduleUuid"/>
     <input name="moduleVersionMajor"/>
     <input name="moduleVersionMinor"/>
     <input name="inserttimeouttime"/>
     <input name="nosourcepresentsupported"/>
     <input name="sourcepresenttimeouttime"/>
     <input name="capturetimeouttime"/>
     <input name="maxFARRequested"/>
     <input name="maxFRRRequested"/>
     <input name="farprecedence"/>
     <input name="verifytimeout"/>

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

     <!-- Invoke the functions BioSPI_ModuleLoad and BioSPI_ModuleAttach
exposed by the BSP under test.
          The input value for the parameter "deviceIDOrNull" is "0", therefore
the assertion will test the BSP's default device. -->
     <invoke activity="LoadAndAttach"
          package="be0a1330-d59e-11d8-9669-0800200c9a66"
          break_on_break="true">
        <input name="moduleUuid" var="moduleUuid"/>
        <input name="moduleVersionMajor" var="moduleVersionMajor"/>
        <input name="moduleVersionMinor" var="moduleVersionMinor"/>
        <input name="deviceIDOrNull" value="0"/>
```

```
        <input name="module" var="_modulehandle"/>
        <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="ModuleHandle" var="_modulehandle"/>
        <input name="Purpose"
             var="__BioAPI_PURPOSE_ENROLL_FOR_VERIFICATION_ONLY"/>
        <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="ModuleHandle" var="_modulehandle"/>
        <input name="MaxFARRequested" var="maxFARRequested"/>
        <input name="MaxFRRRequested" var="maxFRRRequested"/>
        <input name="FARPrecedence" var="farprecedence"/>
        <input name="StoredTemplate_Form"
             var="__BioAPI_BIR_HANDLE_INPUT"/>
        <input name="StoredTemplate_BIRHandle"
             var="template_handle"/>
        <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="FARAchieved" setvar="farAchieved"/>
        <output name="FRRAchieved" setvar="frrAchieved"/>
        <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_ModuleDetach and BioSPI_ModuleUnload -->
     <invoke activity="DetachAndUnload"
          package="be0a1330-d59e-11d8-9669-0800200c9a66" >
        <input name="moduleUuid" var="moduleUuid"/>
        <input name="module" var="_modulehandle"/>
     </invoke>
   </activity>

</package>
```

16.37.3 Assertion 19.2 BioSPI_Verify_FARUnspecified

16.37.3.1 Test Purpose:
To test failure on unspecified FAR.

16.37.3.2 Test Scenario:
BioSPI_Verify is passed valid parameters except for MaxFARRequested.

16.37.3.3 Expected Results:
Return code other than BioAPI_OK.

16.37.3.4 XML:

```
<package name="018ed978-07e1-1085-8934-0002a5d5fd2e">
   <author>
      author
   </author>

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

   <assertion name="BioSPI_Verify_FARUnspecified" model="BSPTesting">
      <description>
         This assertion invokes the function BioSPI_Verify with MaxFARRequested
equal to NULL.  The function is expected to return an error.
         The relevant text in BioAPI 1.1 is quoted below from subclause 2.5.3.7.
         ____________________________________________________________
         BioAPI_RETURN BioAPI BioAPI_Verify
            (BioAPI_HANDLE ModuleHandle,
            const BioAPI_FAR *MaxFARRequested,
```

```
          const BioAPI_FRR *MaxFRRRequested,
          const BioAPI_BOOL *FARPrecedence,
          const BioAPI_INPUT_BIR *StoredTemplate,
          BioAPI_BIR_HANDLE_PTR AdaptedBIR,
          BioAPI_BOOL *Result,
          BioAPI_FAR_PTR FARAchieved,
          BioAPI_FRR_PTR FRRAchieved,
          BioAPI_DATA_PTR *Payload,
          sint32 Timeout,
          BioAPI_BIR_HANDLE_PTR AuditData);
```

This function captures biometric data from the attached device, and compares it against the StoredTemplate.

Boolean Result indicates whether verification was successful or not, and the FARAchieved is a FAR value indicating how closely the BIRs actually matched.

Section 2.5.3.7:
The application must request a maximum FAR value criterion for a successful match, and may also (optionally) request a maximum FRR criterion for a successful match.

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 with FAR set to NULL.
5) Check the return code, which is expected to be an error code.

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

```
     </description>

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

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

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

     <!-- 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 the BioSPI_Enroll -->
     <input name="_capturetimeout"/>

     <!-- MaxFRRRequested for BioSPI_VerifyMatch -->
     <input name="_maxFRRRequested"/>

     <!-- MaxFARPrecedence for BioSPI_VerifyMatch -->
```

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

     <!-- 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="moduleUuid" var="_moduleUuid"/>
        <input name="moduleVersionMajor" var="_moduleVersionMajor"/>
        <input name="moduleVersionMinor" var="_moduleVersionMinor"/>
        <input name="inserttimeouttime" var="_inserttimeout"/>
        <input name="nosourcepresentsupported"
             var="_noSourcePresentSupported"/>
        <input name="sourcepresenttimeouttime"
             var="_sourcepresenttimeout"/>
        <input name="capturetimeouttime" var="_capturetimeout"/>
        <input name="maxFRRRequested" var="_maxFRRRequested"/>
        <input name="farprecedence" var="_farprecedence"/>
        <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="be0a1330-d59e-11d8-9669-0800200c9a66"
        function="BioSPI_ModuleEventHandler"/>
   </assertion>

   <activity name="BioSPI_Verify">
     <input name="moduleUuid"/>
     <input name="moduleVersionMajor"/>
     <input name="moduleVersionMinor"/>
     <input name="inserttimeouttime"/>
     <input name="nosourcepresentsupported"/>
     <input name="sourcepresenttimeouttime"/>
     <input name="capturetimeouttime"/>
     <input name="maxFRRRequested"/>
     <input name="farprecedence"/>
     <input name="verifyTimeout"/>

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

     <!-- Invoke the functions BioSPI_ModuleLoad and BioSPI_ModuleAttach
exposed by the BSP under test.
          The input value for the parameter "deviceIDOrNull" is "0", therefore
the assertion will test the BSP's default device. -->
     <invoke activity="LoadAndAttach"
          package="be0a1330-d59e-11d8-9669-0800200c9a66"
          break_on_break="true">
        <input name="moduleUuid" var="moduleUuid"/>
        <input name="moduleVersionMajor" var="moduleVersionMajor"/>
        <input name="moduleVersionMinor" var="moduleVersionMinor"/>
        <input name="deviceIDOrNull" value="0"/>
        <input name="module" var="_modulehandle"/>
        <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="ModuleHandle" var="_modulehandle"/>
        <input name="Purpose"
             var="__BioAPI_PURPOSE_ENROLL_FOR_VERIFICATION_ONLY"/>
        <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="ModuleHandle" var="_modulehandle"/>
        <input name="MaxFRRRequested" var="maxFRRRequested"/>
        <input name="FARPrecedence" var="farprecedence"/>
        <input name="StoredTemplate_Form"
             var="__BioAPI_BIR_HANDLE_INPUT"/>
        <input name="StoredTemplate_BIRHandle"
             var="template_handle"/>
        <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="FARAchieved" setvar="farAchieved"/>
        <output name="FRRAchieved" setvar="frrAchieved"/>
        <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
BioAPIERR_BSP_INVALID_INPUT_POINTER.
         </description>
         <equal_to var1="return"
               var2="__BioAPIERR_BSP_INVALID_INPUT_POINTER"/>
      </assert_condition>

      <!-- Invoke the functions BioSPI_ModuleDetach and BioSPI_ModuleUnload -->
      <invoke activity="DetachAndUnload"
            package="be0a1330-d59e-11d8-9669-0800200c9a66" >
         <input name="moduleUuid" var="moduleUuid"/>
         <input name="module" var="_modulehandle"/>
      </invoke>
   </activity>

</package>
```

16.37.4 Assertion 19.3 BioSPI_Verify_FRR

16.37.4.1 Test Purpose:
To test BioSPI_Verify using FRR as matching criteria. (The input MaxFRRRequested parameter is not NULL, and FARPrecedence set to BioAPI_FALSE).

16.37.4.2 Test Scenario:
1) Load the BSP under test
2) Attach the BSP under test
3) Call the function BioSPI_Enroll to create a template.
4) Call the function BioSPI_Verify with FRR specified, check the output parameter FRRAchived.
5) Detach and unload BSP.

16.37.4.3 Expected Results:
The returned FRRAchived is acceptable.

16.37.4.4 XML:

```
<package name="302849f0-31a2-11d9-9669-0800200c9a66">
   <author>
      author
   </author>

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

   <assertion name="BioSPI_Verify_FRR" model="BSPTesting">
      <description>
         This assertion invokes the function BioSPI_Verify exposed by the BSP
under test using FRR as verification criteria.  The function is expected to
return BioAPI_OK.
         The relevant text in BioAPI 1.1 is quoted below from subclauses 2.5.3.4
and 2.2.1.2.
```

```
        ____________________________________________________________
        BioAPI_RETURN BioAPI BioAPI_Verify
           (BioAPI_HANDLE ModuleHandle,
           const BioAPI_FAR *MaxFARRequested,
           const BioAPI_FRR *MaxFRRRequested,
           const BioAPI_BOOL *FARPrecedence,
           const BioAPI_INPUT_BIR *StoredTemplate,
           BioAPI_BIR_HANDLE_PTR AdaptedBIR,
           BioAPI_BOOL *Result,
           BioAPI_FAR_PTR FARAchieved,
           BioAPI_FRR_PTR FRRAchieved,
           BioAPI_DATA_PTR *Payload,
           sint32 Timeout,
           BioAPI_BIR_HANDLE_PTR AuditData);

        This function captures biometric data from the attached device, and
compares it against the StoredTemplate.

        Boolean Result indicates whether verification was successful or not, and
the FARAchieved is a FAR value indicating how closely the BIRs actually
matched.

        The BSP implementation may optionally return the corresponding FRR that
was achieved through the FRRAchieved return parameter.
        ____________________________________________________________
        Section 2.5.3.4:
        If a maximum FRR value is provided, the application must also indicate
via the FARPrecedence parameter, which one takes precedence.
        The Boolean Result indicates whether verification was successful or not,
and the FARAchieved is a FAR value indicating how closely the BIRs actually
matched.
        The BSP implementation may optionally return the corresponding FRR that
was achieved through the FRRAchieved return parameter.
        Section 2.2.1.2:
        #define BioAPI_FRR (0x00010000)
        If set, indicates BSP supports the return of actual FRR during matching
operations (Verify, VerifyMatch, Identify, IdentifyMatch).
        ____________________________________________________________

        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.
           4) Call the function BioSPI_Verify with FRR specified, check the
output parameter FRRAchived.
           5) Detach and unload BSP.

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

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

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

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

     <!-- 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"/>

     <!-- MaxFRRRequested for BioSPI_VerifyMatch -->
     <input name="_maxFRRRequested"/>

     <!-- Timeout for BioSPI_Verify -->
     <input name="_verifytimeout"/>

     <!-- Indicates whether the BSP under test claims support for FRR -->
     <input name="_supportFRR"/>

     <!-- Invocation of the primary activity of this assertion with input
parameter values assigned from the assertion's parameters. -->
     <invoke activity="BioSPI_Verify">
        <input name="moduleUuid" var="_moduleUuid"/>
        <input name="moduleVersionMajor" var="_moduleVersionMajor"/>
        <input name="moduleVersionMinor" var="_moduleVersionMinor"/>
        <input name="inserttimeouttime" var="_inserttimeout"/>
        <input name="nosourcepresentsupported"
             var="_noSourcePresentSupported"/>
        <input name="sourcepresenttimeouttime"
             var="_sourcepresenttimeout"/>
        <input name="capturetimeouttime" var="_capturetimeout"/>
        <input name="maxFRRRequested" var="_maxFRRRequested"/>
        <input name="verifytimeout" var="_verifytimeout"/>
        <input name="supportFRR" var="_supportFRR"/>
     </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="be0a1330-d59e-11d8-9669-0800200c9a66"
          function="BioSPI_ModuleEventHandler"/>
   </assertion>

   <activity name="BioSPI_Verify">
     <input name="moduleUuid"/>
     <input name="moduleVersionMajor"/>
     <input name="moduleVersionMinor"/>
     <input name="inserttimeouttime"/>
     <input name="nosourcepresentsupported"/>
     <input name="sourcepresenttimeouttime"/>
     <input name="capturetimeouttime"/>
     <input name="maxFRRRequested"/>
     <input name="verifytimeout"/>
     <input name="supportFRR"/>

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

     <!-- Invoke the functions BioSPI_ModuleLoad and BioSPI_ModuleAttach
```

```
exposed by the BSP under test.
          The input value for the parameter "deviceIDOrNull" is "0", therefore
the assertion will test the BSP's default device. -->
     <invoke activity="LoadAndAttach"
          package="be0a1330-d59e-11d8-9669-0800200c9a66"
          break_on_break="true">
        <input name="moduleUuid" var="moduleUuid"/>
        <input name="moduleVersionMajor" var="moduleVersionMajor"/>
        <input name="moduleVersionMinor" var="moduleVersionMinor"/>
        <input name="deviceIDOrNull" value="0"/>
        <input name="module" var="_modulehandle"/>
        <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="ModuleHandle" var="_modulehandle"/>
        <input name="Purpose"
             var="__BioAPI_PURPOSE_ENROLL_FOR_VERIFICATION_ONLY"/>
        <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 verify against the newly created
template -->
     <invoke function="BioSPI_Verify" >
        <input name="ModuleHandle" var="_modulehandle"/>
```

```
        <input name="MaxFARRequested" value="1000"/>
        <input name="MaxFRRRequested" var="maxFRRRequested"/>
        <input name="FARPrecedence" value="false"/>
        <input name="StoredTemplate_Form"
             var="__BioAPI_BIR_HANDLE_INPUT"/>
        <input name="StoredTemplate_BIRHandle"
             var="template_handle"/>
        <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="FARAchieved" setvar="farAchieved"/>
        <output name="FRRAchieved" setvar="frrAchieved"/>
        <output name="Payload"  setvar="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>

     <invoke activity="check_FRR_supported"
          package="be0a1330-d59e-11d8-9669-0800200c9a66"
          break_on_break="true" >
        <only_if>
          <same_as var1="_supportFRR" value2="true"/>
        </only_if>
        <input name="frrAchieved" var="frrAchieved"/>
     </invoke>

     <invoke activity="check_FRR_not_supported"
          package="be0a1330-d59e-11d8-9669-0800200c9a66"
          break_on_break="true">
        <only_if>
          <same_as var1="_supportFRR" value2="false"/>
        </only_if>
        <input name="frrAchieved" var="frrAchieved"/>
     </invoke>

     <!-- Invoke the functions BioSPI_ModuleDetach and BioSPI_ModuleUnload -->
     <invoke activity="DetachAndUnload"
          package="be0a1330-d59e-11d8-9669-0800200c9a66" >
        <input name="moduleUuid" var="moduleUuid"/>
        <input name="module" var="_modulehandle"/>
     </invoke>
   </activity>

</package>
```

16.38 Feature 19b. Model/template adaptation

16.38.1 BioAPI Specification References:
3.3.4.7

16.38.2 Assertion 19b.1 BioSPI_Verify_TemplateAdaptation

16.38.2.1 Test Purpose:
To test Model/template adaptation if supported by the BSP.

16.38.2.2 Test Scenario:
If the BSP supports Model/template adaptation, call BioSPI_Verify with valid parameters.

16.38.2.3 Expected Results:
Return code BioAPI_OK and a valid AdaptedBIR.

16.38.2.4 XML:

```
<package name="0de718a0-31b9-11d9-9669-0800200c9a66">
   <author>
      author
   </author>
   <description>
      This package contains the assertion "BioSPI_Verify_TemplateAdaptation"
(see the "description" element of the assertion below).
   </description>
   <assertion name="BioSPI_Verify_TemplateAdaptation" model="BSPTesting">
      <description>
         This assertion checks if calling the function BioSPI_Verify with
adaptation returns BioAPI_OK.
         The relevant text in BioAPI 1.1 is quoted below from subclauses 2.2.1.2,
4.2.4.2 and 2.5.3.7.
         ____________________________________________________
         BioAPI_RETURN BioAPI BioAPI_Verify
            (BioAPI_HANDLE ModuleHandle,
            const BioAPI_FAR *MaxFARRequested,
            const BioAPI_FRR *MaxFRRRequested,
            const BioAPI_BOOL *FARPrecedence,
            const BioAPI_INPUT_BIR *StoredTemplate,
            BioAPI_BIR_HANDLE_PTR AdaptedBIR,
            BioAPI_BOOL *Result,
            BioAPI_FAR_PTR FARAchieved,
            BioAPI_FRR_PTR FRRAchieved,
            BioAPI_DATA_PTR *Payload,
            sint32 Timeout,
            BioAPI_BIR_HANDLE_PTR AuditData);

         This function captures biometric data from the attached device, and
compares it against the StoredTemplate.

         Boolean Result indicates whether verification was successful or not, and
the FARAchieved is a FAR value indicating how closely the BIRs actually
matched.

         Optionally, a new AdaptedBIR may be constructed. The Verify function may
be split between client and server if a streaming callback has been set. Either
the client or the server can initiate the operation.
         ____________________________________________________
         Section 2.5.3.7:
         If the match is successful, an attempt may be made to adapt the
```

StoredTemplate with information taken from the ProcessedBIR. (Not all BSPs perform adaptation).

The resulting AdaptedBIR should now be considered an optimal enrollment, and be saved in the enrollment database. (It is up to the application whether or not it uses or discards this data).

It is important to note that adaptation may not occur in all cases.

Parameters: AdaptedBIR (output/optional) - a pointer to the handle of the adapted BIR.

This parameter can be NULL if an Adapted BIR is not desired.

Not all BSPs support the adaptation of BIRs.

The function may return a handle value of BioAPI_UNSUPPORTED_BIR_HANDLE to indicate that adaptation is not supported or a value of BioAPI_INVALID_BIR_HANDLE to indicate that adaptation was not possible.

Section 2.2.1.2:

#define BioAPI_ADAPTATION (0x00020000)

If set, BSP supports BIR adaptation (return of Verify or VerifyMatch operation).

Section 4.2.4.2:

The BSP makes the decision as to when and if the adaptation should be performed (based on such factors as quality, elapsed time, and significant differences).

If the BSP does not support adaptation, the returned AdaptedBIR will be set to -2.

If the BSP supports adaptation, but for some reason is not able or chooses not to perform the adaptation, then the return AdaptedBIR is set to -1.

The BSP must post to the module registry its support for adaptation.

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.
4) Call BioSPI_Verify with AdaptedBIR set to a non-NULL value.
5) Check the return code, which is expected to be BioAPI_OK.
6) Depending on whether the BSP supports adaptation or not, the adapted BIR is expected to be BioAPI_UNSUPPORTED_BIR_HANDLE or BioAPI_INVALID_BIR_HANDLE.
7) Detach and unload the BSP under test.

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

```
</description>
<!-- UUID of the BSP under test -->
<input name="_moduleUuid"/>
<!-- Major version number of the BSP under test -->
<input name="_moduleVersionMajor"/>
<!-- Minor version number of the BSP under test -->
<input name="_moduleVersionMinor"/>
<!-- 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"/>
<!-- MaxFARRequested for BioSPI_VerifyMatch -->
<input name="_maxFARRequested"/>
<!-- MaxFRRRequested for BioSPI_VerifyMatch -->
<input name="_maxFRRRequested"/>
```

```
     <!-- MaxFARPrecedence for BioSPI_VerifyMatch -->
     <input name="_farprecedence"/>
     <!-- Timeout for BioSPI_Verify -->
     <input name="_verifytimeout"/>
     <!-- Support adaptation or not -->
     <input name="_supportAdaptation"/>
     <!-- Invocation of the primary activity of this assertion with input
parameter values assigned from the assertion's parameters. -->
     <invoke activity="BioSPI_Verify">
        <input name="moduleUuid" var="_moduleUuid"/>
        <input name="moduleVersionMajor" var="_moduleVersionMajor"/>
        <input name="moduleVersionMinor" var="_moduleVersionMinor"/>
        <input name="inserttimeouttime" var="_inserttimeout"/>
        <input name="nosourcepresentsupported" var="_noSourcePresentSupported"/>
        <input name="sourcepresenttimeouttime" var="_sourcepresenttimeout"/>
        <input name="capturetimeouttime" var="_capturetimeout"/>
        <input name="maxFARRequested" var="_maxFARRequested"/>
        <input name="maxFRRRequested" var="_maxFRRRequested"/>
        <input name="farprecedence" var="_farprecedence"/>
        <input name="verifytimeout" var="_verifytimeout"/>
        <input name="supportAdaptation" var="_supportAdaptation"/>
     </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="be0a1330-d59e-11d8-9669-
0800200c9a66" function="BioSPI_ModuleEventHandler"/>
   </assertion>
   <activity name="BioSPI_Verify">
     <input name="moduleUuid"/>
     <input name="moduleVersionMajor"/>
     <input name="moduleVersionMinor"/>
     <input name="inserttimeouttime"/>
     <input name="nosourcepresentsupported"/>
     <input name="sourcepresenttimeouttime"/>
     <input name="capturetimeouttime"/>
     <input name="maxFARRequested"/>
     <input name="maxFRRRequested"/>
     <input name="farprecedence"/>
     <input name="verifytimeout"/>
     <input name="supportAdaptation"/>
     <!-- This assertion will use ModuleHandle "1" for all BioSPI calls that
require it -->
     <set name="_modulehandle" value="1"/>
     <!-- Invoke the functions BioSPI_ModuleLoad and BioSPI_ModuleAttach
exposed by the BSP under test.
          The input value for the parameter "deviceIDOrNull" is "0", therefore
the assertion will test the BSP's default device. -->
     <invoke activity="LoadAndAttach" package="be0a1330-d59e-11d8-9669-
0800200c9a66" break_on_break="true">
        <input name="moduleUuid" var="moduleUuid"/>
        <input name="moduleVersionMajor" var="moduleVersionMajor"/>
        <input name="moduleVersionMinor" var="moduleVersionMinor"/>
        <input name="deviceIDOrNull" value="0"/>
        <input name="module" var="_modulehandle"/>
        <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="ModuleHandle" var="_modulehandle"/>
        <input name="Purpose"
var="__BioAPI_PURPOSE_ENROLL_FOR_VERIFICATION_ONLY"/>
        <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 capture and verify against the
stored template -->
     <invoke function="BioSPI_Verify">
        <input name="ModuleHandle" var="_modulehandle"/>
        <input name="MaxFARRequested" var="maxFARRequested"/>
        <input name="MaxFRRRequested" var="maxFRRRequested"/>
        <input name="FARPrecedence" var="farprecedence"/>
        <input name="StoredTemplate_Form" var="__BioAPI_BIR_HANDLE_INPUT"/>
        <input name="StoredTemplate_BIRHandle" var="template_handle"/>
        <input name="no_AdaptedBIR" value="false"/>
        <input name="Timeout" var="verifytimeout"/>
        <input name="no_AuditData" value="true"/>
        <input name="no_Result" value="false"/>
        <input name="no_Payload" value="true"/>
        <output name="Result" setvar="result"/>
        <output name="AdaptedBIR" setvar="adaptedbir_handle"/>
        <output name="FARAchieved" setvar="farAchieved"/>
        <output name="FRRAchieved" setvar="frrAchieved"/>
        <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>
```

```
     <invoke activity="check_adaptation_supported" package="be0a1330-d59e-11d8-
9669-0800200c9a66" break_on_break="true">
        <only_if>
           <same_as var1="supportAdaptation" value2="true"/>
        </only_if>
        <input name="adaptedbir_handle" var="adaptedbir_handle"/>
     </invoke>
     <invoke activity="check_adaptation_not_supported" package="be0a1330-d59e-
11d8-9669-0800200c9a66" break_on_break="true">
        <only_if>
           <same_as var1="supportAdaptation" value2="false"/>
        </only_if>
        <input name="adaptedbir_handle" var="adaptedbir_handle"/>
     </invoke>
     <!-- Invoke the functions BioSPI_ModuleDetach and BioSPI_ModuleUnload -->
     <invoke activity="DetachAndUnload" package="be0a1330-d59e-11d8-9669-
0800200c9a66">
        <input name="moduleUuid" var="moduleUuid"/>
        <input name="module" var="_modulehandle"/>
     </invoke>
   </activity>
</package>
```

16.39 Feature 19e. Return of payload

16.39.1 BioAPI Specification References:
3.3.4.7

16.39.2 Assertion 19e.1 BioSPI_Verify_Payload

16.39.2.1 Test Purpose:
To test Return of payload if supported by the BSP and verification succeeds.

16.39.2.2 Test Scenario:
If the BSP supports the return of payload, call BioSPI_Verify with valid parameters. ProcessedBIR and Stored template should be a "match."

16.39.2.3 Expected Results:
Return code BioAPI_OK, Result of BioAPI_TRUE and a valid Payload.

16.39.2.4 XML:

```
<package name="12b4a540-31c4-11d9-9669-0800200c9a66">
   <author>
      author
   </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 1.1 is quoted below from subclauses 2.2.1.2
and 2.5.3.7.
         ____________________________________________________________
         BioAPI_RETURN BioAPI BioAPI_Verify
```

```
          (BioAPI_HANDLE ModuleHandle,
          const BioAPI_FAR *MaxFARRequested,
          const BioAPI_FRR *MaxFRRRequested,
          const BioAPI_BOOL *FARPrecedence,
          const BioAPI_INPUT_BIR *StoredTemplate,
          BioAPI_BIR_HANDLE_PTR AdaptedBIR,
          BioAPI_BOOL *Result,
          BioAPI_FAR_PTR FARAchieved,
          BioAPI_FRR_PTR FRRAchieved,
          BioAPI_DATA_PTR *Payload,
          sint32 Timeout,
          BioAPI_BIR_HANDLE_PTR AuditData);
```

This function captures biometric data from the attached device, and compares it against the StoredTemplate.

Boolean Result indicates whether verification was successful or not, and the FARAchieved is a FAR value indicating how closely the BIRs actually matched.

If the StoredTemplate contains a payload, the Payload may be returned upon successful verification.

Section 2.5.3.7:
Parameters: Payload (output/optional) - if the StoredTemplate contains a payload, it is returned in an allocated BioAPI_DATA structure if the FARAchieved satisfies the policy of the BSP.
Section 2.2.1.2:
#define BioAPI_PAYLOAD (0x00001000)
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 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 fail, an UNDECIDED conformity response is issued.

```
     </description>
     <!-- UUID of the BSP under test -->
     <input name="_moduleUuid"/>
     <!-- Major version number of the BSP under test -->
     <input name="_moduleVersionMajor"/>
     <!-- Minor version number of the BSP under test -->
     <input name="_moduleVersionMinor"/>
     <!-- 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"/>
      <!-- MaxFARRequested for BioSPI_VerifyMatch -->
      <input name="_maxFARRequested"/>
      <!-- MaxFRRRequested for BioSPI_VerifyMatch -->
      <input name="_maxFRRRequested"/>
      <!-- MaxFARPrecedence for BioSPI_VerifyMatch -->
      <input name="_farprecedence"/>
      <!-- Timeout for BioSPI_Verify -->
      <input name="_verifytimeout"/>
      <!-- Indicates whether the BSP claims support for the payload.-->
      <input name="_payloadSupported"/>
      <!-- PayloadPolicy -->
      <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_Verify">
         <input name="moduleUuid" var="_moduleUuid"/>
         <input name="moduleVersionMajor" var="_moduleVersionMajor"/>
         <input name="moduleVersionMinor" var="_moduleVersionMinor"/>
         <input name="inserttimeouttime" var="_inserttimeout"/>
         <input name="nosourcepresentsupported" var="_noSourcePresentSupported"/>
         <input name="sourcepresenttimeouttime" var="_sourcepresenttimeout"/>
         <input name="capturetimeouttime" var="_capturetimeout"/>
         <input name="maxFARRequested" var="_maxFARRequested"/>
         <input name="maxFRRRequested" var="_maxFRRRequested"/>
         <input name="farprecedence" var="_farprecedence"/>
         <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="be0a1330-d59e-11d8-9669-
0800200c9a66" function="BioSPI_ModuleEventHandler"/>
   </assertion>
   <activity name="BioSPI_Verify">
      <input name="moduleUuid"/>
      <input name="moduleVersionMajor"/>
      <input name="moduleVersionMinor"/>
      <input name="inserttimeouttime"/>
      <input name="nosourcepresentsupported"/>
      <input name="sourcepresenttimeouttime"/>
      <input name="capturetimeouttime"/>
      <input name="maxFARRequested"/>
      <input name="maxFRRRequested"/>
      <input name="farprecedence"/>
      <input name="verifytimeout"/>
      <input name="payloadSupported"/>
      <input name="payloadPolicy"/>
      <input name="payload"/>
      <!-- This assertion will use ModuleHandle "1" for all BioSPI calls that
require it -->
      <set name="_modulehandle" value="1"/>
      <!-- Invoke the functions BioSPI_ModuleLoad and BioSPI_ModuleAttach
exposed by the BSP under test.
            The input value for the parameter "deviceIDOrNull" is "0", therefore
the assertion will test the BSP's default device. -->
      <invoke activity="LoadAndAttach" package="be0a1330-d59e-11d8-9669-
0800200c9a66" break_on_break="true">
```

```
        <input name="moduleUuid" var="moduleUuid"/>
        <input name="moduleVersionMajor" var="moduleVersionMajor"/>
        <input name="moduleVersionMinor" var="moduleVersionMinor"/>
        <input name="deviceIDOrNull" value="0"/>
        <input name="module" var="_modulehandle"/>
        <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="ModuleHandle" var="_modulehandle"/>
        <input name="Purpose"
var="__BioAPI_PURPOSE_ENROLL_FOR_VERIFICATION_ONLY"/>
        <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 verify against the
stored template -->
     <invoke function="BioSPI_Verify">
        <input name="ModuleHandle" var="_modulehandle"/>
        <input name="MaxFARRequested" var="maxFARRequested"/>
        <input name="MaxFRRRequested" var="maxFRRRequested"/>
        <input name="FARPrecedence" var="farprecedence"/>
        <input name="StoredTemplate_Form" var="__BioAPI_BIR_HANDLE_INPUT"/>
        <input name="StoredTemplate_BIRHandle" var="template_handle"/>
        <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="FARAchieved" setvar="farAchieved"/>
        <output name="FRRAchieved" setvar="frrAchieved"/>
        <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="be0a1330-d59e-
11d8-9669-0800200c9a66" 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="farAchieved" var="farAchieved"/>
     </invoke>
     <!-- Check the value of the output parameter "payload" (BSP not claiming
to support the payload) -->
     <invoke activity="payloadNotSupport_checkPayload" package="be0a1330-d59e-
11d8-9669-0800200c9a66" 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_ModuleDetach and BioSPI_ModuleUnload -->
     <invoke activity="DetachAndUnload" package="be0a1330-d59e-11d8-9669-
0800200c9a66">
        <input name="moduleUuid" var="moduleUuid"/>
        <input name="module" var="_modulehandle"/>
     </invoke>
   </activity>
</package>
```

16.40 Feature 19f. Return of raw/audit data

16.40.1 BioAPI Specification References:
3.3.4.7

16.40.2 Assertion 19f.1 BioSPI_Verify_AuditData

16.40.2.1 Test Purpose:
To test the return of "raw" data if the BSP supports return of raw/audit data.

16.40.2.2 Test Scenario:

If the BSP supports the return of raw/audit data, call BioSPI_Verify with valid parameters, including AuditData.

16.40.2.3 Expected Results:
Return code BioAPI_OK and a valid AuditData.

16.40.2.4 XML:

```
<package name="51b2a530-31c9-11d9-9669-0800200c9a66">
   <author>
      author
   </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 1.1 is quoted below from subclauses 2.2.1.2,
4.2.4.2 and 2.5.3.7.
         ____________________________________________________
         BioAPI_RETURN BioAPI BioAPI_Verify
            (BioAPI_HANDLE ModuleHandle,
            const BioAPI_FAR *MaxFARRequested,
            const BioAPI_FRR *MaxFRRRequested,
            const BioAPI_BOOL *FARPrecedence,
            const BioAPI_INPUT_BIR *StoredTemplate,
            BioAPI_BIR_HANDLE_PTR AdaptedBIR,
            BioAPI_BOOL *Result,
            BioAPI_FAR_PTR FARAchieved,
            BioAPI_FRR_PTR FRRAchieved,
            BioAPI_DATA_PTR *Payload,
            sint32 Timeout,
            BioAPI_BIR_HANDLE_PTR AuditData);

         This function captures biometric data from the attached device, and
compares it against the StoredTemplate.

         Boolean Result indicates whether verification was successful or not, and
the FARAchieved is a FAR value indicating how closely the BIRs actually
matched.
         ____________________________________________________
         Section 2.5.3.7:
         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.
         Section 2.2.1.2: BioAPI_OPTIONS_MASK
         #define BioAPI_RAW (0x00000001)
         If set, indicates that the BSP supports the return of raw/audit data.
         Section 4.2.4.2:
         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.
        NOTE: BioAPI SPECIFICATION AMBIGUOUS
        If audit data is not supported, Section 2.5.3.7 and Section 4.2.4.2
suggest different return value for 'AuditData'.
        ____________________________________________________

        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 fail, an UNDECIDED conformity
response is issued.
     </description>

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

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

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

     <!-- 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"/>

     <!-- MaxFARRequested for BioSPI_VerifyMatch -->
     <input name="_maxFARRequested" />

     <!-- MaxFRRRequested for BioSPI_VerifyMatch -->
     <input name="_maxFRRRequested" />

     <!-- MaxFARPrecedence for BioSPI_VerifyMatch -->
     <input name="_farprecedence" />

     <!-- 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="moduleUuid" var="_moduleUuid"/>
        <input name="moduleVersionMajor" var="_moduleVersionMajor"/>
        <input name="moduleVersionMinor" var="_moduleVersionMinor"/>
        <input name="inserttimeouttime" var="_inserttimeout"/>
        <input name="nosourcepresentsupported"
             var="_noSourcePresentSupported" />
        <input name="sourcepresenttimeouttime"
             var="_sourcepresenttimeout"/>
        <input name="capturetimeouttime" var="_capturetimeout"/>
        <input name="maxFARRequested" var="_maxFARRequested"/>
        <input name="maxFRRRequested" var="_maxFRRRequested"/>
        <input name="farprecedence" var="_farprecedence" />
        <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="be0a1330-d59e-11d8-9669-0800200c9a66"
          function="BioSPI_ModuleEventHandler"/>
   </assertion>

   <activity name="BioSPI_Verify">
     <input name="moduleUuid"/>
     <input name="moduleVersionMajor"/>
     <input name="moduleVersionMinor"/>
     <input name="inserttimeouttime"/>
     <input name="nosourcepresentsupported" />
     <input name="sourcepresenttimeouttime"/>
     <input name="capturetimeouttime"/>
     <input name="maxFARRequested" />
     <input name="maxFRRRequested" />
     <input name="farprecedence" />
     <input name="verifytimeout" />
     <input name="supportAuditData" />

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

     <!-- Invoke the functions BioSPI_ModuleLoad and BioSPI_ModuleAttach
exposed by the BSP under test.
          The input value for the parameter "deviceIDOrNull" is "0", therefore
the assertion will test the BSP's default device. -->
     <invoke activity="LoadAndAttach"
          package="be0a1330-d59e-11d8-9669-0800200c9a66"
          break_on_break="true">
        <input name="moduleUuid" var="moduleUuid"/>
        <input name="moduleVersionMajor" var="moduleVersionMajor"/>
        <input name="moduleVersionMinor" var="moduleVersionMinor"/>
        <input name="deviceIDOrNull" value="0"/>
        <input name="module" var="_modulehandle"/>
        <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="ModuleHandle" var="_modulehandle"/>
        <input name="Purpose"
             var="__BioAPI_PURPOSE_ENROLL_FOR_VERIFICATION_ONLY"/>
        <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 verify against the
stored template -->
     <invoke function="BioSPI_Verify" >
        <input name="ModuleHandle" var="_modulehandle"/>
        <input name="MaxFARRequested" var="maxFARRequested" />
        <input name="MaxFRRRequested" var="maxFRRRequested" />
        <input name="FARPrecedence" var="farprecedence" />
        <input name="StoredTemplate_Form"
             var="__BioAPI_BIR_HANDLE_INPUT" />
        <input name="StoredTemplate_BIRHandle"
             var="template_handle" />
        <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="FARAchieved" setvar="farAchieved" />
```

```
        <output name="FRRAchieved" setvar="frrAchieved" />
        <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 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" />
        </only_if>
        <input name="ModuleHandle" var="_modulehandle"/>
        <input name="Handle" var="auditbir_handle"/>
     </invoke>

     <!-- Invoke the functions BioSPI_ModuleDetach and BioSPI_ModuleUnload -->
     <invoke activity="DetachAndUnload"
          package="be0a1330-d59e-11d8-9669-0800200c9a66" >
        <input name="moduleUuid" var="moduleUuid" />
        <input name="module" var="_modulehandle" />
     </invoke>
   </activity>

   <activity name="check_audit_data_type" >
     <input name="ModuleHandle" />
     <input name="Handle" />
```

```
     <!-- Invoke the function BioSPI_GetHeaderFromHandle. -->
     <invoke function="BioSPI_GetHeaderFromHandle">
        <input name="ModuleHandle" var="_modulehandle"/>
        <input name="Handle" var="Handle"/>
        <output name="Length" setvar="length"/>
        <output name="HeaderVersion" setvar="headerversion"/>
        <output name="ProcessedLevel" setvar="processedLevel"/>
        <output name="FormatOwner" setvar="formatowner"/>
        <output name="FormatId" setvar="formatid"/>
        <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, 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>
```

16.41 Feature 19h. BIR signing (by BSP)

16.41.1 BioAPI Specification References:
1.5, 2.1.7

16.41.2 Assertion 19h.1 BioSPI_Verify_BIRSignedTemplateAdaptation

16.41.2.1 Test Purpose:
To test BIR signing if both BIR signing and Model/template adaptation are supported by the BSP.

16.41.2.2 Test Scenario:
If the BSP supports both BIR signing and Model/template adaptation, call BioSPI_Verify with valid parameters, including AdaptedBIR. The presented biometric should "match" the StoredTemplate.

16.41.2.3 Expected Results:

Return code BioAPI_OK, Result of BioAPI_TRUE and a valid, signed AdaptedBIR.

16.41.2.4 XML:

16.42 Feature 19i. BIR encryption (by BSP)

16.42.1 BioAPI Specification References:
1.5, 2.1.7

16.42.2 Assertion 19i.1 BioSPI_Verify_BIREncryptedTemplateAdaptation

16.42.2.1 Test Purpose:
To test BIR encryption if both BIR encryption and Model/template adaptation are supported by the BSP.

16.42.2.2 Test Scenario:
If the BSP supports both BIR encryption and Model/template adaptation, call BioSPI_Verify with valid parameters, including AdaptedBIR. The presented biometric should "match" the StoredTemplate.

16.42.2.3 Expected Results:
Return code BioAPI_OK, Result of BioAPI_TRUE and a valid AdaptedBIR.

16.42.2.4 XML:

16.43 Feature 21. BioSPI Import

16.43.1 BioAPI Specification References:
3.3.4.9

16.43.2 Assertion 21.1 BioSPI_Import_ValidParam

16.43.2.1 Test Purpose:
To test BioSPI_Import with valid parameters.

16.43.2.2 Test Scenario:
Call BioSPI_Import with valid parameters.

16.43.2.3 Expected Results:
Return code of BioAPI_OK and a valid ConstructedBIR.

16.43.2.4 XML:

```
<package name="00769b48-09e9-1085-8059-0002a5d5fd2e">
   <author>Author</author>
   <description>
      This package contains the assertion "BioSPI_Import_ValidParam"
   </description>
   <assertion name="BioSPI_Import_ValidParam" model="BSPTesting">
      <description>
         This assertion invokes BioSPI_Import with valid input parameters and
```

verify if the return code is BioAPI_OK.

The relevant text in BioAPI 1.1 is quoted below from subclauses 2.5.3.9.

```
BioAPI_RETURN BioAPI BioSPI_Import
(BioAPI_HANDLE ModuleHandle,
const BioAPI_DATA *InputData,
BioAPI_BIR_BIOMETRIC_DATA_FORMAT InputFormat,
BioAPI_BIR_PURPOSE Purpose,
BioAPI_BIR_HANDLE_PTR ConstructedBIR);
```

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_Enroll to create a BIR.
4) Invoke function BioSPI_GetBIRFromHandle to get the BIR.
5) Invoke function BioSPI_Import and verify the return code.
6) Invoke function BioSPI_GetBIRFromHandle to verify the constructed BIR.
7) Detach and unload the BSP.

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

```
      </description>
      <!-- UUID of the BSP under test -->
      <input name="_moduleUuid"/>
      <!-- Major version number of the BSP under test -->
      <input name="_moduleVersionMajor"/>
      <!-- Minor version number of the BSP under test -->
      <input name="_moduleVersionMinor"/>
      <!-- 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 capture -->
      <input name="_capturetimeout"/>
      <!-- Purpose for import -->
      <input name="_purpose"/>
      <!-- Invocation of the primary activity of this assertion with input
parameter values assigned from the assertion's parameters. -->
      <invoke activity="BioSPI_Import">
         <input name="moduleUuid" var="_moduleUuid"/>
         <input name="moduleVersionMajor" var="_moduleVersionMajor"/>
         <input name="moduleVersionMinor" var="_moduleVersionMinor"/>
         <input name="inserttimeouttime" var="_inserttimeout"/>
         <input name="nosourcepresentsupported" var="_noSourcePresentSupported"/>
         <input name="sourcepresenttimeouttime" var="_sourcepresenttimeout"/>
         <input name="capturetimeout" var="_capturetimeout"/>
         <input name="purpose" var="_purpose"/>
      </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="be0a1330-d59e-11d8-9669-
0800200c9a66" function="BioSPI_ModuleEventHandler"/>
   </assertion>
   <activity name="BioSPI_Import">
      <input name="moduleUuid"/>
      <input name="moduleVersionMajor"/>
```

```
     <input name="moduleVersionMinor"/>
     <input name="inserttimeouttime"/>
     <input name="nosourcepresentsupported"/>
     <input name="sourcepresenttimeouttime"/>
     <input name="capturetimeout"/>
     <input name="purpose"/>
     <!-- This assertion will use ModuleHandle "1" for all BioSPI calls that
require it -->
     <set name="_modulehandle" value="1"/>
     <!-- Invoke the functions BioSPI_ModuleLoad and BioSPI_ModuleAttach
exposed by the BSP under test.
          The input value for the parameter "deviceIDOrNull" is "0", therefore
the assertion will test the BSP's default device. -->
     <invoke activity="LoadAndAttach" package="be0a1330-d59e-11d8-9669-
0800200c9a66" break_on_break="true">
        <input name="moduleUuid" var="moduleUuid"/>
        <input name="moduleVersionMajor" var="moduleVersionMajor"/>
        <input name="moduleVersionMinor" var="moduleVersionMinor"/>
        <input name="deviceIDOrNull" value="0"/>
        <input name="module" var="_modulehandle"/>
        <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="ModuleHandle" var="_modulehandle"/>
        <input name="Purpose"
var="__BioAPI_PURPOSE_ENROLL_FOR_VERIFICATION_ONLY"/>
        <input name="Timeout" var="capturetimeout"/>
        <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="ModuleHandle" var="_modulehandle"/>
        <input name="Handle" var="newtemplate_handle"/>
        <output name="Length" setvar="length"/>
        <output name="HeaderVersion" setvar="headerversion"/>
        <output name="ProcessedLevel" setvar="processedlevel"/>
        <output name="FormatOwner" setvar="formatowner"/>
        <output name="FormatId" setvar="formatid"/>
        <output name="Quality" setvar="quality"/>
        <output name="Purpose" setvar="purpose"/>
        <output name="BiometricData" setvar="biometricdata"/>
        <output name="Signature" setvar="signature"/>
        <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_GetBIRFromHandle has returned BioAPI_OK.
        </description>
        <equal_to var1="return" var2="__BioAPI_OK"/>
     </assert_condition>
     <!-- Invoke the function BioSPI_Import -->
     <invoke function="BioSPI_Import">
        <input name="ModuleHandle" var="_modulehandle"/>
        <input name="InputData" var="biometricdata"/>
        <input name="InputFormatOwner" var="formatowner"/>
        <input name="InputFormatId" var="formatid"/>
        <input name="Purpose" var="purpose"/>
        <output name="ConstructedBIR" setvar="constructedBIR_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_Import has returned BioAPI_OK.
        </description>
        <equal_to var1="return" var2="__BioAPI_OK"/>
     </assert_condition>
     <!-- Invoke function BioSPI_GetBIRFromHandle to verify the constructed BIR
is valid -->
     <invoke function="BioSPI_GetBIRFromHandle">
        <input name="ModuleHandle" var="_modulehandle"/>
        <input name="Handle" var="constructedBIR_handle"/>
        <output name="Length" setvar="length"/>
        <output name="HeaderVersion" setvar="headerversion"/>
        <output name="ProcessedLevel" setvar="processedlevel"/>
        <output name="FormatOwner" setvar="formatowner"/>
        <output name="FormatId" setvar="formatid"/>
        <output name="Quality" setvar="quality"/>
        <output name="Purpose" setvar="purpose"/>
        <output name="BiometricData" setvar="biometricdata"/>
        <output name="Signature" setvar="signature"/>
        <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_GetBIRFromHandle 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="be0a1330-d59e-11d8-9669-
0800200c9a66">
        <input name="moduleUuid" var="moduleUuid"/>
        <input name="module" var="_modulehandle"/>
     </invoke>
   </activity>
</package>
```

16.43.3 Assertion 21.2 BioSPI_Import_InvalidModuleHandle

16.43.3.1 Test Purpose:
To test BioSPI_Import with an invalid module handle.

16.43.3.2 Test Scenario:
Call BioSPI_Import with an invalid module handle.

16.43.3.3 Expected Results:
Return code other than BioAPI_OK and an invalid ConstructedBIR.

16.43.3.4 XML:

```
<package name="04c0a4f0-0a35-1085-bf8a-0002a5d5fd2e">
   <author>Author</author>
   <description>
      This package contains the assertion "BioSPI_Import_InvalidModuleHandle"
   </description>
   <assertion name="BioSPI_Import_InvalidModuleHandle" model="BSPTesting">
      <description>
         This assertion invokes BioSPI_Import with invalid module handle and
verify if the return code is BioAPIERR_H_FRAMEWORK_INVALID_MODULE_HANDLE.
         The relevant text in BioAPI 1.1 is quoted below from subclauses 2.5.3.9.
         ____________________________________________________________
         BioAPI_RETURN BioAPI BioSPI_Import
         (BioAPI_HANDLE ModuleHandle,
         const BioAPI_DATA *InputData,
         BioAPI_BIR_BIOMETRIC_DATA_FORMAT InputFormat,
         BioAPI_BIR_PURPOSE Purpose,
         BioAPI_BIR_HANDLE_PTR ConstructedBIR);
         ____________________________________________________________
         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_Enroll to create a BIR.
            4) Invoke function BioSPI_GetBIRFromHandle to get the BIR.
            5) Invoke function BioSPI_Import with an invalid module handle and
verify the return code.
            6) Detach and unload the BSP.

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

```
     <!-- UUID of the BSP under test -->
     <input name="_moduleUuid"/>
     <!-- Major version number of the BSP under test -->
     <input name="_moduleVersionMajor"/>
     <!-- Minor version number of the BSP under test -->
     <input name="_moduleVersionMinor"/>
     <!-- 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 capture -->
     <input name="_capturetimeout"/>
     <!-- Purpose for import -->
     <input name="_purpose"/>
     <!-- Invocation of the primary activity of this assertion with input
parameter values assigned from the assertion's parameters. -->
     <invoke activity="BioSPI_Import">
        <input name="moduleUuid" var="_moduleUuid"/>
        <input name="moduleVersionMajor" var="_moduleVersionMajor"/>
        <input name="moduleVersionMinor" var="_moduleVersionMinor"/>
        <input name="inserttimeouttime" var="_inserttimeout"/>
        <input name="nosourcepresentsupported" var="_noSourcePresentSupported"/>
        <input name="sourcepresenttimeouttime" var="_sourcepresenttimeout"/>
        <input name="capturetimeout" var="_capturetimeout"/>
        <input name="purpose" var="_purpose"/>
     </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="be0a1330-d59e-11d8-9669-
0800200c9a66" function="BioSPI_ModuleEventHandler"/>
   </assertion>
   <activity name="BioSPI_Import">
     <input name="moduleUuid"/>
     <input name="moduleVersionMajor"/>
     <input name="moduleVersionMinor"/>
     <input name="inserttimeouttime"/>
     <input name="nosourcepresentsupported"/>
     <input name="sourcepresenttimeouttime"/>
     <input name="capturetimeout"/>
     <input name="purpose"/>
     <!-- This assertion will use ModuleHandle "1" for all BioSPI calls that
require it -->
     <set name="_modulehandle" value="1"/>
     <!-- Invoke the functions BioSPI_ModuleLoad and BioSPI_ModuleAttach
exposed by the BSP under test.
          The input value for the parameter "deviceIDOrNull" is "0", therefore
the assertion will test the BSP's default device. -->
     <invoke activity="LoadAndAttach" package="be0a1330-d59e-11d8-9669-
0800200c9a66" break_on_break="true">
        <input name="moduleUuid" var="moduleUuid"/>
        <input name="moduleVersionMajor" var="moduleVersionMajor"/>
        <input name="moduleVersionMinor" var="moduleVersionMinor"/>
        <input name="deviceIDOrNull" value="0"/>
        <input name="module" var="_modulehandle"/>
        <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="ModuleHandle" var="_modulehandle"/>
        <input name="Purpose"
var="__BioAPI_PURPOSE_ENROLL_FOR_VERIFICATION_ONLY"/>
        <input name="Timeout" var="capturetimeout"/>
        <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="ModuleHandle" var="_modulehandle"/>
        <input name="Handle" var="newtemplate_handle"/>
        <output name="Length" setvar="length"/>
        <output name="HeaderVersion" setvar="headerversion"/>
        <output name="ProcessedLevel" setvar="processedlevel"/>
        <output name="FormatOwner" setvar="formatowner"/>
        <output name="FormatId" setvar="formatid"/>
        <output name="Quality" setvar="quality"/>
        <output name="Purpose" setvar="purpose"/>
        <output name="BiometricData" setvar="biometricdata"/>
        <output name="Signature" setvar="signature"/>
        <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_GetBIRFromHandle has returned BioAPI_OK.
        </description>
        <equal_to var1="return" var2="__BioAPI_OK"/>
     </assert_condition>
```

```
     <!-- Invoke the function BioSPI_Import -->
     <invoke function="BioSPI_Import">
        <input name="ModuleHandle" value="0"/>
        <input name="InputData" var="biometricdata"/>
        <input name="InputFormatOwner" var="formatowner"/>
        <input name="InputFormatId" var="formatid"/>
        <input name="Purpose" var="purpose"/>
        <output name="ConstructedBIR" setvar="constructedBIR_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_Import has returned
BioAPIERR_H_FRAMEWORK_INVALID_MODULE_HANDLE.
        </description>
        <equal_to var1="return"
var2="__BioAPIERR_H_FRAMEWORK_INVALID_MODULE_HANDLE "/>
     </assert_condition>
     <!-- Invoke the functions BioSPI_ModuleDetach and BioSPI_ModuleUnload -->
     <invoke activity="DetachAndUnload" package="be0a1330-d59e-11d8-9669-
0800200c9a66">
        <input name="moduleUuid" var="moduleUuid"/>
        <input name="module" var="_modulehandle"/>
     </invoke>
   </activity>
</package>
```

16.43.4 Assertion 21.3 BioSPI_Import_InvalidInputData

16.43.4.1 Test Purpose:
To test BioSPI_Import with NULL input data.

16.43.4.2 Test Scenario:
Call BioSPI_Import with NULL input data.

16.43.4.3 Expected Results:
Return code other than BioAPI_OK.

16.43.4.4 XML:

16.43.5 Assertion 21.4 BioSPI_Import_InvalidPurpose

16.43.5.1 Test Purpose:
To test BioSPI_Import with an invalid purpose.

16.43.5.2 Test Scenario:
Call BioSPI_Import with an invalid purpose.

16.43.5.3 Expected Results:
Return code other than BioAPI_OK.

16.43.5.4 XML:

16.44 Feature 23. BioSPI Db Open

16.44.1 BioAPI Specification References:
3.3.5.1

16.44.2 Assertion 23.1 BioSPI_DbOpen_ValidParam

16.44.2.1 Test Purpose:
To test BioSPI_DbOpen with valid parameters if supported by the BSP.

16.44.2.2 Test Scenario:
If supported by the BSP, call BioSPI_DbOpen with valid parameters.

16.44.2.3 Expected Results:
Return code BioAPI_OK, a valid DbHandle and a valid Cursor.

16.44.2.4 XML:

```
<package name="03caee98-08ef-1085-80f2-0002a5d5fd2e">
   <author>Author</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
verify if the return code is BioAPI_OK.
         The relevant text in BioAPI 1.1 is quoted below from subclauses 2.5.4.1.
         ____________________________________________________________
         BioAPI_RETURN BioAPI BioAPI_DbOpen
            (BioAPI_HANDLE ModuleHandle,
            const uint8 *DbName,
            BioAPI_DB_ACCESS_TYPE AccessRequest,
            BioAPI_DB_HANDLE_PTR DbHandle,
            BioAPI_DB_CURSOR_PTR Cursor);

         This function opens the data store with the specified name under the
specified access mode. A database Cursor is set to point to the first record in
the database. Note: the default database (if any) is always open.

         ____________________________________________________________
         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 fail, an UNDECIDED conformity
response is issued.
      </description>
      <!-- UUID of the BSP under test -->
      <input name="_moduleUuid"/>
      <!-- Major version number of the BSP under test -->
      <input name="_moduleVersionMajor"/>
      <!-- Minor version number of the BSP under test -->
      <input name="_moduleVersionMinor"/>
      <!-- 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="_dbName"/>
      <!-- Read Access Request to the database -->
      <input name="_readAccessRequest"/>
      <!-- Write Access Request -->
      <input name="_writeAccessRequest"/>
      <!-- Indicates if the BSP support DB creation-->
      <input name="_supportDBCreation"/>
      <!-- Invocation of the primary activity of this assertion with input
parameter values assigned from the assertion's parameters. -->
      <invoke activity="BioSPI_DbOpen">
         <input name="moduleUuid" var="_moduleUuid"/>
         <input name="moduleVersionMajor" var="_moduleVersionMajor"/>
         <input name="moduleVersionMinor" var="_moduleVersionMinor"/>
         <input name="inserttimeouttime" var="_inserttimeout"/>
         <input name="nosourcepresentsupported" var="_noSourcePresentSupported"/>
         <input name="sourcepresenttimeouttime" var="_sourcepresenttimeout"/>
         <input name="dbName" var="_dbName"/>
         <input name="readAccessRequest" var="_readAccessRequest"/>
         <input name="writeAccessRequest" var="_writeAccessRequest"/>
         <input name="supportDBCreation" var="_supportDBCreation"/>
      </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="be0a1330-d59e-11d8-9669-
0800200c9a66" function="BioSPI_ModuleEventHandler"/>
   </assertion>
   <activity name="BioSPI_DbOpen">
      <input name="moduleUuid"/>
      <input name="moduleVersionMajor"/>
      <input name="moduleVersionMinor"/>
      <input name="inserttimeouttime"/>
      <input name="nosourcepresentsupported"/>
      <input name="sourcepresenttimeouttime"/>
      <input name="dbName"/>
      <input name="readAccessRequest"/>
      <input name="writeAccessRequest"/>
      <input name="supportDBCreation"/>
      <!-- This assertion will use ModuleHandle "1" for all BioSPI calls that
require it -->
      <set name="_modulehandle" value="1"/>
      <!-- Invoke the functions BioSPI_ModuleLoad and BioSPI_ModuleAttach
exposed by the BSP under test.
            The input value for the parameter "deviceIDOrNull" is "0", therefore
the assertion will test the BSP's default device. -->
      <invoke activity="LoadAndAttach" package="be0a1330-d59e-11d8-9669-
0800200c9a66" break_on_break="true">
         <input name="moduleUuid" var="moduleUuid"/>
         <input name="moduleVersionMajor" var="moduleVersionMajor"/>
         <input name="moduleVersionMinor" var="moduleVersionMinor"/>
         <input name="deviceIDOrNull" value="0"/>
         <input name="module" var="_modulehandle"/>
         <input name="eventtimeouttime" var="inserttimeouttime"/>
      </invoke>
      <invoke activity="PrepareDBTesting" package="be0a1330-d59e-11d8-9669-
0800200c9a66" break_on_break="true">
```

```
        <only_if>
           <same_as var1="supportDBCreation" value2="true"/>
        </only_if>
        <input name="ModuleHandle" var="_modulehandle"/>
        <input name="dbName" var="dbName"/>
        <input name="nosourcepresentsupported" var="nosourcepresentsupported"/>
     </invoke>
     <!-- Invoke the function BioSPI_DbOpen to open the specified database. -->
     <invoke function="BioSPI_DbOpen">
        <input name="ModuleHandle" var="_modulehandle"/>
        <input name="DbName" var="dbName"/>
        <input name="ReadAccessRequest" var="readAccessRequest"/>
        <input name="WriteAccessRequest" var="writeAccessRequest"/>
        <output name="DbHandle" setvar="dbHandle"/>
        <output name="Cursor" setvar="cursor"/>
        <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="ModuleHandle" var="_modulehandle"/>
        <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="be0a1330-d59e-11d8-9669-
0800200c9a66" break_on_break="true">
        <only_if>
           <same_as var1="supportDBCreation" value2="true"/>
        </only_if>
        <input name="ModuleHandle" var="_modulehandle"/>
        <input name="dbName" var="dbName"/>
     </invoke>
     <!-- Invoke the functions BioSPI_ModuleDetach and BioSPI_ModuleUnload -->
     <invoke activity="DetachAndUnload" package="be0a1330-d59e-11d8-9669-
0800200c9a66">
        <input name="moduleUuid" var="moduleUuid"/>
        <input name="module" var="_modulehandle"/>
     </invoke>
   </activity>
</package>
```

16.44.3 Assertion 23.2 BioSPI_DbOpen_InvalidModuleHandle

16.44.3.1 Test Purpose:
To test BioSPI_DbOpen with an invalid module handle if supported by the BSP.

16.44.3.2 Test Scenario:
If supported by the BSP, call BioSPI_DbOpen with an invalid module handle.

16.44.3.3 Expected Results:
Return code other than BioAPI_OK.

16.44.3.4 XML:

```
<package name="0373dd88-08f7-1085-ab39-0002a5d5fd2e">
   <author>Author</author>
   <description>
      This package contains the assertion "BioSPI_DbOpen_InvalidModuleHandle"
   </description>
   <assertion name="BioSPI_DbOpen_InvalidModuleHandle" model="BSPTesting">
      <description>
         This assertion invokes BioSPI_DbOpen with invalid module handle and
verify if the return code is BioAPIERR_H_FRAMEWORK_INVALID_MODULE_HANDLE.
         The relevant text in BioAPI 1.1 is quoted below from subclauses 2.5.4.1.
         ____________________________________________________________
         BioAPI_RETURN BioAPI BioAPI_DbOpen
            (BioAPI_HANDLE ModuleHandle,
            const uint8 *DbName,
            BioAPI_DB_ACCESS_TYPE AccessRequest,
            BioAPI_DB_HANDLE_PTR DbHandle,
            BioAPI_DB_CURSOR_PTR Cursor);

         This function opens the data store with the specified name under the
specified access mode. A database Cursor is set to point to the first record in
the database. Note: the default database (if any) is always open.

         ____________________________________________________________
         and text from subclause 2.3.2.3
         ____________________________________________________________
         #define BioAPIERR_H_FRAMEWORK_INVALID_MODULE_HANDLE \
                       (BioAPI_H_FRAMEWORK_BASE_ERROR +
BioAPI_ERRORCODE_COMMON_EXTENT + 1)
         The given service provider handle is not valid
         ____________________________________________________________

         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 module handle.
            4) Verify the return code, it is expected to be
BioAPIERR_H_FRAMEWORK_INVALID_MODULE_HANDLE.
            5) Detach and unload the BSP.

         If any of the intermediate operations fail, an UNDECIDED conformity
response is issued.
      </description>
      <!-- UUID of the BSP under test -->
      <input name="_moduleUuid"/>
      <!-- Major version number of the BSP under test -->
      <input name="_moduleVersionMajor"/>
      <!-- Minor version number of the BSP under test -->
```

```
      <input name="_moduleVersionMinor"/>
      <!-- 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="_dbName"/>
      <!-- Read Access Request to the database -->
      <input name="_readAccessRequest"/>
      <!-- Write Access Request -->
      <input name="_writeAccessRequest"/>
      <!-- Indicates if the BSP support DB creation-->
      <input name="_supportDBCreation"/>
      <!-- Invocation of the primary activity of this assertion with input
parameter values assigned from the assertion's parameters. -->
      <invoke activity="BioSPI_DbOpen">
         <input name="moduleUuid" var="_moduleUuid"/>
         <input name="moduleVersionMajor" var="_moduleVersionMajor"/>
         <input name="moduleVersionMinor" var="_moduleVersionMinor"/>
         <input name="inserttimeouttime" var="_inserttimeout"/>
         <input name="nosourcepresentsupported" var="_noSourcePresentSupported"/>
         <input name="sourcepresenttimeouttime" var="_sourcepresenttimeout"/>
         <input name="dbName" var="_dbName"/>
         <input name="readAccessRequest" var="_readAccessRequest"/>
         <input name="writeAccessRequest" var="_writeAccessRequest"/>
         <input name="supportDBCreation" var="_supportDBCreation"/>
      </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="be0a1330-d59e-11d8-9669-
0800200c9a66" function="BioSPI_ModuleEventHandler"/>
   </assertion>
   <activity name="BioSPI_DbOpen">
      <input name="moduleUuid"/>
      <input name="moduleVersionMajor"/>
      <input name="moduleVersionMinor"/>
      <input name="inserttimeouttime"/>
      <input name="nosourcepresentsupported"/>
      <input name="sourcepresenttimeouttime"/>
      <input name="dbName"/>
      <input name="readAccessRequest"/>
      <input name="writeAccessRequest"/>
      <input name="supportDBCreation"/>
      <!-- This assertion will use ModuleHandle "1" for all BioSPI calls that
require it -->
      <set name="_modulehandle" value="1"/>
      <!-- Invoke the functions BioSPI_ModuleLoad and BioSPI_ModuleAttach
exposed by the BSP under test.
           The input value for the parameter "deviceIDOrNull" is "0", therefore
the assertion will test the BSP's default device. -->
      <invoke activity="LoadAndAttach" package="be0a1330-d59e-11d8-9669-
0800200c9a66" break_on_break="true">
         <input name="moduleUuid" var="moduleUuid"/>
         <input name="moduleVersionMajor" var="moduleVersionMajor"/>
         <input name="moduleVersionMinor" var="moduleVersionMinor"/>
         <input name="deviceIDOrNull" value="0"/>
         <input name="module" var="_modulehandle"/>
         <input name="eventtimeouttime" var="inserttimeouttime"/>
      </invoke>
```

```
     <invoke activity="PrepareDBTesting" package="be0a1330-d59e-11d8-9669-
0800200c9a66" break_on_break="true">
        <only_if>
           <same_as var1="supportDBCreation" value2="true"/>
        </only_if>
        <input name="ModuleHandle" var="_modulehandle"/>
        <input name="dbName" var="dbName"/>
        <input name="nosourcepresentsupported" var="nosourcepresentsupported"/>
     </invoke>
     <!-- Invoke the function BioSPI_DbOpen with invalid module handle. -->
     <invoke function="BioSPI_DbOpen">
        <input name="ModuleHandle" value="0"/>
        <input name="DbName" var="dbName"/>
        <input name="ReadAccessRequest" var="readAccessRequest"/>
        <input name="WriteAccessRequest" var="writeAccessRequest"/>
        <output name="DbHandle" setvar="dbHandle"/>
        <output name="Cursor" setvar="cursor"/>
        <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_H_FRAMEWORK_INVALID_MODULE_HANDLE.
        </description>
        <equal_to var1="return"
var2="__BioAPIERR_H_FRAMEWORK_INVALID_MODULE_HANDLE"/>
     </assert_condition>
     <invoke activity="CleanUpDBTesting" package="be0a1330-d59e-11d8-9669-
0800200c9a66" break_on_break="true">
        <only_if>
           <same_as var1="supportDBCreation" value2="true"/>
        </only_if>
        <input name="ModuleHandle" var="_modulehandle"/>
        <input name="dbName" var="dbName"/>
     </invoke>
     <!-- Invoke the functions BioSPI_ModuleDetach and BioSPI_ModuleUnload -->
     <invoke activity="DetachAndUnload" package="be0a1330-d59e-11d8-9669-
0800200c9a66">
        <input name="moduleUuid" var="moduleUuid"/>
        <input name="module" var="_modulehandle"/>
     </invoke>
   </activity>
</package>
```

16.44.4 Assertion 23.3 BioSPI_DbOpen_InvalidDBName

16.44.4.1 Test Purpose:
To test BioSPI_DbOpen with an invalid DB name if supported by the BSP.

16.44.4.2 Test Scenario:
Call BioSPI_DbOpen with an invalid DB name.

16.44.4.3 Expected Results:
Return code other than BioAPI_OK.

16.44.4.4 XML:

16.44.5 Assertion 23.4 BioSPI_DbOpen_InvalidRequest

16.44.5.1 Test Purpose:
To test BioSPI_DbOpen with an invalid request if supported by the BSP.

16.44.5.2 Test Scenario:
Call BioSPI_DbOpen with an invalid request.

16.44.5.3 Expected Results:
Return code other than BioAPI_OK.

16.44.5.4 XML:

16.45 Feature 24. BioSPI Db Close

16.45.1 BioAPI Specification References:
3.3.5.2

16.45.2 Assertion 24.1 BioSPI_DbClose_ValidParam

16.45.2.1 Test Purpose:
To test BioSPI_DbClose with valid parameters if supported by the BSP.

16.45.2.2 Test Scenario:
If supported by the BSP, call BioSPI_DbClose with valid parameters.

16.45.2.3 Expected Results:
Return code BioAPI_OK.

16.45.2.4 XML:

```
<package name="02a56228-0900-1085-80f7-0002a5d5fd2e">
   <author>Author</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
verify if the return code is BioAPI_OK.
         The relevant text in BioAPI 1.1 is quoted below from subclauses 2.5.4.2.
         ____________________________________________________________
         BioAPI_RETURN BioAPI BioAPI_DbClose
            (BioAPI_HANDLE ModuleHandle,
            BioAPI_DB_HANDLE DbHandle);

         This function closes an open 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) Invoke function BioSPI_DbOpen to open the specified database.
            4) Invoke function BioSPI_DbClose with valid input parameters. Verify
```

```
the return code.
          5) Detach and unload the BSP.

        If any of the intermediate operations fail, an UNDECIDED conformity
response is issued.
     </description>
     <!-- UUID of the BSP under test -->
     <input name="_moduleUuid"/>
     <!-- Major version number of the BSP under test -->
     <input name="_moduleVersionMajor"/>
     <!-- Minor version number of the BSP under test -->
     <input name="_moduleVersionMinor"/>
     <!-- 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="_dbName"/>
     <!-- Indicates if the BSP support DB creation-->
     <input name="_supportDBCreation"/>
     <!-- Invocation of the primary activity of this assertion with input
parameter values assigned from the assertion's parameters. -->
     <invoke activity="BioSPI_DbClose">
        <input name="moduleUuid" var="_moduleUuid"/>
        <input name="moduleVersionMajor" var="_moduleVersionMajor"/>
        <input name="moduleVersionMinor" var="_moduleVersionMinor"/>
        <input name="inserttimeouttime" var="_inserttimeout"/>
        <input name="nosourcepresentsupported" var="_noSourcePresentSupported"/>
        <input name="sourcepresenttimeouttime" var="_sourcepresenttimeout"/>
        <input name="dbName" var="_dbName"/>
        <input name="supportDBCreation" var="_supportDBCreation"/>
     </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="be0a1330-d59e-11d8-9669-
0800200c9a66" function="BioSPI_ModuleEventHandler"/>
   </assertion>
   <activity name="BioSPI_DbClose">
     <input name="moduleUuid"/>
     <input name="moduleVersionMajor"/>
     <input name="moduleVersionMinor"/>
     <input name="inserttimeouttime"/>
     <input name="nosourcepresentsupported"/>
     <input name="sourcepresenttimeouttime"/>
     <input name="dbName"/>
     <input name="supportDBCreation"/>
     <!-- This assertion will use ModuleHandle "1" for all BioSPI calls that
require it -->
     <set name="_modulehandle" value="1"/>
     <!-- Invoke the functions BioSPI_ModuleLoad and BioSPI_ModuleAttach
exposed by the BSP under test.
          The input value for the parameter "deviceIDOrNull" is "0", therefore
the assertion will test the BSP's default device. -->
     <invoke activity="LoadAndAttach" package="be0a1330-d59e-11d8-9669-
0800200c9a66" break_on_break="true">
        <input name="moduleUuid" var="moduleUuid"/>
        <input name="moduleVersionMajor" var="moduleVersionMajor"/>
        <input name="moduleVersionMinor" var="moduleVersionMinor"/>
        <input name="deviceIDOrNull" value="0"/>
```

```
        <input name="module" var="_modulehandle"/>
        <input name="eventtimeouttime" var="inserttimeouttime"/>
     </invoke>
     <!-- If the BSP supports database creation, invoke BioSPI_DbDelete, in
preparation for the database creation -->
     <invoke function="BioSPI_DbDelete">
        <only_if>
           <same_as var1="supportDBCreation" value2="true"/>
        </only_if>
        <input name="ModuleHandle" var="_modulehandle"/>
        <input name="DbName" var="dbName"/>
        <return setvar="return"/>
     </invoke>
     <!-- Invoke BioSPI_DbCreate if the BSP supports database creation -->
     <invoke function="BioSPI_DbCreate">
        <only_if>
           <same_as var1="supportDBCreation" value2="true"/>
        </only_if>
        <input name="ModuleHandle" var="_modulehandle"/>
        <input name="DbName" var="dbName"/>
        <input name="ReadAccessRequest" value="true"/>
        <input name="WriteAccessRequest" value="true"/>
        <output name="DbHandle" setvar="dbHandle"/>
        <return setvar="return"/>
     </invoke>
     <!-- Open the specified database if database creation is not supported. --
>
     <invoke function="BioSPI_DbOpen">
        <only_if>
           <same_as var1="supportDBCreation" value2="false"/>
        </only_if>
        <input name="ModuleHandle" var="_modulehandle"/>
        <input name="DbName" var="dbName"/>
        <input name="ReadAccessRequest" value="true"/>
        <input name="WriteAccessRequest" value="true"/>
        <output name="DbHandle" setvar="dbHandle"/>
        <output name="Cursor" setvar="cursor"/>
        <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>
           If database creation is supported by the BSP, the function
BioSPI_DbCreate has returned BioAPI_OK and the output dbHandle is a valid DB
handle. If database creation is not supported by the BSP, 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>
     <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="ModuleHandle" var="_modulehandle"/>
        <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="ModuleHandle" var="_modulehandle"/>
        <input name="BIRToStore_Form" var="__BioAPI_BIR_HANDLE_INPUT"/>
        <input name="BIRToStore_BIRHandle" var="newtemplate_handle"/>
        <input name="DbHandle" var="dbHandle"/>
        <input name="no_Uuid" value="false"/>
        <output name="Uuid" 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.
        </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="ModuleHandle" var="_modulehandle"/>
        <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="be0a1330-d59e-11d8-9669-
0800200c9a66" break_on_break="true">
        <only_if>
           <same_as var1="supportDBCreation" value2="true"/>
        </only_if>
        <input name="ModuleHandle" var="_modulehandle"/>
        <input name="dbName" var="dbName"/>
     </invoke>
     <!-- Invoke the functions BioSPI_ModuleDetach and BioSPI_ModuleUnload -->
     <invoke activity="DetachAndUnload" package="be0a1330-d59e-11d8-9669-
0800200c9a66">
        <input name="moduleUuid" var="moduleUuid"/>
        <input name="module" var="_modulehandle"/>
     </invoke>
   </activity>
</package>
```

16.45.3 Assertion 24.2 BioSPI_DbClose_InvalidModuleHandle

16.45.3.1 Test Purpose:
To test BioSPI_DbClose with an invalid module handle if supported by the BSP.

16.45.3.2 Test Scenario:
If supported by the BSP, call BioSPI_DbClose with an invalid module handle.

16.45.3.3 Expected Results:
Return code other than BioAPI_OK.

16.45.3.4 XML:

```
<package name="0033e140-0907-1085-adb5-0002a5d5fd2e">
   <author>Author</author>
   <description>
      This package contains the assertion "BioSPI_DbClose_InvalidModuleHandle"
   </description>
   <assertion name="BioSPI_DbClose_InvalidModuleHandle" model="BSPTesting">
      <description>
         This assertion invokes BioSPI_DbClose with invalid module handle and
verify if the return code is BioAPIERR_H_FRAMEWORK_INVALID_MODULE_HANDLE.
         The relevant text in BioAPI 1.1 is quoted below from
         subclauses 2.5.4.2.
         ____________________________________________________________
         BioAPI_RETURN BioAPI BioAPI_DbClose
            (BioAPI_HANDLE ModuleHandle,
            BioAPI_DB_HANDLE DbHandle);

         This function closes an open database.
         ____________________________________________________________
         and text from subclause 2.3.2.3
         ____________________________________________________________
         #define BioAPIERR_H_FRAMEWORK_INVALID_MODULE_HANDLE \
                        (BioAPI_H_FRAMEWORK_BASE_ERROR +
BioAPI_ERRORCODE_COMMON_EXTENT + 1)
         The given service provider handle is not valid
         ____________________________________________________________

         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) Invoke function BioSPI_DbClose with invalid module handle. Verify
the return code.
          5) Detach and unload the BSP.

        If any of the intermediate operations fail, an UNDECIDED conformity
response is issued.
     </description>
     <!-- UUID of the BSP under test -->
     <input name="_moduleUuid"/>
     <!-- Major version number of the BSP under test -->
     <input name="_moduleVersionMajor"/>
     <!-- Minor version number of the BSP under test -->
     <input name="_moduleVersionMinor"/>
     <!-- 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="_dbName"/>
     <!-- Indicates if the BSP support DB creation-->
     <input name="_supportDBCreation"/>
     <!-- Invocation of the primary activity of this assertion with input
parameter values assigned from the assertion's parameters. -->
     <invoke activity="BioSPI_DbClose">
        <input name="moduleUuid" var="_moduleUuid"/>
        <input name="moduleVersionMajor" var="_moduleVersionMajor"/>
        <input name="moduleVersionMinor" var="_moduleVersionMinor"/>
        <input name="inserttimeouttime" var="_inserttimeout"/>
        <input name="nosourcepresentsupported" var="_noSourcePresentSupported"/>
        <input name="sourcepresenttimeouttime" var="_sourcepresenttimeout"/>
        <input name="dbName" var="_dbName"/>
        <input name="supportDBCreation" var="_supportDBCreation"/>
     </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="be0a1330-d59e-11d8-9669-
0800200c9a66" function="BioSPI_ModuleEventHandler"/>
   </assertion>
   <activity name="BioSPI_DbClose">
     <input name="moduleUuid"/>
     <input name="moduleVersionMajor"/>
     <input name="moduleVersionMinor"/>
     <input name="inserttimeouttime"/>
     <input name="nosourcepresentsupported"/>
     <input name="sourcepresenttimeouttime"/>
     <input name="dbName"/>
     <input name="supportDBCreation"/>
     <!-- This assertion will use ModuleHandle "1" for all BioSPI calls that
require it -->
     <set name="_modulehandle" value="1"/>
     <!-- Invoke the functions BioSPI_ModuleLoad and BioSPI_ModuleAttach
exposed by the BSP under test.
          The input value for the parameter "deviceIDOrNull" is "0", therefore
the assertion will test the BSP's default device. -->
```

```
     <invoke activity="LoadAndAttach" package="be0a1330-d59e-11d8-9669-
0800200c9a66" break_on_break="true">
        <input name="moduleUuid" var="moduleUuid"/>
        <input name="moduleVersionMajor" var="moduleVersionMajor"/>
        <input name="moduleVersionMinor" var="moduleVersionMinor"/>
        <input name="deviceIDOrNull" value="0"/>
        <input name="module" var="_modulehandle"/>
        <input name="eventtimeouttime" var="inserttimeouttime"/>
     </invoke>
     <invoke activity="PrepareDBTesting" package="be0a1330-d59e-11d8-9669-
0800200c9a66" break_on_break="true">
        <only_if>
           <same_as var1="supportDBCreation" value2="true"/>
        </only_if>
        <input name="ModuleHandle" var="_modulehandle"/>
        <input name="dbName" var="dbName"/>
        <input name="nosourcepresentsupported" var="nosourcepresentsupported"/>
     </invoke>
     <!-- Invoke the function BioSPI_DbOpen to open the specified database. -->
     <invoke function="BioSPI_DbOpen">
        <input name="ModuleHandle" var="_modulehandle"/>
        <input name="DbName" var="dbName"/>
        <input name="ReadAccessRequest" value="true"/>
        <input name="WriteAccessRequest" value="true"/>
        <output name="DbHandle" setvar="dbHandle"/>
        <output name="Cursor" setvar="cursor"/>
        <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="ModuleHandle" 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_MODULE_HANDLE.
        </description>
        <equal_to var1="return"
var2="__BioAPIERR_H_FRAMEWORK_INVALID_MODULE_HANDLE"/>
     </assert_condition>
     <invoke activity="CleanUpDBTesting" package="be0a1330-d59e-11d8-9669-
0800200c9a66" break_on_break="true">
        <only_if>
           <same_as var1="supportDBCreation" value2="true"/>
        </only_if>
```

```
        <input name="ModuleHandle" var="_modulehandle"/>
        <input name="dbName" var="dbName"/>
      </invoke>
      <!-- Invoke the functions BioSPI_ModuleDetach and BioSPI_ModuleUnload -->
      <invoke activity="DetachAndUnload" package="be0a1330-d59e-11d8-9669-
0800200c9a66">
        <input name="moduleUuid" var="moduleUuid"/>
        <input name="module" var="_modulehandle"/>
      </invoke>
   </activity>
</package>
```

16.45.4 Assertion 24.3 BioSPI_DbClose_InvalidDBHandle

16.45.4.1 Test Purpose:
To test BioSPI_DbClose with an invalid DB handle if supported by the BSP.

16.45.4.2 Test Scenario:
If supported by the BSP, call BioSPI_DbClose with an invalid DB handle.

16.45.4.3 Expected Results:
Return code other than BioAPI_OK.

16.45.4.4 XML:

16.46 Feature 25. BioSPI Db Create

16.46.1 BioAPI Specification References:
3.3.5.3

16.46.2 Assertion 25.1 BioSPI_DbCreate_ValidParam

16.46.2.1 Test Purpose:
To test BioSPI_DbCreate with valid parameters if suported by the BSP.

16.46.2.2 Test Scenario:
If supported by the BSP, call BioSPI_DbCreate with valid parameters.

16.46.2.3 Expected Results:
Return code BioAPI_OK and a valid DbHandle.

16.46.2.4 XML:

```
<package name="055b3358-0908-1085-9db7-0002a5d5fd2e">
   <author>Author</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
verify if the return code is BioAPI_OK.
         The relevant text in BioAPI 1.1 is quoted below from subclauses 2.5.4.1.
         ____________________________________________________________
         BioAPI_RETURN BioAPI BioAPI_DbCreate
            (BioAPI_HANDLE ModuleHandle,
```

```
          const uint8 *DbName,
          BioAPI_DB_ACCESS_TYPE AccessRequest,
          BioAPI_DB_HANDLE_PTR DbHandle);

          This function creates and opens a new database. The name of the new
database is specified by the input parameter DbName.
        ____________________________________________________________
        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 fail, an UNDECIDED conformity
response is issued.
     </description>
     <!-- UUID of the BSP under test -->
     <input name="_moduleUuid"/>
     <!-- Major version number of the BSP under test -->
     <input name="_moduleVersionMajor"/>
     <!-- Minor version number of the BSP under test -->
     <input name="_moduleVersionMinor"/>
     <!-- Timeout for the BioAPI_NOTIFY_INSERT event -->
     <input name="_inserttimeout"/>
     <!-- Database Name to be opened -->
     <input name="_dbName"/>
     <!-- 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="moduleUuid" var="_moduleUuid"/>
        <input name="moduleVersionMajor" var="_moduleVersionMajor"/>
        <input name="moduleVersionMinor" var="_moduleVersionMinor"/>
        <input name="inserttimeouttime" var="_inserttimeout"/>
        <input name="dbName" var="_dbName"/>
        <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="be0a1330-d59e-11d8-9669-
0800200c9a66" function="BioSPI_ModuleEventHandler"/>
   </assertion>
   <activity name="BioSPI_DbCreate">
     <input name="moduleUuid"/>
     <input name="moduleVersionMajor"/>
     <input name="moduleVersionMinor"/>
     <input name="inserttimeouttime"/>
     <input name="dbSupported"/>
     <input name="dbName"/>
     <input name="readAccessRequest"/>
     <input name="writeAccessRequest"/>
     <!-- This assertion will use ModuleHandle "1" for all BioSPI calls that
require it -->
     <set name="_modulehandle" value="1"/>
     <!-- Invoke the functions BioSPI_ModuleLoad and BioSPI_ModuleAttach
```

```
exposed by the BSP under test.
          The input value for the parameter "deviceIDOrNull" is "0", therefore
the assertion will test the BSP's default device. -->
     <invoke activity="LoadAndAttach" package="be0a1330-d59e-11d8-9669-
0800200c9a66" break_on_break="true">
        <input name="moduleUuid" var="moduleUuid"/>
        <input name="moduleVersionMajor" var="moduleVersionMajor"/>
        <input name="moduleVersionMinor" var="moduleVersionMinor"/>
        <input name="deviceIDOrNull" value="0"/>
        <input name="module" var="_modulehandle"/>
        <input name="eventtimeouttime" var="inserttimeouttime"/>
     </invoke>
     <!-- Remove the database if it already exists -->
     <invoke function="BioSPI_DbDelete">
        <input name="ModuleHandle" var="_modulehandle"/>
        <input name="DbName" var="dbName"/>
        <return setvar="return"/>
     </invoke>
     <!-- Invoke the function BioSPI_DbCreate to create the specified database.
-->
     <invoke function="BioSPI_DbCreate">
        <input name="ModuleHandle" var="_modulehandle"/>
        <input name="DbName" var="dbName"/>
        <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="ModuleHandle" var="_modulehandle"/>
        <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="ModuleHandle" var="_modulehandle"/>
        <input name="DbName" var="dbName"/>
        <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="be0a1330-d59e-11d8-9669-
0800200c9a66">
        <input name="moduleUuid" var="moduleUuid"/>
        <input name="module" var="_modulehandle"/>
     </invoke>
   </activity>
</package>
```

16.46.3 Assertion 25.2 BioSPI_DbCreate_InvalidModuleHandle

16.46.3.1 Test Purpose:
To test BioSPI_DbCreate with an invalid module handle if suported by the BSP.

16.46.3.2 Test Scenario:
If supported by the BSP, call BioSPI_DbCreate with an invalid module handle.

16.46.3.3 Expected Results:
Return code other than BioAPI_OK and a DbHandle of BioAPI_DB_INVALID_HANDLE.

16.46.3.4 XML:

```
<package name="019ecb80-093c-1085-9f29-0002a5d5fd2e">
   <author>Author</author>
   <description>
      This package contains the assertion "BioSPI_DbCreate_InvalidModuleHandle"
   </description>
   <assertion name="BioSPI_DbCreate_InvalidModuleHandle" model="BSPTesting">
      <description>
         This assertion invokes BioSPI_DbCreate with invalid module handle and
verify if the return code is BioAPIERR_H_FRAMEWORK_INVALID_MODULE_HANDLE.
         The relevant text in BioAPI 1.1 is quoted below from subclauses 2.5.4.1.
         ____________________________________________________________
         BioAPI_RETURN BioAPI BioAPI_DbCreate
            (BioAPI_HANDLE ModuleHandle,
            const uint8 *DbName,
            BioAPI_DB_ACCESS_TYPE AccessRequest,
            BioAPI_DB_HANDLE_PTR DbHandle);

            This function creates and opens a new database. The name of the new
database is specified by the input parameter DbName.
         ____________________________________________________________
         and text from subclause 2.3.2.3
         ____________________________________________________________
         #define BioAPIERR_H_FRAMEWORK_INVALID_MODULE_HANDLE \
                       (BioAPI_H_FRAMEWORK_BASE_ERROR +
BioAPI_ERRORCODE_COMMON_EXTENT + 1)
         The given service provider handle is not valid
         ____________________________________________________________

         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_H_FRAMEWORK_INVALID_MODULE_HANDLE.
          5) Detach and unload the BSP.

        If any of the intermediate operations fail, an UNDECIDED conformity
response is issued.
     </description>
     <!-- UUID of the BSP under test -->
     <input name="_moduleUuid"/>
     <!-- Major version number of the BSP under test -->
     <input name="_moduleVersionMajor"/>
     <!-- Minor version number of the BSP under test -->
     <input name="_moduleVersionMinor"/>
     <!-- Timeout for the BioAPI_NOTIFY_INSERT event -->
     <input name="_inserttimeout"/>
     <!-- Database Name to be opened -->
     <input name="_dbName"/>
     <!-- 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="moduleUuid" var="_moduleUuid"/>
        <input name="moduleVersionMajor" var="_moduleVersionMajor"/>
        <input name="moduleVersionMinor" var="_moduleVersionMinor"/>
        <input name="inserttimeouttime" var="_inserttimeout"/>
        <input name="dbName" var="_dbName"/>
        <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="be0a1330-d59e-11d8-9669-
0800200c9a66" function="BioSPI_ModuleEventHandler"/>
   </assertion>
   <activity name="BioSPI_DbCreate">
     <input name="moduleUuid"/>
     <input name="moduleVersionMajor"/>
     <input name="moduleVersionMinor"/>
     <input name="inserttimeouttime"/>
     <input name="nosourcepresentsupported"/>
     <input name="sourcepresenttimeouttime"/>
     <input name="dbName"/>
     <input name="readAccessRequest"/>
     <input name="writeAccessRequest"/>
     <!-- This assertion will use ModuleHandle "1" for all BioSPI calls that
require it -->
     <set name="_modulehandle" value="1"/>
     <!-- Invoke the functions BioSPI_ModuleLoad and BioSPI_ModuleAttach
exposed by the BSP under test.
          The input value for the parameter "deviceIDOrNull" is "0", therefore
the assertion will test the BSP's default device. -->
     <invoke activity="LoadAndAttach" package="be0a1330-d59e-11d8-9669-
0800200c9a66" break_on_break="true">
        <input name="moduleUuid" var="moduleUuid"/>
        <input name="moduleVersionMajor" var="moduleVersionMajor"/>
        <input name="moduleVersionMinor" var="moduleVersionMinor"/>
        <input name="deviceIDOrNull" value="0"/>
```

```
        <input name="module" var="_modulehandle"/>
        <input name="eventtimeouttime" var="inserttimeouttime"/>
     </invoke>
     <!-- Remove the database if it already exists -->
     <invoke function="BioSPI_DbDelete">
        <input name="ModuleHandle" var="_modulehandle"/>
        <input name="DbName" var="dbName"/>
        <return setvar="return"/>
     </invoke>
     <!-- Invoke the function BioSPI_DbCreate to create the specified database.
-->
     <invoke function="BioSPI_DbCreate">
        <input name="ModuleHandle" value="0"/>
        <input name="DbName" var="dbName"/>
        <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_H_FRAMEWORK_INVALID_MODULE_HANDLE and the output DbHandle is set to
BioAPI_DB_INVALID_HANDLE.
        </description>
        <equal_to var1="return"
var2="__BioAPIERR_H_FRAMEWORK_INVALID_MODULE_HANDLE "/>
        <equal_to var1="dbHandle" var2="__BioAPI_DB_INVALID_HANDLE"/>
     </assert_condition>
     <!-- Invoke the functions BioSPI_ModuleDetach and BioSPI_ModuleUnload -->
     <invoke activity="DetachAndUnload" package="be0a1330-d59e-11d8-9669-
0800200c9a66">
        <input name="moduleUuid" var="moduleUuid"/>
        <input name="module" var="_modulehandle"/>
     </invoke>
   </activity>
</package>
```

16.46.4 Assertion 25.3 BioSPI_DbCreate_DbProtected

16.46.4.1 Test Purpose:
To test BioSPI_DbCreate existing database protection if suported by the BSP.

16.46.4.2 Test Scenario:
If supported by the BSP, call BioSPI_DbCreate with valid parameters, with DbName the name of an existing database.

16.46.4.3 Expected Results:
Return code of BioAPIERR_BSP_DATABASE_ALREADY_EXISTS and an invalid DbHandle.

16.46.4.4 XML:

```
<package name="00b93610-093b-1085-b648-0002a5d5fd2e">
   <author>Author</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 verify if the second invocation results returning error code
BioAPIERR_BSP_DATABASE_ALREADY_EXISTS.
         The relevant text in BioAPI 1.1 is quoted below from subclauses 2.5.4.1.
         ____________________________________________________________
         BioAPI_RETURN BioAPI BioAPI_DbCreate
            (BioAPI_HANDLE ModuleHandle,
            const uint8 *DbName,
            BioAPI_DB_ACCESS_TYPE AccessRequest,
            BioAPI_DB_HANDLE_PTR DbHandle);

            This function creates and opens a new database. The name of the new
database is specified by the input parameter DbName.

         Errors
            BioAPIERR_BSP_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_BSP_DATABASE_ALREADY_EXISTS.
            5) Detach and unload the BSP.

         If any of the intermediate operations fail, an UNDECIDED conformity
response is issued.
      </description>
      <!-- UUID of the BSP under test -->
      <input name="_moduleUuid"/>
      <!-- Major version number of the BSP under test -->
      <input name="_moduleVersionMajor"/>
      <!-- Minor version number of the BSP under test -->
      <input name="_moduleVersionMinor"/>
      <!-- Timeout for the BioAPI_NOTIFY_INSERT event -->
      <input name="_inserttimeout"/>
      <!-- Database Name to be opened -->
      <input name="_dbName"/>
      <!-- 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="moduleUuid" var="_moduleUuid"/>
         <input name="moduleVersionMajor" var="_moduleVersionMajor"/>
         <input name="moduleVersionMinor" var="_moduleVersionMinor"/>
         <input name="inserttimeouttime" var="_inserttimeout"/>
         <input name="dbName" var="_dbName"/>
         <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="be0a1330-d59e-11d8-9669-
0800200c9a66" function="BioSPI_ModuleEventHandler"/>
   </assertion>
   <activity name="BioSPI_DbCreate">
      <input name="moduleUuid"/>
      <input name="moduleVersionMajor"/>
      <input name="moduleVersionMinor"/>
      <input name="inserttimeouttime"/>
      <input name="dbName"/>
      <input name="readAccessRequest"/>
      <input name="writeAccessRequest"/>
      <!-- This assertion will use ModuleHandle "1" for all BioSPI calls that
require it -->
      <set name="_modulehandle" value="1"/>
      <!-- Invoke the functions BioSPI_ModuleLoad and BioSPI_ModuleAttach
exposed by the BSP under test.
           The input value for the parameter "deviceIDOrNull" is "0", therefore
the assertion will test the BSP's default device. -->
      <invoke activity="LoadAndAttach" package="be0a1330-d59e-11d8-9669-
0800200c9a66" break_on_break="true">
         <input name="moduleUuid" var="moduleUuid"/>
         <input name="moduleVersionMajor" var="moduleVersionMajor"/>
         <input name="moduleVersionMinor" var="moduleVersionMinor"/>
         <input name="deviceIDOrNull" value="0"/>
         <input name="module" var="_modulehandle"/>
         <input name="eventtimeouttime" var="inserttimeouttime"/>
      </invoke>
      <!-- Remove the database if it already exists -->
      <invoke function="BioSPI_DbDelete">
         <input name="ModuleHandle" var="_modulehandle"/>
         <input name="DbName" var="dbName"/>
         <return setvar="return"/>
      </invoke>
      <!-- Invoke the function BioSPI_DbCreate to create the specified database.
-->
      <invoke function="BioSPI_DbCreate">
         <input name="ModuleHandle" var="_modulehandle"/>
         <input name="DbName" var="dbName"/>
         <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="ModuleHandle" var="_modulehandle"/>
         <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="ModuleHandle" var="_modulehandle"/>
        <input name="DbName" var="dbName"/>
        <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_BSP_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="ModuleHandle" var="_modulehandle"/>
        <input name="DbName" var="dbName"/>
        <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="be0a1330-d59e-11d8-9669-
0800200c9a66">
        <input name="moduleUuid" var="moduleUuid"/>
        <input name="module" var="_modulehandle"/>
     </invoke>
   </activity>
</package>
```

16.46.5 Assertion 25.4 BioSPI_DbCreate_InvalidDBName

16.46.5.1 Test Purpose:

To test BioSPI_DbCreate with an invalid DB name if suported by the BSP.

16.46.5.2 Test Scenario:
If supported by the BSP, call BioSPI_DbCreate with an invalid DB name.

16.46.5.3 Expected Results:
Return code other than BioAPI_OK and a DbHandle of BioAPI_DB_INVALID_HANDLE.

16.46.5.4 XML:

16.46.6 Assertion 25.5 BioSPI_DbCreate_InvalidRequest

16.46.6.1 Test Purpose:
To test BioSPI_DbCreate with an invalid request if suported by the BSP.

16.46.6.2 Test Scenario:
If supported by the BSP, call BioSPI_DbCreate with an invalid request.

16.46.6.3 Expected Results:
Return code other than BioAPI_OK and a DbHandle of BioAPI_DB_INVALID_HANDLE.

16.46.6.4 XML:

16.47 Feature 26. BioSPI Db Delete

16.47.1 BioAPI Specification References:
3.3.5.4

16.47.2 Assertion 26.1 BioSPI_DbDelete_ValidParam

16.47.2.1 Test Purpose:
To test BioSPI_DbDelete with valid parameters if supported by the BSP.

16.47.2.2 Test Scenario:
If supported by the BSP, call BioSPI_DbDelete with valid parameters on an existing, closed database.

16.47.2.3 Expected Results:
Return code of BioAPI_OK and the database deleted.

16.47.2.4 XML:

```
<package name="054198a8-093c-1085-8078-0002a5d5fd2e">
   <author>Author</author>
   <description>
      This package contains the assertion "BioSPI_DbDelete_ValidParam"
   </description>
   <assertion name="BioSPI_DbDelete_ValidParam" model="BSPTesting">
      <description>
         This assertion invokes BioSPI_DbDelete with valid input parameters and
verify if the return code is BioAPI_OK.
         The relevant text in BioAPI 1.1 is quoted below from subclauses 2.5.4.4.
         ____________________________________________________________
         BioAPI_RETURN BioAPI BioAPI_DbDelete
         (BioAPI_HANDLE ModuleHandle,
         const uint8 *DbName);
```

```
        This function deletes all records from the specified database and
removes all state information associated with that 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) Invoke function BioSPI_DbCreate to create the specified database.
           4) Invoke function BioSPI_DbClose to close the created database.
           5) Invoke function BioSPI_DbDelete to delete the created database.
           6) Verify the return code, it is expected to be BioAPI_OK.
           7) Detach and unload the BSP.

        If any of the intermediate operations fail, an UNDECIDED conformity
response is issued.
     </description>
     <!-- UUID of the BSP under test -->
     <input name="_moduleUuid"/>
     <!-- Major version number of the BSP under test -->
     <input name="_moduleVersionMajor"/>
     <!-- Minor version number of the BSP under test -->
     <input name="_moduleVersionMinor"/>
     <!-- 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="_dbName"/>
     <!-- Indicates if the BSP support DB creation-->
     <input name="_supportDBCreation"/>
     <!-- Invocation of the primary activity of this assertion with input
parameter values assigned from the assertion's parameters. -->
     <invoke activity="BioSPI_DbDelete">
        <input name="moduleUuid" var="_moduleUuid"/>
        <input name="moduleVersionMajor" var="_moduleVersionMajor"/>
        <input name="moduleVersionMinor" var="_moduleVersionMinor"/>
        <input name="inserttimeouttime" var="_inserttimeout"/>
        <input name="nosourcepresentsupported" var="_noSourcePresentSupported"/>
        <input name="sourcepresenttimeouttime" var="_sourcepresenttimeout"/>
        <input name="dbName" var="_dbName"/>
        <input name="supportDBCreation" var="_supportDBCreation"/>
     </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="be0a1330-d59e-11d8-9669-
0800200c9a66" function="BioSPI_ModuleEventHandler"/>
   </assertion>
   <activity name="BioSPI_DbDelete">
     <input name="moduleUuid"/>
     <input name="moduleVersionMajor"/>
     <input name="moduleVersionMinor"/>
     <input name="inserttimeouttime"/>
     <input name="nosourcepresentsupported"/>
     <input name="sourcepresenttimeouttime"/>
     <input name="dbName"/>
     <input name="supportDBCreation"/>
     <!-- This assertion will use ModuleHandle "1" for all BioSPI calls that
require it -->
```

```
     <set name="_modulehandle" value="1"/>
     <!-- Invoke the functions BioSPI_ModuleLoad and BioSPI_ModuleAttach
exposed by the BSP under test.
          The input value for the parameter "deviceIDOrNull" is "0", therefore
the assertion will test the BSP's default device. -->
     <invoke activity="LoadAndAttach" package="be0a1330-d59e-11d8-9669-
0800200c9a66" break_on_break="true">
        <input name="moduleUuid" var="moduleUuid"/>
        <input name="moduleVersionMajor" var="moduleVersionMajor"/>
        <input name="moduleVersionMinor" var="moduleVersionMinor"/>
        <input name="deviceIDOrNull" value="0"/>
        <input name="module" var="_modulehandle"/>
        <input name="eventtimeouttime" var="inserttimeouttime"/>
     </invoke>
     <invoke activity="PrepareDBTesting" package="be0a1330-d59e-11d8-9669-
0800200c9a66" break_on_break="true">
        <only_if>
          <same_as var1="supportDBCreation" value2="true"/>
        </only_if>
        <input name="ModuleHandle" var="_modulehandle"/>
        <input name="dbName" var="dbName"/>
        <input name="nosourcepresentsupported" var="nosourcepresentsupported"/>
     </invoke>
     <!-- Invoke the function BioSPI_DbDelete with valid input parameters -->
     <invoke function="BioSPI_DbDelete">
        <input name="ModuleHandle" var="_modulehandle"/>
        <input name="DbName" var="dbName"/>
        <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 BioAPI_OK
        </description>
        <equal_to var1="return" var2="__BioAPI_OK"/>
     </assert_condition>
     <!-- Invoke the functions BioSPI_ModuleDetach and BioSPI_ModuleUnload -->
     <invoke activity="DetachAndUnload" package="be0a1330-d59e-11d8-9669-
0800200c9a66">
        <input name="moduleUuid" var="moduleUuid"/>
        <input name="module" var="_modulehandle"/>
     </invoke>
   </activity>
</package>
```

16.47.3 Assertion 26.2 BioSPI_DbDelete_OpenDbProtected

16.47.3.1 Test Purpose:
To test that BioSPI_DbDelete does not delete an open database if supported by the BSP.

16.47.3.2 Test Scenario:
If supported by the BSP, open a database, then call BioSPI_DbDelete with a valid ModuleHandle and DbName the name of the open database.

16.47.3.3 Expected Results:
Return code BioAPIERR_BSP_DATABASE_IS_OPEN and the database not deleted.

16.47.3.4 XML:

```
<package name="00581ab0-0970-1085-9505-0002a5d5fd2e">
   <author>Author</author>
   <description>
      This package contains the assertion "BioSPI_DbDelete_OpenDbProtected"
   </description>
   <assertion name="BioSPI_DbDelete_OpenDbProtected" model="BSPTesting">
      <description>
         This assertion invokes BioSPI_DbDelete when the database is opened and
verify if the return code is BioAPIERR_BSP_DATABASE_IS_OPEN.
         The relevant text in BioAPI 1.1 is quoted below from subclauses 2.5.4.4.
         ____________________________________________________________
         BioAPI_RETURN BioAPI BioAPI_DbDelete
         (BioAPI_HANDLE ModuleHandle,
         const uint8 *DbName);
         This function deletes all records from the specified database and
removes all state information associated with that database.

         Errors
            BioAPIERR_BSP_DATABASE_IS_OPEN
         ____________________________________________________________
         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.
            5) Invoke function BioSPI_DbDelete with valid parameters.
            6) Verify the return code, it is expected to be
BioAPIERR_BSP_DATABASE_IS_OPEN.
            7) Invoke function BioSPI_DbClose to close the database.
            8) Invoke function BioSPI_DbDelete to delete the database.
            9) Detach and unload the BSP.

         If any of the intermediate operations fail, an UNDECIDED conformity
response is issued.
      </description>
      <!-- UUID of the BSP under test -->
      <input name="_moduleUuid"/>
      <!-- Major version number of the BSP under test -->
      <input name="_moduleVersionMajor"/>
      <!-- Minor version number of the BSP under test -->
      <input name="_moduleVersionMinor"/>
      <!-- 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="_dbName"/>
      <!-- Invocation of the primary activity of this assertion with input
parameter values assigned from the assertion's parameters. -->
      <invoke activity="BioSPI_DbDelete">
         <input name="moduleUuid" var="_moduleUuid"/>
         <input name="moduleVersionMajor" var="_moduleVersionMajor"/>
         <input name="moduleVersionMinor" var="_moduleVersionMinor"/>
         <input name="inserttimeouttime" var="_inserttimeout"/>
         <input name="nosourcepresentsupported" var="_noSourcePresentSupported"/>
         <input name="sourcepresenttimeouttime" var="_sourcepresenttimeout"/>
         <input name="dbName" var="_dbName"/>
      </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="be0a1330-d59e-11d8-9669-
0800200c9a66" function="BioSPI_ModuleEventHandler"/>
   </assertion>
   <activity name="BioSPI_DbDelete">
      <input name="moduleUuid"/>
      <input name="moduleVersionMajor"/>
      <input name="moduleVersionMinor"/>
      <input name="inserttimeouttime"/>
      <input name="nosourcepresentsupported"/>
      <input name="sourcepresenttimeouttime"/>
      <input name="dbName"/>
      <input name="readAccessRequest"/>
      <input name="writeAccessRequest"/>
      <!-- This assertion will use ModuleHandle "1" for all BioSPI calls that
require it -->
      <set name="_modulehandle" value="1"/>
      <!-- Invoke the functions BioSPI_ModuleLoad and BioSPI_ModuleAttach
exposed by the BSP under test.
           The input value for the parameter "deviceIDOrNull" is "0", therefore
the assertion will test the BSP's default device. -->
      <invoke activity="LoadAndAttach" package="be0a1330-d59e-11d8-9669-
0800200c9a66" break_on_break="true">
         <input name="moduleUuid" var="moduleUuid"/>
         <input name="moduleVersionMajor" var="moduleVersionMajor"/>
         <input name="moduleVersionMinor" var="moduleVersionMinor"/>
         <input name="deviceIDOrNull" value="0"/>
         <input name="module" var="_modulehandle"/>
         <input name="eventtimeouttime" var="inserttimeouttime"/>
      </invoke>
      <!-- Remove the database if it already exists -->
      <invoke function="BioSPI_DbDelete">
         <input name="ModuleHandle" var="_modulehandle"/>
         <input name="DbName" var="dbName"/>
         <return setvar="return"/>
      </invoke>
      <!-- Invoke the function BioSPI_DbCreate to create the specified database.
-->
      <invoke function="BioSPI_DbCreate">
         <input name="ModuleHandle" var="_modulehandle"/>
         <input name="DbName" var="dbName"/>
         <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>
      <!-- Invoke the function BioSPI_DbDelete valid parameters -->
      <invoke function="BioSPI_DbDelete">
         <input name="ModuleHandle" var="_modulehandle"/>
         <input name="DbName" var="dbName"/>
         <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_BSP_DATABASE_IS_OPEN
         </description>
         <equal_to var1="return" var2="__BioAPIERR_BSP_DATABASE_IS_OPEN"/>
      </assert_condition>
      <!-- Invoke the function BioSPI_DbClose with valid input parameters -->
      <invoke function="BioSPI_DbClose">
         <input name="ModuleHandle" var="_modulehandle"/>
         <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 parameters to delete
the created database -->
      <invoke function="BioSPI_DbDelete">
         <input name="ModuleHandle" var="_modulehandle"/>
         <input name="DbName" var="dbName"/>
         <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="be0a1330-d59e-11d8-9669-
0800200c9a66">
         <input name="moduleUuid" var="moduleUuid"/>
         <input name="module" var="_modulehandle"/>
      </invoke>
   </activity>
</package>
```

16.47.4 Assertion 26.3 BioSPI_DbDelete_InvalidModuleHandle

16.47.4.1 Test Purpose:

To test BioSPI_DbDelete with invalid module handle if supported by the BSP.

16.47.4.2 Test Scenario:

If supported by the BSP, call BioSPI_DbDelete with an invalid module handle.

16.47.4.3 Expected Results:
Return code other than BioAPI_OK and the database not deleted if a valid database name was passed into BioSPi_DbDelete.

16.47.4.4 XML:

```
<package name="035a1fb0-096e-1085-ba36-0002a5d5fd2e">
   <author>Author</author>
   <description>
      This package contains the assertion "BioSPI_DbDelete_InvalidModuleHandle"
   </description>
   <assertion name="BioSPI_DbDelete_InvalidModuleHandle" model="BSPTesting">
      <description>
         This assertion invokes BioSPI_DbDelete with an invalid module handle and
verify if the return code is BioAPIERR_H_FRAMEWORK_INVALID_MODULE_HANDLE.
         The relevant text in BioAPI 1.1 is quoted below from subclauses 2.5.4.4.
         ____________________________________________________________
         BioAPI_RETURN BioAPI BioAPI_DbDelete
         (BioAPI_HANDLE ModuleHandle,
         const uint8 *DbName);
         This function deletes all records from the specified database and
removes all state information associated with that database.
         ____________________________________________________________
         and text from subclause 2.3.2.3
         ____________________________________________________________
         #define BioAPIERR_H_FRAMEWORK_INVALID_MODULE_HANDLE \
                      (BioAPI_H_FRAMEWORK_BASE_ERROR +
BioAPI_ERRORCODE_COMMON_EXTENT + 1)
         The given service provider handle is not valid
         ____________________________________________________________

         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_H_FRAMEWORK_INVALID_MODULE_HANDLE.
            7) Detach and unload the BSP.

         If any of the intermediate operations fail, an UNDECIDED conformity
response is issued.
      </description>
      <!-- UUID of the BSP under test -->
      <input name="_moduleUuid"/>
      <!-- Major version number of the BSP under test -->
      <input name="_moduleVersionMajor"/>
      <!-- Minor version number of the BSP under test -->
      <input name="_moduleVersionMinor"/>
      <!-- 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="_dbName"/>
      <!-- Indicates if the BSP support DB creation-->
      <input name="_supportDBCreation"/>
```

```
     <!-- Invocation of the primary activity of this assertion with input
parameter values assigned from the assertion's parameters. -->
     <invoke activity="BioSPI_DbDelete">
        <input name="moduleUuid" var="_moduleUuid"/>
        <input name="moduleVersionMajor" var="_moduleVersionMajor"/>
        <input name="moduleVersionMinor" var="_moduleVersionMinor"/>
        <input name="inserttimeouttime" var="_inserttimeout"/>
        <input name="nosourcepresentsupported" var="_noSourcePresentSupported"/>
        <input name="sourcepresenttimeouttime" var="_sourcepresenttimeout"/>
        <input name="dbName" var="_dbName"/>
        <input name="supportDBCreation" var="_supportDBCreation"/>
     </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="be0a1330-d59e-11d8-9669-
0800200c9a66" function="BioSPI_ModuleEventHandler"/>
   </assertion>
   <activity name="BioSPI_DbDelete">
     <input name="moduleUuid"/>
     <input name="moduleVersionMajor"/>
     <input name="moduleVersionMinor"/>
     <input name="inserttimeouttime"/>
     <input name="nosourcepresentsupported"/>
     <input name="sourcepresenttimeouttime"/>
     <input name="dbName"/>
     <input name="supportDBCreation"/>
     <!-- This assertion will use ModuleHandle "1" for all BioSPI calls that
require it -->
     <set name="_modulehandle" value="1"/>
     <!-- Invoke the functions BioSPI_ModuleLoad and BioSPI_ModuleAttach
exposed by the BSP under test.
          The input value for the parameter "deviceIDOrNull" is "0", therefore
the assertion will test the BSP's default device. -->
     <invoke activity="LoadAndAttach" package="be0a1330-d59e-11d8-9669-
0800200c9a66" break_on_break="true">
        <input name="moduleUuid" var="moduleUuid"/>
        <input name="moduleVersionMajor" var="moduleVersionMajor"/>
        <input name="moduleVersionMinor" var="moduleVersionMinor"/>
        <input name="deviceIDOrNull" value="0"/>
        <input name="module" var="_modulehandle"/>
        <input name="eventtimeouttime" var="inserttimeouttime"/>
     </invoke>
     <invoke activity="PrepareDBTesting" package="be0a1330-d59e-11d8-9669-
0800200c9a66" break_on_break="true">
        <only_if>
           <same_as var1="supportDBCreation" value2="true"/>
        </only_if>
        <input name="ModuleHandle" var="_modulehandle"/>
        <input name="dbName" var="dbName"/>
        <input name="nosourcepresentsupported" var="nosourcepresentsupported"/>
     </invoke>
     <!-- Invoke the function BioSPI_DbDelete with an invalid module handle -->
     <invoke function="BioSPI_DbDelete">
        <input name="ModuleHandle" value="0"/>
        <input name="DbName" var="dbName"/>
        <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_H_FRAMEWORK_INVALID_MODULE_HANDLE
        </description>
        <equal_to var1="return"
var2="__BioAPIERR_H_FRAMEWORK_INVALID_MODULE_HANDLE"/>
     </assert_condition>
     <!-- Invoke the function BioSPI_DbDelete with  valid parameters to delete
the created database -->
     <invoke function="BioSPI_DbDelete">
        <input name="ModuleHandle" var="_modulehandle"/>
        <input name="DbName" var="dbName"/>
        <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="be0a1330-d59e-11d8-9669-
0800200c9a66">
        <input name="moduleUuid" var="moduleUuid"/>
        <input name="module" var="_modulehandle"/>
     </invoke>
   </activity>
</package>
```

16.48 Feature 27. BioSPI Db Set Cursor

16.48.1 BioAPI Specification References:
3.3.5.5

16.48.2 Assertion 27.1 BioSPI_DbSetCursor_ValidParam

16.48.2.1 Test Purpose:
To test BioSPI_DbSetCursor with valid parameters if supported by the BSP.

16.48.2.2 Test Scenario:
If supported by the BSP, call BioSPI_DbSetCursor with valid parameters.

16.48.2.3 Expected Results:
Return code of BioAPI_OK and a valid Cursor.

16.48.2.4 XML:

```
<package name="0362e5c8-0978-1085-899a-0002a5d5fd2e">
   <author>Author</author>
   <description>
      This package contains the assertion "BioSPI_DbSetCursor_ValidParam"
   </description>
   <assertion name="BioSPI_DbSetCursor_ValidParam" model="BSPTesting">
      <description>
```

```
        This assertion invokes BioSPI_DbSetCursor with valid input parameters
and verify if the return code is BioAPI_OK.
        The relevant text in BioAPI 1.1 is quoted below from subclauses 2.5.4.5.
        ____________________________________________________________
        BioAPI_RETURN BioAPI BioAPI_DbSetCursor
        (BioAPI_HANDLE ModuleHandle,
        BioAPI_DB_HANDLE DbHandle,
        const BioAPI_UUID *KeyValue,
        BioAPI_DB_CURSOR_PTR Cursor);
        The Cursor is set to point to the record indicated by the KeyValue in
the database identified by the
                    DbHandle.
        ____________________________________________________________
        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) Invoke function BioSPI_DbSetCursor with valid input parameters.
           5) Verify the return code, it is expected to be BioAPI_OK.
           6) Detach and unload the BSP.

        If any of the intermediate operations fail, an UNDECIDED conformity
response is issued.
     </description>
     <!-- UUID of the BSP under test -->
     <input name="_moduleUuid"/>
     <!-- Major version number of the BSP under test -->
     <input name="_moduleVersionMajor"/>
     <!-- Minor version number of the BSP under test -->
     <input name="_moduleVersionMinor"/>
     <!-- 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="_dbName"/>
     <!-- Indicates if the BSP support DB creation-->
     <input name="_supportDBCreation"/>
     <!-- Invocation of the primary activity of this assertion with input
parameter values assigned from the assertion's parameters. -->
     <invoke activity="BioSPI_DbSetCursor">
        <input name="moduleUuid" var="_moduleUuid"/>
        <input name="moduleVersionMajor" var="_moduleVersionMajor"/>
        <input name="moduleVersionMinor" var="_moduleVersionMinor"/>
        <input name="inserttimeouttime" var="_inserttimeout"/>
        <input name="nosourcepresentsupported" var="_noSourcePresentSupported"/>
        <input name="sourcepresenttimeouttime" var="_sourcepresenttimeout"/>
        <input name="dbName" var="_dbName"/>
        <input name="supportDBCreation" var="_supportDBCreation"/>
     </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="be0a1330-d59e-11d8-9669-
0800200c9a66" function="BioSPI_ModuleEventHandler"/>
   </assertion>
   <activity name="BioSPI_DbSetCursor">
     <input name="moduleUuid"/>
```

```
     <input name="moduleVersionMajor"/>
     <input name="moduleVersionMinor"/>
     <input name="inserttimeouttime"/>
     <input name="nosourcepresentsupported"/>
     <input name="sourcepresenttimeouttime"/>
     <input name="dbName"/>
     <input name="supportDBCreation"/>
     <!-- This assertion will use ModuleHandle "1" for all BioSPI calls that
require it -->
     <set name="_modulehandle" value="1"/>
     <!-- Invoke the functions BioSPI_ModuleLoad and BioSPI_ModuleAttach
exposed by the BSP under test.
          The input value for the parameter "deviceIDOrNull" is "0", therefore
the assertion will test the BSP's default device. -->
     <invoke activity="LoadAndAttach" package="be0a1330-d59e-11d8-9669-
0800200c9a66" break_on_break="true">
        <input name="moduleUuid" var="moduleUuid"/>
        <input name="moduleVersionMajor" var="moduleVersionMajor"/>
        <input name="moduleVersionMinor" var="moduleVersionMinor"/>
        <input name="deviceIDOrNull" value="0"/>
        <input name="module" var="_modulehandle"/>
        <input name="eventtimeouttime" var="inserttimeouttime"/>
     </invoke>
     <invoke activity="PrepareDBTesting" package="be0a1330-d59e-11d8-9669-
0800200c9a66" break_on_break="true">
        <only_if>
           <same_as var1="supportDBCreation" value2="true"/>
        </only_if>
        <input name="ModuleHandle" var="_modulehandle"/>
        <input name="dbName" var="dbName"/>
        <input name="nosourcepresentsupported" var="nosourcepresentsupported"/>
        <output name="biruuid" setvar="biruuid"/>
     </invoke>
     <!-- Invoke the function BioSPI_DbOpen to open the specified database. -->
     <invoke function="BioSPI_DbOpen">
        <input name="ModuleHandle" var="_modulehandle"/>
        <input name="DbName" var="dbName"/>
        <input name="ReadAccessRequest" value="true"/>
        <input name="WriteAccessRequest" value="true"/>
        <output name="DbHandle" setvar="dbHandle"/>
        <output name="Cursor" setvar="cursor"/>
        <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_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 function BioSPI_DbSetCursor with valid input parameters -->
     <invoke function="BioSPI_DbSetCursor">
        <input name="ModuleHandle" var="_modulehandle"/>
        <input name="DbHandle" var="dbHandle"/>
        <input name="KeyValue" var="biruuid"/>
        <output name="Cursor" setvar="cursor"/>
        <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_DbSetCursor 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="ModuleHandle" var="_modulehandle"/>
        <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 activity="CleanUpDBTesting" package="be0a1330-d59e-11d8-9669-
0800200c9a66" break_on_break="true">
        <only_if>
           <same_as var1="supportDBCreation" value2="true"/>
        </only_if>
        <input name="ModuleHandle" var="_modulehandle"/>
        <input name="dbName" var="dbName"/>
     </invoke>
     <!-- Invoke the functions BioSPI_ModuleDetach and BioSPI_ModuleUnload -->
     <invoke activity="DetachAndUnload" package="be0a1330-d59e-11d8-9669-
0800200c9a66">
        <input name="moduleUuid" var="moduleUuid"/>
        <input name="module" var="_modulehandle"/>
     </invoke>
   </activity>
</package>
```

16.48.3 Assertion 27.2 BioSPI_DbSetCursor_RecordNotFound

16.48.3.1 Test Purpose:
To test BioSPI_DbSetCursor "record not found" if supported by the BSP.

16.48.3.2 Test Scenario:
If supported by the BSP, call BioSPI_DbSetCursor with valid parameters but KeyValue not in the database.

16.48.3.3 Expected Results:
Return code of BioAPIERR_BSP_RECORD_NOT_FOUND.

16.48.3.4 XML:

```
<package name="00ff0cf8-0979-1085-8859-0002a5d5fd2e">
   <author>Author</author>
   <description>
      This package contains the assertion "BioSPI_DbSetCursor_RecordNotFound"
```

```
   </description>
   <assertion name="BioSPI_DbSetCursor_RecordNotFound" model="BSPTesting">
      <description>
         This assertion invokes BioSPI_DbSetCursor with a keyvalue that does not
exist in the database, and verify if the return code is
BioAPIERR_BSP_RECORD_NOT_FOUND.
         The relevant text in BioAPI 1.1 is quoted below from subclauses 2.5.4.5.
         ____________________________________________________________
         BioAPI_RETURN BioAPI BioAPI_DbSetCursor
         (BioAPI_HANDLE ModuleHandle,
         BioAPI_DB_HANDLE DbHandle,
         const BioAPI_UUID *KeyValue,
         BioAPI_DB_CURSOR_PTR Cursor);
         The Cursor is set to point to the record indicated by the KeyValue in
the database identified by the
                     DbHandle.
         Errors
            BioAPIERR_BSP_RECORD_NOT_FOUND
         ____________________________________________________________
         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) Invoke function BioSPI_DbSetCursor with a keyvalue that does not
exist in the database.
            5) Verify the return code, it is expected to be
BioAPIERR_BSP_RECORD_NOT_FOUND.
            6) Detach and unload the BSP.

         If any of the intermediate operations fail, an UNDECIDED conformity
response is issued.
      </description>
      <!-- UUID of the BSP under test -->
      <input name="_moduleUuid"/>
      <!-- Major version number of the BSP under test -->
      <input name="_moduleVersionMajor"/>
      <!-- Minor version number of the BSP under test -->
      <input name="_moduleVersionMinor"/>
      <!-- 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="_dbName"/>
      <!-- Indicates if the BSP support DB creation-->
      <input name="_supportDBCreation"/>
      <!-- BIR Keyvalue -->
      <input name="_keyvalue"/>
      <!-- Invocation of the primary activity of this assertion with input
parameter values assigned from the assertion's parameters. -->
      <invoke activity="BioSPI_DbSetCursor">
         <input name="moduleUuid" var="_moduleUuid"/>
         <input name="moduleVersionMajor" var="_moduleVersionMajor"/>
         <input name="moduleVersionMinor" var="_moduleVersionMinor"/>
         <input name="inserttimeouttime" var="_inserttimeout"/>
         <input name="nosourcepresentsupported" var="_noSourcePresentSupported"/>
         <input name="sourcepresenttimeouttime" var="_sourcepresenttimeout"/>
         <input name="dbName" var="_dbName"/>
```

```
        <input name="supportDBCreation" var="_supportDBCreation"/>
        <input name="keyvalue" var="_keyvalue"/>
     </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="be0a1330-d59e-11d8-9669-
0800200c9a66" function="BioSPI_ModuleEventHandler"/>
   </assertion>
   <activity name="BioSPI_DbSetCursor">
     <input name="moduleUuid"/>
     <input name="moduleVersionMajor"/>
     <input name="moduleVersionMinor"/>
     <input name="inserttimeouttime"/>
     <input name="nosourcepresentsupported"/>
     <input name="sourcepresenttimeouttime"/>
     <input name="dbName"/>
     <input name="supportDBCreation"/>
     <input name="keyvalue"/>
     <!-- This assertion will use ModuleHandle "1" for all BioSPI calls that
require it -->
     <set name="_modulehandle" value="1"/>
     <!-- Invoke the functions BioSPI_ModuleLoad and BioSPI_ModuleAttach
exposed by the BSP under test.
          The input value for the parameter "deviceIDOrNull" is "0", therefore
the assertion will test the BSP's default device. -->
     <invoke activity="LoadAndAttach" package="be0a1330-d59e-11d8-9669-
0800200c9a66" break_on_break="true">
        <input name="moduleUuid" var="moduleUuid"/>
        <input name="moduleVersionMajor" var="moduleVersionMajor"/>
        <input name="moduleVersionMinor" var="moduleVersionMinor"/>
        <input name="deviceIDOrNull" value="0"/>
        <input name="module" var="_modulehandle"/>
        <input name="eventtimeouttime" var="inserttimeouttime"/>
     </invoke>
     <invoke activity="PrepareDBTesting" package="be0a1330-d59e-11d8-9669-
0800200c9a66" break_on_break="true">
        <only_if>
           <same_as var1="supportDBCreation" value2="true"/>
        </only_if>
        <input name="ModuleHandle" var="_modulehandle"/>
        <input name="dbName" var="dbName"/>
        <input name="nosourcepresentsupported" var="nosourcepresentsupported"/>
        <output name="biruuid" setvar="biruuid"/>
     </invoke>
     <!-- Invoke the function BioSPI_DbOpen to open the specified database. -->
     <invoke function="BioSPI_DbOpen">
        <input name="ModuleHandle" var="_modulehandle"/>
        <input name="DbName" var="dbName"/>
        <input name="ReadAccessRequest" value="true"/>
        <input name="WriteAccessRequest" value="true"/>
        <output name="DbHandle" setvar="dbHandle"/>
        <output name="Cursor" setvar="cursor"/>
        <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_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 function BioSPI_DbSetCursor with valid input parameters -->
     <invoke function="BioSPI_DbSetCursor">
        <input name="ModuleHandle" var="_modulehandle"/>
        <input name="DbHandle" var="dbHandle"/>
        <input name="KeyValue" var="keyvalue"/>
        <output name="Cursor" setvar="cursor"/>
        <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_DbSetCursor has returned
BioAPIERR_BSP_RECORD_NOT_FOUND
        </description>
        <equal_to var1="return" var2="__BioAPIERR_BSP_RECORD_NOT_FOUND "/>
     </assert_condition>
     <!-- Invoke the function BioSPI_DbClose with valid input parameters -->
     <invoke function="BioSPI_DbClose">
        <input name="ModuleHandle" var="_modulehandle"/>
        <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 activity="CleanUpDBTesting" package="be0a1330-d59e-11d8-9669-
0800200c9a66" break_on_break="true">
        <only_if>
          <same_as var1="supportDBCreation" value2="true"/>
        </only_if>
        <input name="ModuleHandle" var="_modulehandle"/>
        <input name="dbName" var="dbName"/>
     </invoke>
     <!-- Invoke the functions BioSPI_ModuleDetach and BioSPI_ModuleUnload -->
     <invoke activity="DetachAndUnload" package="be0a1330-d59e-11d8-9669-
0800200c9a66">
        <input name="moduleUuid" var="moduleUuid"/>
        <input name="module" var="_modulehandle"/>
     </invoke>
   </activity>
</package>
```

16.48.4 Assertion 27.3 BioSPI_DbSetCursor_InvalidModuleHandle

16.48.4.1 Test Purpose:

To test BioSPI_DbSetCursor with an invalid module handle if supported by the BSP.

16.48.4.2 Test Scenario:

If supported by the BSP, call BioSPI_DbSetCursor with and invalid module handle.

16.48.4.3 Expected Results:
Return code other than BioAPI_OK and an invalid Cursor.

16.48.4.4 XML:

```
<package name="058dd448-0978-1085-a946-0002a5d5fd2e">
   <author>Author</author>
   <description>
      This package contains the assertion
"BioSPI_DbSetCursor_InvalidModuleHandle"
   </description>
   <assertion name="BioSPI_DbSetCursor_InvalidModuleHandle" model="BSPTesting">
      <description>
         This assertion invokes BioSPI_DbSetCursor with invalid module handle and
verify if the return code is BioAPIERR_H_FRAMEWORK_INVALID_MODULE_HANDLE.
         The relevant text in BioAPI 1.1 is quoted below from subclauses 2.5.4.5.
         ____________________________________________________________
         BioAPI_RETURN BioAPI BioAPI_DbSetCursor
         (BioAPI_HANDLE ModuleHandle,
         BioAPI_DB_HANDLE DbHandle,
         const BioAPI_UUID *KeyValue,
         BioAPI_DB_CURSOR_PTR Cursor);
         The Cursor is set to point to the record indicated by the KeyValue in
the database identified by the
                        DbHandle.
         ____________________________________________________________
         and text from subclause 2.3.2.3
         ____________________________________________________________
         #define BioAPIERR_H_FRAMEWORK_INVALID_MODULE_HANDLE \
                        (BioAPI_H_FRAMEWORK_BASE_ERROR +
BioAPI_ERRORCODE_COMMON_EXTENT + 1)
         The given service provider handle is not valid
         ____________________________________________________________

         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) Invoke function BioSPI_DbSetCursor with invalid module handle.
            5) Verify the return code, it is expected to be
BioAPIERR_H_FRAMEWORK_INVALID_MODULE_HANDLE.
            6) Detach and unload the BSP.

         If any of the intermediate operations fail, an UNDECIDED conformity
response is issued.
      </description>
      <!-- UUID of the BSP under test -->
      <input name="_moduleUuid"/>
      <!-- Major version number of the BSP under test -->
      <input name="_moduleVersionMajor"/>
      <!-- Minor version number of the BSP under test -->
      <input name="_moduleVersionMinor"/>
      <!-- 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="_dbName"/>
      <!-- Indicates if the BSP support DB creation-->
      <input name="_supportDBCreation"/>
      <!-- Invocation of the primary activity of this assertion with input
parameter values assigned from the assertion's parameters. -->
      <invoke activity="BioSPI_DbSetCursor">
         <input name="moduleUuid" var="_moduleUuid"/>
         <input name="moduleVersionMajor" var="_moduleVersionMajor"/>
         <input name="moduleVersionMinor" var="_moduleVersionMinor"/>
         <input name="inserttimeouttime" var="_inserttimeout"/>
         <input name="nosourcepresentsupported" var="_noSourcePresentSupported"/>
         <input name="sourcepresenttimeouttime" var="_sourcepresenttimeout"/>
         <input name="dbName" var="_dbName"/>
         <input name="supportDBCreation" var="_supportDBCreation"/>
      </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="be0a1330-d59e-11d8-9669-
0800200c9a66" function="BioSPI_ModuleEventHandler"/>
   </assertion>
   <activity name="BioSPI_DbSetCursor">
      <input name="moduleUuid"/>
      <input name="moduleVersionMajor"/>
      <input name="moduleVersionMinor"/>
      <input name="inserttimeouttime"/>
      <input name="nosourcepresentsupported"/>
      <input name="sourcepresenttimeouttime"/>
      <input name="dbName"/>
      <input name="supportDBCreation"/>
      <!-- This assertion will use ModuleHandle "1" for all BioSPI calls that
require it -->
      <set name="_modulehandle" value="1"/>
      <!-- Invoke the functions BioSPI_ModuleLoad and BioSPI_ModuleAttach
exposed by the BSP under test.
           The input value for the parameter "deviceIDOrNull" is "0", therefore
the assertion will test the BSP's default device. -->
      <invoke activity="LoadAndAttach" package="be0a1330-d59e-11d8-9669-
0800200c9a66" break_on_break="true">
         <input name="moduleUuid" var="moduleUuid"/>
         <input name="moduleVersionMajor" var="moduleVersionMajor"/>
         <input name="moduleVersionMinor" var="moduleVersionMinor"/>
         <input name="deviceIDOrNull" value="0"/>
         <input name="module" var="_modulehandle"/>
         <input name="eventtimeouttime" var="inserttimeouttime"/>
      </invoke>
      <invoke activity="PrepareDBTesting" package="be0a1330-d59e-11d8-9669-
0800200c9a66" break_on_break="true">
         <only_if>
            <same_as var1="supportDBCreation" value2="true"/>
         </only_if>
         <input name="ModuleHandle" var="_modulehandle"/>
         <input name="dbName" var="dbName"/>
         <input name="nosourcepresentsupported" var="nosourcepresentsupported"/>
         <output name="biruuid" setvar="biruuid"/>
      </invoke>
      <!-- Invoke the function BioSPI_DbOpen to open the specified database. -->
      <invoke function="BioSPI_DbOpen">
         <input name="ModuleHandle" var="_modulehandle"/>
         <input name="DbName" var="dbName"/>
         <input name="ReadAccessRequest" value="true"/>
         <input name="WriteAccessRequest" value="true"/>
```

```
        <output name="DbHandle" setvar="dbHandle"/>
        <output name="Cursor" setvar="cursor"/>
        <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_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 function BioSPI_DbSetCursor with valid input parameters -->
     <invoke function="BioSPI_DbSetCursor">
        <input name="ModuleHandle" value="0"/>
        <input name="DbHandle" var="dbHandle"/>
        <input name="KeyValue" var="biruuid"/>
        <output name="Cursor" setvar="cursor"/>
        <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_DbSetCursor has returned
BioAPIERR_H_FRAMEWORK_INVALID_MODULE_HANDLE.
        </description>
        <equal_to var1="return"
var2="__BioAPIERR_H_FRAMEWORK_INVALID_MODULE_HANDLE "/>
     </assert_condition>
     <!-- Invoke the function BioSPI_DbClose with valid input parameters -->
     <invoke function="BioSPI_DbClose">
        <input name="ModuleHandle" var="_modulehandle"/>
        <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 activity="CleanUpDBTesting" package="be0a1330-d59e-11d8-9669-
0800200c9a66" break_on_break="true">
        <only_if>
          <same_as var1="supportDBCreation" value2="true"/>
        </only_if>
        <input name="ModuleHandle" var="_modulehandle"/>
        <input name="dbName" var="dbName"/>
     </invoke>
     <!-- Invoke the functions BioSPI_ModuleDetach and BioSPI_ModuleUnload -->
     <invoke activity="DetachAndUnload" package="be0a1330-d59e-11d8-9669-
0800200c9a66">
```

```
        <input name="moduleUuid" var="moduleUuid"/>
        <input name="module" var="_modulehandle"/>
      </invoke>
   </activity>
</package>
```

16.48.5 Assertion 27.4 BioSPI_DbSetCursor_InvalidDBHandle

16.48.5.1 Test Purpose:
To test BioSPI_DbSetCursor with an invalid DB handle if supported by the BSP.

16.48.5.2 Test Scenario:
If supported by the BSP, call BioSPI_DbSetCursor with and invalid DB handle.

16.48.5.3 Expected Results:
Return code other than BioAPI_OK and an invalid Cursor.

16.48.5.4 XML:

16.49 Feature 28. BioSPI Db Free Cursor

16.49.1 BioAPI Specification References:
3.3.5.6

16.49.2 Assertion 28.1 BioSPI_DbFreeCursor_ValidParam

16.49.2.1 Test Purpose:
To test BioSPI_DbFreeCursor with valid parameters if supported by the BSP.

16.49.2.2 Test Scenario:
If supported by the BSP, call BioSPI_DbFreeCursor with valid parameters.

16.49.2.3 Expected Results:
Return code BioAPI_OK.

16.49.2.4 XML:

```
<package name="05109d98-097f-1085-a9f1-0002a5d5fd2e">
   <author>Author</author>
   <description>
      This package contains the assertion "BioSPI_DbFreeCursor_ValidParam"
   </description>
   <assertion name="BioSPI_DbFreeCursor_ValidParam" model="BSPTesting">
      <description>
         This assertion invokes BioSPI_DbFreeCursor with valid input parameters
and verify if the return code is BioAPI_OK.
         The relevant text in BioAPI 1.1 is quoted below from subclauses 2.5.4.6.
         ____________________________________________________________
         BioAPI_RETURN BioAPI BioAPI_DbFreeCursor
         (BioAPI_HANDLE ModuleHandle,
         BioAPI_DB_CURSOR_PTR Cursor);
         Frees memory and resources associated with the specified Cursor.
         ____________________________________________________________
         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) Invoke function BioSPI_DbSetCursor with valid input parameters.
          5) Invoke function BioSPI_DbFreeCursor with valid input parameters.
          6) Verify the return code, it is expected to be BioAPI_OK.
          7) Detach and unload the BSP.

        If any of the intermediate operations fail, an UNDECIDED conformity
response is issued.
     </description>
     <!-- UUID of the BSP under test -->
     <input name="_moduleUuid"/>
     <!-- Major version number of the BSP under test -->
     <input name="_moduleVersionMajor"/>
     <!-- Minor version number of the BSP under test -->
     <input name="_moduleVersionMinor"/>
     <!-- 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="_dbName"/>
     <!-- Indicates if the BSP support DB creation-->
     <input name="_supportDBCreation"/>
     <!-- Invocation of the primary activity of this assertion with input
parameter values assigned from the assertion's parameters. -->
     <invoke activity="BioSPI_DbFreeCursor">
        <input name="moduleUuid" var="_moduleUuid"/>
        <input name="moduleVersionMajor" var="_moduleVersionMajor"/>
        <input name="moduleVersionMinor" var="_moduleVersionMinor"/>
        <input name="inserttimeouttime" var="_inserttimeout"/>
        <input name="nosourcepresentsupported" var="_noSourcePresentSupported"/>
        <input name="sourcepresenttimeouttime" var="_sourcepresenttimeout"/>
        <input name="dbName" var="_dbName"/>
        <input name="supportDBCreation" var="_supportDBCreation"/>
     </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="be0a1330-d59e-11d8-9669-
0800200c9a66" function="BioSPI_ModuleEventHandler"/>
   </assertion>
   <activity name="BioSPI_DbFreeCursor">
     <input name="moduleUuid"/>
     <input name="moduleVersionMajor"/>
     <input name="moduleVersionMinor"/>
     <input name="inserttimeouttime"/>
     <input name="nosourcepresentsupported"/>
     <input name="sourcepresenttimeouttime"/>
     <input name="dbName"/>
     <input name="supportDBCreation"/>
     <!-- This assertion will use ModuleHandle "1" for all BioSPI calls that
require it -->
     <set name="_modulehandle" value="1"/>
     <!-- Invoke the functions BioSPI_ModuleLoad and BioSPI_ModuleAttach
exposed by the BSP under test.
          The input value for the parameter "deviceIDOrNull" is "0", therefore
the assertion will test the BSP's default device. -->
     <invoke activity="LoadAndAttach" package="be0a1330-d59e-11d8-9669-
```

```
0800200c9a66" break_on_break="true">
        <input name="moduleUuid" var="moduleUuid"/>
        <input name="moduleVersionMajor" var="moduleVersionMajor"/>
        <input name="moduleVersionMinor" var="moduleVersionMinor"/>
        <input name="deviceIDOrNull" value="0"/>
        <input name="module" var="_modulehandle"/>
        <input name="eventtimeouttime" var="inserttimeouttime"/>
     </invoke>
     <invoke activity="PrepareDBTesting" package="be0a1330-d59e-11d8-9669-
0800200c9a66" break_on_break="true">
        <only_if>
           <same_as var1="supportDBCreation" value2="true"/>
        </only_if>
        <input name="ModuleHandle" var="_modulehandle"/>
        <input name="dbName" var="dbName"/>
        <input name="nosourcepresentsupported" var="nosourcepresentsupported"/>
        <output name="biruuid" setvar="biruuid"/>
     </invoke>
     <!-- Invoke the function BioSPI_DbOpen to open the specified database. -->
     <invoke function="BioSPI_DbOpen">
        <input name="ModuleHandle" var="_modulehandle"/>
        <input name="DbName" var="dbName"/>
        <input name="ReadAccessRequest" value="true"/>
        <input name="WriteAccessRequest" value="true"/>
        <output name="DbHandle" setvar="dbHandle"/>
        <output name="Cursor" setvar="cursor"/>
        <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_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 function BioSPI_DbSetCursor with valid input parameters -->
     <invoke function="BioSPI_DbSetCursor">
        <input name="ModuleHandle" var="_modulehandle"/>
        <input name="DbHandle" var="dbHandle"/>
        <input name="KeyValue" var="biruuid"/>
        <output name="Cursor" setvar="cursor"/>
        <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_DbSetCursor has returned BioAPI_OK
        </description>
        <equal_to var1="return" var2="__BioAPI_OK"/>
     </assert_condition>
     <!-- Invoke the function BioSPI_DbFreeCursor with valid input parameters -
->
     <invoke function="BioSPI_DbFreeCursor">
        <input name="ModuleHandle" var="_modulehandle"/>
        <input name="Cursor" var="cursor"/>
```

```
        <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_DbFreeCursor 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="ModuleHandle" var="_modulehandle"/>
        <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 activity="CleanUpDBTesting" package="be0a1330-d59e-11d8-9669-
0800200c9a66" break_on_break="true">
        <only_if>
           <same_as var1="supportDBCreation" value2="true"/>
        </only_if>
        <input name="ModuleHandle" var="_modulehandle"/>
        <input name="dbName" var="dbName"/>
      </invoke>
      <!-- Invoke the functions BioSPI_ModuleDetach and BioSPI_ModuleUnload -->
      <invoke activity="DetachAndUnload" package="be0a1330-d59e-11d8-9669-
0800200c9a66">
        <input name="moduleUuid" var="moduleUuid"/>
        <input name="module" var="_modulehandle"/>
      </invoke>
   </activity>
</package>
```

16.49.3 Assertion 28.2 BioSPI_DbFreeCursor_InvalidModuleHandle

16.49.3.1 Test Purpose:
To test BioSPI_DbFreeCursor with an invalid module handle if supported by the BSP.

16.49.3.2 Test Scenario:
If supported by the BSP, call BioSPI_DbFreeCursor with an invalid module handle.

16.49.3.3 Expected Results:
Return code other than BioAPI_OK.

16.49.3.4 XML:

```
<package name="0097bfa8-0980-1085-8279-0002a5d5fd2e">
   <author>Author</author>
   <description>
      This package contains the assertion
```

```
"BioSPI_DbFreeCursor_InvalidModuleHandle"
   </description>
   <assertion name="BioSPI_DbFreeCursor_InvalidModuleHandle" model="BSPTesting">
      <description>
         This assertion invokes BioSPI_DbFreeCursor with invalid module handle
and verify if the return code is BioAPIERR_H_FRAMEWORK_INVALID_MODULE_HANDLE.
         The relevant text in BioAPI 1.1 is quoted below from subclauses 2.5.4.6.
         ____________________________________________________________
         BioAPI_RETURN BioAPI BioAPI_DbFreeCursor
         (BioAPI_HANDLE ModuleHandle,
         BioAPI_DB_CURSOR_PTR Cursor);
         Frees memory and resources associated with the specified Cursor.
         ____________________________________________________________
         and text from subclause 2.3.2.3
         ____________________________________________________________
         #define BioAPIERR_H_FRAMEWORK_INVALID_MODULE_HANDLE \
                      (BioAPI_H_FRAMEWORK_BASE_ERROR +
BioAPI_ERRORCODE_COMMON_EXTENT + 1)
         The given service provider handle is not valid
         ____________________________________________________________

         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) Invoke function BioSPI_DbSetCursor with valid input parameters.
            5) Invoke function BioSPI_DbFreeCursor with invalid moduel handle.
            6) Verify the return code, it is expected to be
BioAPIERR_H_FRAMEWORK_INVALID_MODULE_HANDLE.
            7) Detach and unload the BSP.

         If any of the intermediate operations fail, an UNDECIDED conformity
response is issued.
      </description>
      <!-- UUID of the BSP under test -->
      <input name="_moduleUuid"/>
      <!-- Major version number of the BSP under test -->
      <input name="_moduleVersionMajor"/>
      <!-- Minor version number of the BSP under test -->
      <input name="_moduleVersionMinor"/>
      <!-- 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="_dbName"/>
      <!-- Indicates if the BSP support DB creation-->
      <input name="_supportDBCreation"/>
      <!-- Invocation of the primary activity of this assertion with input
parameter values assigned from the assertion's parameters. -->
      <invoke activity="BioSPI_DbFreeCursor">
         <input name="moduleUuid" var="_moduleUuid"/>
         <input name="moduleVersionMajor" var="_moduleVersionMajor"/>
         <input name="moduleVersionMinor" var="_moduleVersionMinor"/>
         <input name="inserttimeouttime" var="_inserttimeout"/>
         <input name="nosourcepresentsupported" var="_noSourcePresentSupported"/>
         <input name="sourcepresenttimeouttime" var="_sourcepresenttimeout"/>
         <input name="dbName" var="_dbName"/>
```

```
        <input name="supportDBCreation" var="_supportDBCreation"/>
      </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="be0a1330-d59e-11d8-9669-
0800200c9a66" function="BioSPI_ModuleEventHandler"/>
   </assertion>
   <activity name="BioSPI_DbFreeCursor">
      <input name="moduleUuid"/>
      <input name="moduleVersionMajor"/>
      <input name="moduleVersionMinor"/>
      <input name="inserttimeouttime"/>
      <input name="nosourcepresentsupported"/>
      <input name="sourcepresenttimeouttime"/>
      <input name="dbName"/>
      <input name="supportDBCreation"/>
      <!-- This assertion will use ModuleHandle "1" for all BioSPI calls that
require it -->
      <set name="_modulehandle" value="1"/>
      <!-- Invoke the functions BioSPI_ModuleLoad and BioSPI_ModuleAttach
exposed by the BSP under test.
           The input value for the parameter "deviceIDOrNull" is "0", therefore
the assertion will test the BSP's default device. -->
      <invoke activity="LoadAndAttach" package="be0a1330-d59e-11d8-9669-
0800200c9a66" break_on_break="true">
        <input name="moduleUuid" var="moduleUuid"/>
        <input name="moduleVersionMajor" var="moduleVersionMajor"/>
        <input name="moduleVersionMinor" var="moduleVersionMinor"/>
        <input name="deviceIDOrNull" value="0"/>
        <input name="module" var="_modulehandle"/>
        <input name="eventtimeouttime" var="inserttimeouttime"/>
      </invoke>
      <invoke activity="PrepareDBTesting" package="be0a1330-d59e-11d8-9669-
0800200c9a66" break_on_break="true">
        <only_if>
           <same_as var1="supportDBCreation" value2="true"/>
        </only_if>
        <input name="ModuleHandle" var="_modulehandle"/>
        <input name="dbName" var="dbName"/>
        <input name="nosourcepresentsupported" var="nosourcepresentsupported"/>
        <output name="biruuid" setvar="biruuid"/>
      </invoke>
      <!-- Invoke the function BioSPI_DbOpen to open the specified database. -->
      <invoke function="BioSPI_DbOpen">
        <input name="ModuleHandle" var="_modulehandle"/>
        <input name="DbName" var="dbName"/>
        <input name="ReadAccessRequest" value="true"/>
        <input name="WriteAccessRequest" value="true"/>
        <output name="DbHandle" setvar="dbHandle"/>
        <output name="Cursor" setvar="cursor"/>
        <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_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 function BioSPI_DbSetCursor with valid input parameters -->
     <invoke function="BioSPI_DbSetCursor">
        <input name="ModuleHandle" var="_modulehandle"/>
        <input name="DbHandle" var="dbHandle"/>
        <input name="KeyValue" var="biruuid"/>
        <output name="Cursor" setvar="cursor"/>
        <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_DbSetCursor has returned BioAPI_OK
        </description>
        <equal_to var1="return" var2="__BioAPI_OK"/>
     </assert_condition>
     <!-- Invoke the function BioSPI_DbFreeCursor with valid input parameters -
->
     <invoke function="BioSPI_DbFreeCursor">
        <input name="ModuleHandle" value="0"/>
        <input name="Cursor" var="cursor"/>
        <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_DbFreeCursor has returned
BioAPIERR_H_FRAMEWORK_INVALID_MODULE_HANDLE.
        </description>
        <equal_to var1="return"
var2="__BioAPIERR_H_FRAMEWORK_INVALID_MODULE_HANDLE "/>
     </assert_condition>
     <!-- Invoke the function BioSPI_DbClose with valid input parameters -->
     <invoke function="BioSPI_DbClose">
        <input name="ModuleHandle" var="_modulehandle"/>
        <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 activity="CleanUpDBTesting" package="be0a1330-d59e-11d8-9669-
0800200c9a66" break_on_break="true">
        <only_if>
           <same_as var1="supportDBCreation" value2="true"/>
        </only_if>
        <input name="ModuleHandle" var="_modulehandle"/>
        <input name="dbName" var="dbName"/>
     </invoke>
```

```
     <!-- Invoke the functions BioSPI_ModuleDetach and BioSPI_ModuleUnload -->
     <invoke activity="DetachAndUnload" package="be0a1330-d59e-11d8-9669-
0800200c9a66">
        <input name="moduleUuid" var="moduleUuid"/>
        <input name="module" var="_modulehandle"/>
     </invoke>
   </activity>
</package>
```

16.49.4 Assertion 28.3 BioSPI_DbFreeCursor_InvalidCursor

16.49.4.1 Test Purpose:
To test BioSPI_DbFreeCursor "invalid cursor" if supported by the BSP.

16.49.4.2 Test Scenario:
If supported by the BSP, call BioSPI_DbFreeCursor with a valid ModuleHandle and an invalid Cursor.

16.49.4.3 Expected Results:
Return code BioAPIERR_BSP_CURSOR_IS_INVALID.

16.49.4.4 XML:

```
<package name="028a3cf0-0980-1085-88df-0002a5d5fd2e">
   <author>Author</author>
   <description>
      This package contains the assertion "BioSPI_DbFreeCursor_InvalidCursor"
   </description>
   <assertion name="BioSPI_DbFreeCursor_InvalidCursor" model="BSPTesting">
      <description>
         This assertion invokes BioSPI_DbFreeCursor with invalid cursor value and
verify if the return code is BioAPIERR_BSP_CURSOR_IS_INVALID.
         The relevant text in BioAPI 1.1 is quoted below from subclauses 2.5.4.6.
         ____________________________________________________________
         BioAPI_RETURN BioAPI BioAPI_DbFreeCursor
         (BioAPI_HANDLE ModuleHandle,
         BioAPI_DB_CURSOR_PTR Cursor);
         Frees memory and resources associated with the specified Cursor.

         Errors
            BioAPIERR_BSP_CURSOR_IS_INVALID
         ____________________________________________________________
         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) Invoke function BioSPI_DbFreeCursor with invalid cursor value.
            5) Verify the return code, it is expected to be
BioAPIERR_H_FRAMEWORK_INVALID_MODULE_HANDLE.
            6) Detach and unload the BSP.

         If any of the intermediate operations fail, an UNDECIDED conformity
response is issued.
      </description>
      <!-- UUID of the BSP under test -->
      <input name="_moduleUuid"/>
      <!-- Major version number of the BSP under test -->
      <input name="_moduleVersionMajor"/>
      <!-- Minor version number of the BSP under test -->
```

```
     <input name="_moduleVersionMinor"/>
     <!-- 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="_dbName"/>
     <!-- Indicates if the BSP support DB creation-->
     <input name="_supportDBCreation"/>
     <!-- Invocation of the primary activity of this assertion with input
parameter values assigned from the assertion's parameters. -->
     <invoke activity="BioSPI_DbFreeCursor">
        <input name="moduleUuid" var="_moduleUuid"/>
        <input name="moduleVersionMajor" var="_moduleVersionMajor"/>
        <input name="moduleVersionMinor" var="_moduleVersionMinor"/>
        <input name="inserttimeouttime" var="_inserttimeout"/>
        <input name="nosourcepresentsupported" var="_noSourcePresentSupported"/>
        <input name="sourcepresenttimeouttime" var="_sourcepresenttimeout"/>
        <input name="dbName" var="_dbName"/>
        <input name="supportDBCreation" var="_supportDBCreation"/>
     </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="be0a1330-d59e-11d8-9669-
0800200c9a66" function="BioSPI_ModuleEventHandler"/>
   </assertion>
   <activity name="BioSPI_DbFreeCursor">
     <input name="moduleUuid"/>
     <input name="moduleVersionMajor"/>
     <input name="moduleVersionMinor"/>
     <input name="inserttimeouttime"/>
     <input name="nosourcepresentsupported"/>
     <input name="sourcepresenttimeouttime"/>
     <input name="dbName"/>
     <input name="supportDBCreation"/>
     <!-- This assertion will use ModuleHandle "1" for all BioSPI calls that
require it -->
     <set name="_modulehandle" value="1"/>
     <!-- Invoke the functions BioSPI_ModuleLoad and BioSPI_ModuleAttach
exposed by the BSP under test.
          The input value for the parameter "deviceIDOrNull" is "0", therefore
the assertion will test the BSP's default device. -->
     <invoke activity="LoadAndAttach" package="be0a1330-d59e-11d8-9669-
0800200c9a66" break_on_break="true">
        <input name="moduleUuid" var="moduleUuid"/>
        <input name="moduleVersionMajor" var="moduleVersionMajor"/>
        <input name="moduleVersionMinor" var="moduleVersionMinor"/>
        <input name="deviceIDOrNull" value="0"/>
        <input name="module" var="_modulehandle"/>
        <input name="eventtimeouttime" var="inserttimeouttime"/>
     </invoke>
     <invoke activity="PrepareDBTesting" package="be0a1330-d59e-11d8-9669-
0800200c9a66" break_on_break="true">
        <only_if>
          <same_as var1="supportDBCreation" value2="true"/>
        </only_if>
        <input name="ModuleHandle" var="_modulehandle"/>
        <input name="dbName" var="dbName"/>
        <input name="nosourcepresentsupported" var="nosourcepresentsupported"/>
```

```
     </invoke>
     <!-- Invoke the function BioSPI_DbOpen to open the specified database. -->
     <invoke function="BioSPI_DbOpen">
        <input name="ModuleHandle" var="_modulehandle"/>
        <input name="DbName" var="dbName"/>
        <input name="ReadAccessRequest" value="true"/>
        <input name="WriteAccessRequest" value="true"/>
        <output name="DbHandle" setvar="dbHandle"/>
        <output name="Cursor" setvar="cursor"/>
        <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_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_DbFreeCursor with valid input parameters -
->
     <invoke function="BioSPI_DbFreeCursor">
        <input name="ModuleHandle" var="_modulehandle"/>
        <input name="Cursor" 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_DbFreeCursor has returned
BioAPIERR_BSP_CURSOR_IS_INVALID.
        </description>
        <equal_to var1="return" var2="__BioAPIERR_BSP_CURSOR_IS_INVALID "/>
     </assert_condition>
     <!-- Invoke the function BioSPI_DbClose with valid input parameters -->
     <invoke function="BioSPI_DbClose">
        <input name="ModuleHandle" var="_modulehandle"/>
        <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 activity="CleanUpDBTesting" package="be0a1330-d59e-11d8-9669-
0800200c9a66" break_on_break="true">
        <only_if>
          <same_as var1="supportDBCreation" value2="true"/>
        </only_if>
        <input name="ModuleHandle" var="_modulehandle"/>
```

```
        <input name="dbName" var="dbName"/>
      </invoke>
      <!-- Invoke the functions BioSPI_ModuleDetach and BioSPI_ModuleUnload -->
      <invoke activity="DetachAndUnload" package="be0a1330-d59e-11d8-9669-
0800200c9a66">
        <input name="moduleUuid" var="moduleUuid"/>
        <input name="module" var="_modulehandle"/>
      </invoke>
   </activity>
</package>
```

16.50 Feature 29. BioSPI Db Store BIR

16.50.1 BioAPI Specification References:
3.3.5.7

16.50.2 Assertion 29.1 BioSPI_DbStoreBIR_ValidParam

16.50.2.1 Test Purpose:
To test BioSPI_DbStoreBIR with valid parameters if supported by the BSP.

16.50.2.2 Test Scenario:
If supported by the BSP, call BioSPI_DbStoreBIR with valid parameters.

16.50.2.3 Expected Results:
Return code BioAPI_OK.

16.50.2.4 XML:

```
<package name="0509bfc8-0988-1085-82a3-0002a5d5fd2e">
   <author>Author</author>
   <description>
      This package contains the assertion "BioSPI_DbStoreBIR_ValidParam"
   </description>
   <assertion name="BioSPI_DbStoreBIR_ValidParam" model="BSPTesting">
      <description>
         This assertion invokes BioSPI_DbStoreBIR with valid input parameters and
verify if the return code is BioAPI_OK and the output UUID is not NULL.
         The relevant text in BioAPI 1.1 is quoted below from subclauses 2.5.4.7.
         ____________________________________________________________
         BioAPI_RETURN BioAPI BioAPI_DbStoreBIR
            (BioAPI_HANDLE ModuleHandle,
            const BioAPI_INPUT_BIR *BIRToStore,
            BioAPI_DB_HANDLE DbHandle,
            BioAPI_UUID_PTR Uuid);

            The BIR identified by the BIRToStore parameter is stored in the open
database identified by the DbHandle
                        parameter. If the BIRToStore is identified by a BIR
Handle, the input BIR Handle is freed. If the BIRToStore is identified by a
database key value, the BIR is copied to the open database. A new UUID is
assigned to the new BIR in the database, and this UUID can be used as a key
value to access the BIR later.
         ____________________________________________________________
         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_Enroll to generate a BIR.
          5) Invoke function BioSPI_DbStoreBIR to store the enrolled BIR into
database.
          6) Verify the return code, it is expected to be BioAPI_OK.
          7) Invoke function BioSPI_DbDelete to delete the database.
          8) Detach and unload the BSP.

        If any of the intermediate operations fail, an UNDECIDED conformity
response is issued.
     </description>
     <!-- UUID of the BSP under test -->
     <input name="_moduleUuid"/>
     <!-- Major version number of the BSP under test -->
     <input name="_moduleVersionMajor"/>
     <!-- Minor version number of the BSP under test -->
     <input name="_moduleVersionMinor"/>
     <!-- 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 capture -->
     <input name="_capturetimeout"/>
     <!-- Database Name to be opened -->
     <input name="_dbName"/>
     <!-- Indicates if the BSP support DB creation-->
     <input name="_supportDBCreation"/>
     <!-- Invocation of the primary activity of this assertion with input
parameter values assigned from the assertion's parameters. -->
     <invoke activity="BioSPI_DbStoreBIR">
        <input name="moduleUuid" var="_moduleUuid"/>
        <input name="moduleVersionMajor" var="_moduleVersionMajor"/>
        <input name="moduleVersionMinor" var="_moduleVersionMinor"/>
        <input name="inserttimeouttime" var="_inserttimeout"/>
        <input name="nosourcepresentsupported" var="_noSourcePresentSupported"/>
        <input name="sourcepresenttimeouttime" var="_sourcepresenttimeout"/>
        <input name="capturetimeout" var="_capturetimeout"/>
        <input name="dbName" var="_dbName"/>
        <input name="supportDBCreation" var="_supportDBCreation"/>
     </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="be0a1330-d59e-11d8-9669-
0800200c9a66" function="BioSPI_ModuleEventHandler"/>
   </assertion>
   <activity name="BioSPI_DbStoreBIR">
     <input name="moduleUuid"/>
     <input name="moduleVersionMajor"/>
     <input name="moduleVersionMinor"/>
     <input name="inserttimeouttime"/>
     <input name="nosourcepresentsupported"/>
     <input name="sourcepresenttimeouttime"/>
     <input name="capturetimeout"/>
     <input name="dbName"/>
     <input name="supportDBCreation"/>
     <!-- This assertion will use ModuleHandle "1" for all BioSPI calls that
require it -->
     <set name="_modulehandle" value="1"/>
```

```
     <!-- Invoke the functions BioSPI_ModuleLoad and BioSPI_ModuleAttach
exposed by the BSP under test.
          The input value for the parameter "deviceIDOrNull" is "0", therefore
the assertion will test the BSP's default device. -->
     <invoke activity="LoadAndAttach" package="be0a1330-d59e-11d8-9669-
0800200c9a66" break_on_break="true">
        <input name="moduleUuid" var="moduleUuid"/>
        <input name="moduleVersionMajor" var="moduleVersionMajor"/>
        <input name="moduleVersionMinor" var="moduleVersionMinor"/>
        <input name="deviceIDOrNull" value="0"/>
        <input name="module" var="_modulehandle"/>
        <input name="eventtimeouttime" var="inserttimeouttime"/>
     </invoke>
     <!-- If the BSP supports database creation, invoke BioSPI_DbDelete, in
preparation for the database creation -->
     <invoke function="BioSPI_DbDelete">
        <only_if>
           <same_as var1="supportDBCreation" value2="true"/>
        </only_if>
        <input name="ModuleHandle" var="_modulehandle"/>
        <input name="DbName" var="dbName"/>
        <return setvar="return"/>
     </invoke>
     <!-- Invoke BioSPI_DbCreate if the BSP supports database creation -->
     <invoke function="BioSPI_DbCreate">
        <only_if>
           <same_as var1="supportDBCreation" value2="true"/>
        </only_if>
        <input name="ModuleHandle" var="_modulehandle"/>
        <input name="DbName" var="dbName"/>
        <input name="ReadAccessRequest" value="true"/>
        <input name="WriteAccessRequest" value="true"/>
        <output name="DbHandle" setvar="dbHandle"/>
        <return setvar="return"/>
     </invoke>
     <!-- Open the specified database if database creation is not supported. --
>
     <invoke function="BioSPI_DbOpen">
        <only_if>
           <same_as var1="supportDBCreation" value2="false"/>
        </only_if>
        <input name="ModuleHandle" var="_modulehandle"/>
        <input name="DbName" var="dbName"/>
        <input name="ReadAccessRequest" value="true"/>
        <input name="WriteAccessRequest" value="true"/>
        <output name="DbHandle" setvar="dbHandle"/>
        <output name="Cursor" setvar="cursor"/>
        <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>
           If database creation is supported by the BSP, the function
BioSPI_DbCreate has returned BioAPI_OK and the output dbHandle is a valid DB
handle. If database creation is not supported by the BSP, 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>
     <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="ModuleHandle" var="_modulehandle"/>
        <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="ModuleHandle" var="_modulehandle"/>
        <input name="BIRToStore_Form" var="__BioAPI_BIR_HANDLE_INPUT"/>
        <input name="BIRToStore_BIRHandle" var="newtemplate_handle"/>
        <input name="DbHandle" var="dbHandle"/>
        <input name="no_Uuid" value="false"/>
        <output name="Uuid" 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>
        <description>
          The function BioSPI_DbStoreBIR has returned BioAPI_OK.
        </description>
        <equal_to var1="return" var2="__BioAPI_OK"/>
     </assert_condition>
     <!-- Invoke the function BioSPI_DbQueryBIR with valid input parameters -->
```

```
     <invoke function="BioSPI_DbQueryBIR">
        <input name="ModuleHandle" var="_modulehandle"/>
        <input name="DbHandle" var="dbHandle"/>
        <input name="BIRToQuery_Form" value="1"/>
        <input name="BIRToQuery_DbHandle" var="dbHandle"/>
        <input name="BIRToQuery_KeyValue" var="biruuid"/>
        <output name="Uuid" setvar="outputuuid"/>
        <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_DbQueryBIR has returned BioAPI_OK and the output
UUID points to the expected BIR.
        </description>
        <equal_to var1="return" var2="__BioAPI_OK"/>
        <same_as var1="outputuuid" var2="biruuid"/>
     </assert_condition>
     <!-- Invoke the function BioSPI_DbClose with valid input parameters -->
     <invoke function="BioSPI_DbClose">
        <input name="ModuleHandle" var="_modulehandle"/>
        <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 activity="CleanUpDBTesting" package="be0a1330-d59e-11d8-9669-
0800200c9a66" break_on_break="true">
        <only_if>
          <same_as var1="supportDBCreation" value2="true"/>
        </only_if>
        <input name="ModuleHandle" var="_modulehandle"/>
        <input name="dbName" var="dbName"/>
     </invoke>
     <!-- Invoke the functions BioSPI_ModuleDetach and BioSPI_ModuleUnload -->
     <invoke activity="DetachAndUnload" package="be0a1330-d59e-11d8-9669-
0800200c9a66">
        <input name="moduleUuid" var="moduleUuid"/>
        <input name="module" var="_modulehandle"/>
     </invoke>
   </activity>
</package>
```

16.50.3 Assertion 29.2 BioSPI_DbStoreBIR_InvalidModuleHandle

16.50.3.1 Test Purpose:
To test BioSPI_DbStoreBIR with ian nvalid module handle if supported by the BSP.

16.50.3.2 Test Scenario:
If supported by the BSP, call BioSPI_DbStoreBIR with an invalid module handle.

16.50.3.3 Expected Results:
Return code other than BioAPI_OK.

16.50.3.4 XML:

```
<package name="0352a5a0-098a-1085-be57-0002a5d5fd2e">
   <author>Author</author>
   <description>
      This package contains the assertion
"BioSPI_DbStoreBIR_InvalidModuleHandle"
   </description>
   <assertion name="BioSPI_DbStoreBIR_InvalidModuleHandle" model="BSPTesting">
      <description>
         This assertion invokes BioSPI_DbStoreBIR with invalid module handle and
verify if the return code is BioAPIERR_H_FRAMEWORK_INVALID_MODULE_HANDLE.
   The relevant text in BioAPI 1.1 is quoted below from subclauses 2.5.4.7.
         ____________________________________________________________
         BioAPI_RETURN BioAPI BioAPI_DbStoreBIR
            (BioAPI_HANDLE ModuleHandle,
            const BioAPI_INPUT_BIR *BIRToStore,
            BioAPI_DB_HANDLE DbHandle,
            BioAPI_UUID_PTR Uuid);

            The BIR identified by the BIRToStore parameter is stored in the open
database identified by the DbHandle
                        parameter. If the BIRToStore is identified by a BIR
Handle, the input BIR Handle is freed. If the BIRToStore is identified by a
database key value, the BIR is copied to the open database. A new UUID is
assigned to the new BIR in the database, and this UUID can be used as a key
value to access the BIR later.
         ____________________________________________________________
         and text from subclause 2.3.2.3
         ____________________________________________________________
         #define BioAPIERR_H_FRAMEWORK_INVALID_MODULE_HANDLE \
                      (BioAPI_H_FRAMEWORK_BASE_ERROR +
BioAPI_ERRORCODE_COMMON_EXTENT + 1)
         The given service provider handle is not valid
         ____________________________________________________________

         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_Enroll to generate a BIR.
            5) Invoke function BioSPI_DbStoreBIR to store the enrolled BIR into
database with an invalid module handle.
            6) Verify the return code, it is expected to be BioAPI_OK.
            7) Invoke function BioSPI_DbDelete to delete the database.
            8) Detach and unload the BSP.

         If any of the intermediate operations fail, an UNDECIDED conformity
response is issued.
      </description>
      <!-- UUID of the BSP under test -->
      <input name="_moduleUuid"/>
      <!-- Major version number of the BSP under test -->
      <input name="_moduleVersionMajor"/>
      <!-- Minor version number of the BSP under test -->
      <input name="_moduleVersionMinor"/>
      <!-- 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 capture -->
     <input name="_capturetimeout"/>
     <!-- Database Name to be opened -->
     <input name="_dbName"/>
     <!-- Indicates if the BSP support DB creation-->
     <input name="_supportDBCreation"/>
     <!-- Invocation of the primary activity of this assertion with input
parameter values assigned from the assertion's parameters. -->
     <invoke activity="BioSPI_DbStoreBIR">
        <input name="moduleUuid" var="_moduleUuid"/>
        <input name="moduleVersionMajor" var="_moduleVersionMajor"/>
        <input name="moduleVersionMinor" var="_moduleVersionMinor"/>
        <input name="inserttimeouttime" var="_inserttimeout"/>
        <input name="nosourcepresentsupported" var="_noSourcePresentSupported"/>
        <input name="sourcepresenttimeouttime" var="_sourcepresenttimeout"/>
        <input name="capturetimeout" var="_capturetimeout"/>
        <input name="dbName" var="_dbName"/>
        <input name="supportDBCreation" var="_supportDBCreation"/>
     </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="be0a1330-d59e-11d8-9669-
0800200c9a66" function="BioSPI_ModuleEventHandler"/>
   </assertion>
   <activity name="BioSPI_DbStoreBIR">
     <input name="moduleUuid"/>
     <input name="moduleVersionMajor"/>
     <input name="moduleVersionMinor"/>
     <input name="inserttimeouttime"/>
     <input name="nosourcepresentsupported"/>
     <input name="sourcepresenttimeouttime"/>
     <input name="capturetimeout"/>
     <input name="dbName"/>
     <input name="supportDBCreation"/>
     <!-- This assertion will use ModuleHandle "1" for all BioSPI calls that
require it -->
     <set name="_modulehandle" value="1"/>
     <!-- Invoke the functions BioSPI_ModuleLoad and BioSPI_ModuleAttach
exposed by the BSP under test.
          The input value for the parameter "deviceIDOrNull" is "0", therefore
the assertion will test the BSP's default device. -->
     <invoke activity="LoadAndAttach" package="be0a1330-d59e-11d8-9669-
0800200c9a66" break_on_break="true">
        <input name="moduleUuid" var="moduleUuid"/>
        <input name="moduleVersionMajor" var="moduleVersionMajor"/>
        <input name="moduleVersionMinor" var="moduleVersionMinor"/>
        <input name="deviceIDOrNull" value="0"/>
        <input name="module" var="_modulehandle"/>
        <input name="eventtimeouttime" var="inserttimeouttime"/>
     </invoke>
     <invoke activity="PrepareDBTesting" package="be0a1330-d59e-11d8-9669-
0800200c9a66" break_on_break="true">
        <only_if>
          <same_as var1="supportDBCreation" value2="true"/>
        </only_if>
        <input name="ModuleHandle" var="_modulehandle"/>
        <input name="dbName" var="dbName"/>
```

```
        <input name="nosourcepresentsupported" var="nosourcepresentsupported"/>
      </invoke>
      <!-- Invoke the function BioSPI_DbOpen to open the specified database. -->
      <invoke function="BioSPI_DbOpen">
        <input name="ModuleHandle" var="_modulehandle"/>
        <input name="DbName" var="dbName"/>
        <input name="ReadAccessRequest" value="true"/>
        <input name="WriteAccessRequest" value="true"/>
        <output name="DbHandle" setvar="dbHandle"/>
        <output name="Cursor" setvar="cursor"/>
        <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_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>
      <!-- The BSP is ready to capture. Invoke the function BioSPI_Enroll for
the purpose of enrollment. -->
      <invoke function="BioSPI_Enroll">
        <input name="ModuleHandle" var="_modulehandle"/>
        <input name="Purpose"
var="__BioAPI_PURPOSE_ENROLL_FOR_VERIFICATION_ONLY"/>
        <input name="Timeout" var="capturetimeout"/>
        <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 captured BIR into
database -->
      <invoke function="BioSPI_DbStoreBIR">
        <input name="ModuleHandle" value="0"/>
        <input name="BIRToStore_Form" var="__BioAPI_BIR_HANDLE_INPUT"/>
        <input name="BIRToStore_BIRHandle" var="newtemplate_handle"/>
        <input name="DbHandle" var="dbHandle"/>
        <input name="no_Uuid" value="false"/>
        <output name="Uuid" setvar="uuid"/>
        <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_DbStoreBIR has returned
BioAPIERR_H_FRAMEWORK_INVALID_MODULE_HANDLE.
```

```
        </description>
        <equal_to var1="return"
var2="__BioAPIERR_H_FRAMEWORK_INVALID_MODULE_HANDLE "/>
     </assert_condition>
     <!-- Invoke the function BioSPI_DbClose with valid input parameters -->
     <invoke function="BioSPI_DbClose">
        <input name="ModuleHandle" var="_modulehandle"/>
        <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 activity="CleanUpDBTesting" package="be0a1330-d59e-11d8-9669-
0800200c9a66" break_on_break="true">
        <only_if>
           <same_as var1="supportDBCreation" value2="true"/>
        </only_if>
        <input name="ModuleHandle" var="_modulehandle"/>
        <input name="dbName" var="dbName"/>
     </invoke>
     <!-- Invoke the functions BioSPI_ModuleDetach and BioSPI_ModuleUnload -->
     <invoke activity="DetachAndUnload" package="be0a1330-d59e-11d8-9669-
0800200c9a66">
        <input name="moduleUuid" var="moduleUuid"/>
        <input name="module" var="_modulehandle"/>
     </invoke>
   </activity>
</package>
```

16.50.4 Assertion 29.3 BioSPI_DbStoreBIR_InvalidDBHandle

16.50.4.1 Test Purpose:
To test BioSPI_DbStoreBIR with ian nvalid DB handle if supported by the BSP.

16.50.4.2 Test Scenario:
If supported by the BSP, call BioSPI_DbStoreBIR with an invalid DB handle.

16.50.4.3 Expected Results:
Return code other than BioAPI_OK.

16.50.4.4 XML:

16.51 Feature 30. BioSPI Db Get BIR

16.51.1 BioAPI Specification References:
3.3.5.8

16.51.2 Assertion 30.1 BioSPI_DbGetBIR_ValidParam

16.51.2.1 Test Purpose:
To test BioSPI_DbGetBIR with valid parameters if supported by the BSP.

16.51.2.2 Test Scenario:
If supported by the BSP, call BioSPI_DbGetBIR with valid parameters.

16.51.2.3 Expected Results:
Return code BioAPI_OK.

16.51.2.4 XML:

```
<package name="04ac0798-09a2-1085-87aa-0002a5d5fd2e">
   <author>Author</author>
   <description>
      This package contains the assertion "BioSPI_DbGetBIR_ValidParam"
   </description>
   <assertion name="BioSPI_DbGetBIR_ValidParam" model="BSPTesting">
      <description>
         This assertion invokes BioSPI_DbGetBIR with valid input parameters and
verify if the return code is BioAPI_OK.
         The relevant text in BioAPI 1.1 is quoted below from subclauses 2.5.4.8.
         ____________________________________________________________
         BioAPI_RETURN BioAPI BioAPI_DbGetBIR
         (BioAPI_HANDLE ModuleHandle,
         BioAPI_DB_HANDLE DbHandle,
         const BioAPI_UUID *KeyValue,
         BioAPI_BIR_HANDLE_PTR RetrievedBIR,
         BioAPI_DB_CURSOR_PTR Cursor);
         The BIR identified by the KeyValue parameter in the open database
identified by the DbHandle parameter is
                     retrieved.
         ____________________________________________________________
         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_Enroll to generate a BIR.
            5) Invoke function BioSPI_DbStoreBIR to store the enrolled BIR into
database.
            6) Invoke function BioSPI_DbGetBIR to retrieve the stored BIR.
            7) Verify the return code, it is expected to be BioAPI_OK.
            8) Invoke function BioSPI_DbDelete to delete the database.
            9) Detach and unload the BSP.

         If any of the intermediate operations fail, an UNDECIDED conformity
response is issued.
      </description>
      <!-- UUID of the BSP under test -->
      <input name="_moduleUuid"/>
      <!-- Major version number of the BSP under test -->
      <input name="_moduleVersionMajor"/>
      <!-- Minor version number of the BSP under test -->
      <input name="_moduleVersionMinor"/>
      <!-- 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 capture -->
      <input name="_capturetimeout"/>
      <!-- Database Name to be opened -->
      <input name="_dbName"/>
      <!-- Indicates if the BSP support DB creation-->
      <input name="_supportDBCreation"/>
      <!-- Invocation of the primary activity of this assertion with input
parameter values assigned from the assertion's parameters. -->
      <invoke activity="BioSPI_DbGetBIR">
         <input name="moduleUuid" var="_moduleUuid"/>
         <input name="moduleVersionMajor" var="_moduleVersionMajor"/>
         <input name="moduleVersionMinor" var="_moduleVersionMinor"/>
         <input name="inserttimeouttime" var="_inserttimeout"/>
         <input name="nosourcepresentsupported" var="_noSourcePresentSupported"/>
         <input name="sourcepresenttimeouttime" var="_sourcepresenttimeout"/>
         <input name="capturetimeout" var="_capturetimeout"/>
         <input name="dbName" var="_dbName"/>
         <input name="supportDBCreation" var="_supportDBCreation"/>
      </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="be0a1330-d59e-11d8-9669-
0800200c9a66" function="BioSPI_ModuleEventHandler"/>
   </assertion>
   <activity name="BioSPI_DbGetBIR">
      <input name="moduleUuid"/>
      <input name="moduleVersionMajor"/>
      <input name="moduleVersionMinor"/>
      <input name="inserttimeouttime"/>
      <input name="nosourcepresentsupported"/>
      <input name="sourcepresenttimeouttime"/>
      <input name="capturetimeout"/>
      <input name="dbName"/>
      <input name="supportDBCreation"/>
      <!-- This assertion will use ModuleHandle "1" for all BioSPI calls that
require it -->
      <set name="_modulehandle" value="1"/>
      <!-- Invoke the functions BioSPI_ModuleLoad and BioSPI_ModuleAttach
exposed by the BSP under test.
            The input value for the parameter "deviceIDOrNull" is "0", therefore
the assertion will test the BSP's default device. -->
      <invoke activity="LoadAndAttach" package="be0a1330-d59e-11d8-9669-
0800200c9a66" break_on_break="true">
         <input name="moduleUuid" var="moduleUuid"/>
         <input name="moduleVersionMajor" var="moduleVersionMajor"/>
         <input name="moduleVersionMinor" var="moduleVersionMinor"/>
         <input name="deviceIDOrNull" value="0"/>
         <input name="module" var="_modulehandle"/>
         <input name="eventtimeouttime" var="inserttimeouttime"/>
      </invoke>
      <invoke activity="PrepareDBTesting" package="be0a1330-d59e-11d8-9669-
0800200c9a66" break_on_break="true">
         <only_if>
            <same_as var1="supportDBCreation" value2="true"/>
         </only_if>
         <input name="ModuleHandle" var="_modulehandle"/>
         <input name="dbName" var="dbName"/>
         <input name="nosourcepresentsupported" var="nosourcepresentsupported"/>
      </invoke>
      <!-- Invoke the function BioSPI_DbOpen to open the specified database. -->
```

```
     <invoke function="BioSPI_DbOpen">
        <input name="ModuleHandle" var="_modulehandle"/>
        <input name="DbName" var="dbName"/>
        <input name="ReadAccessRequest" value="true"/>
        <input name="WriteAccessRequest" value="true"/>
        <output name="DbHandle" setvar="dbHandle"/>
        <output name="Cursor" setvar="cursor"/>
        <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_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>
     <!-- The BSP is ready to capture. Invoke the function BioSPI_Enroll for
the purpose of enrollment. -->
     <invoke function="BioSPI_Enroll">
        <input name="ModuleHandle" var="_modulehandle"/>
        <input name="Purpose"
var="__BioAPI_PURPOSE_ENROLL_FOR_VERIFICATION_ONLY"/>
        <input name="Timeout" var="capturetimeout"/>
        <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="ModuleHandle" var="_modulehandle"/>
        <input name="BIRToStore_Form" var="__BioAPI_BIR_HANDLE_INPUT"/>
        <input name="BIRToStore_BIRHandle" var="newtemplate_handle"/>
        <input name="DbHandle" var="dbHandle"/>
        <input name="no_Uuid" value="false"/>
        <output name="Uuid" setvar="uuid"/>
        <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_DbStoreBIR has returned BioAPI_OK.
        </description>
        <equal_to var1="return" var2="__BioAPI_OK"/>
     </assert_condition>
     <!-- Invoke the function BioSPI_DbGetBIR with valid input parameters -->
```

```
     <invoke function="BioSPI_DbGetBIR">
        <input name="ModuleHandle" var="_modulehandle"/>
        <input name="DbHandle" var="dbHandle"/>
        <input name="KeyValue" var="uuid"/>
        <output name="RetrievedBIR" setvar="retrievedBir_handle"/>
        <output name="Cursor" setvar="cursor"/>
        <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_DbGetBIR 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="ModuleHandle" var="_modulehandle"/>
        <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 activity="CleanUpDBTesting" package="be0a1330-d59e-11d8-9669-
0800200c9a66" break_on_break="true">
        <only_if>
           <same_as var1="supportDBCreation" value2="true"/>
        </only_if>
        <input name="ModuleHandle" var="_modulehandle"/>
        <input name="dbName" var="dbName"/>
     </invoke>
     <!-- Invoke the functions BioSPI_ModuleDetach and BioSPI_ModuleUnload -->
     <invoke activity="DetachAndUnload" package="be0a1330-d59e-11d8-9669-
0800200c9a66">
        <input name="moduleUuid" var="moduleUuid"/>
        <input name="module" var="_modulehandle"/>
     </invoke>
   </activity>
</package>
```

16.51.3 Assertion 30.2 BioSPI_DbGetBIR_RecordNotFound

16.51.3.1 Test Purpose:
To test BioSPI_DbGetBIR "record not found" if supported by the BSP.

16.51.3.2 Test Scenario:
If supported by the BSP, call BioSPI_DbGetBIR with valid parameters and KeyValue not in the DbHandle database.

16.51.3.3 Expected Results:

Return code BioAPIERR_BSP_RECORD_NOT_FOUND.

16.51.3.4 XML:

```
<package name="0370eba0-09ab-1085-8bb8-0002a5d5fd2e">
   <author>Author</author>
   <description>
      This package contains the assertion "BioSPI_DbGetBIR_RecordNotFound"
   </description>
   <assertion name="BioSPI_DbGetBIR_RecordNotFound" model="BSPTesting">
      <description>
         This assertion invokes BioSPI_DbGetBIR with a key value that does not
exist in the database and verify if the return code is
BioAPIERR_BSP_RECORD_NOT_FOUND.
         The relevant text in BioAPI 1.1 is quoted below from subclauses 2.5.4.8.
         ____________________________________________________________
         BioAPI_RETURN BioAPI BioAPI_DbGetBIR
         (BioAPI_HANDLE ModuleHandle,
         BioAPI_DB_HANDLE DbHandle,
         const BioAPI_UUID *KeyValue,
         BioAPI_BIR_HANDLE_PTR RetrievedBIR,
         BioAPI_DB_CURSOR_PTR Cursor);
         The BIR identified by the KeyValue parameter in the open database
identified by the DbHandle parameter is
                      retrieved.
         Errors
            BioAPIERR_BSP_RECORD_NOT_FOUND
         ____________________________________________________________
         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_Enroll to generate a BIR.
            5) Invoke function BioSPI_DbStoreBIR to store the enrolled BIR into
database.
            6) Invoke function BioSPI_DbGetBIR with a non-existing keyvalue in
the database.
            7) Verify the return code, it is expected to be
BioAPIERR_BSP_RECORD_NOT_FOUND.
            8) Invoke function BioSPI_DbDelete to delete the database.
            9) Detach and unload the BSP.

         If any of the intermediate operations fail, an UNDECIDED conformity
response is issued.
      </description>
      <!-- UUID of the BSP under test -->
      <input name="_moduleUuid"/>
      <!-- Major version number of the BSP under test -->
      <input name="_moduleVersionMajor"/>
      <!-- Minor version number of the BSP under test -->
      <input name="_moduleVersionMinor"/>
      <!-- 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 capture -->
      <input name="_capturetimeout"/>
      <!-- Database Name to be opened -->
```

```
      <input name="_dbName"/>
      <!-- Indicates if the BSP support DB creation-->
      <input name="_supportDBCreation"/>
      <!-- Invocation of the primary activity of this assertion with input
parameter values assigned from the assertion's parameters. -->
      <invoke activity="BioSPI_DbGetBIR">
         <input name="moduleUuid" var="_moduleUuid"/>
         <input name="moduleVersionMajor" var="_moduleVersionMajor"/>
         <input name="moduleVersionMinor" var="_moduleVersionMinor"/>
         <input name="inserttimeouttime" var="_inserttimeout"/>
         <input name="nosourcepresentsupported" var="_noSourcePresentSupported"/>
         <input name="sourcepresenttimeouttime" var="_sourcepresenttimeout"/>
         <input name="capturetimeout" var="_capturetimeout"/>
         <input name="dbName" var="_dbName"/>
         <input name="supportDBCreation" var="_supportDBCreation"/>
      </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="be0a1330-d59e-11d8-9669-
0800200c9a66" function="BioSPI_ModuleEventHandler"/>
   </assertion>
   <activity name="BioSPI_DbGetBIR">
      <input name="moduleUuid"/>
      <input name="moduleVersionMajor"/>
      <input name="moduleVersionMinor"/>
      <input name="inserttimeouttime"/>
      <input name="nosourcepresentsupported"/>
      <input name="sourcepresenttimeouttime"/>
      <input name="capturetimeout"/>
      <input name="dbName"/>
      <input name="supportDBCreation"/>
      <!-- This assertion will use ModuleHandle "1" for all BioSPI calls that
require it -->
      <set name="_modulehandle" value="1"/>
      <!-- Invoke the functions BioSPI_ModuleLoad and BioSPI_ModuleAttach
exposed by the BSP under test.
           The input value for the parameter "deviceIDOrNull" is "0", therefore
the assertion will test the BSP's default device. -->
      <invoke activity="LoadAndAttach" package="be0a1330-d59e-11d8-9669-
0800200c9a66" break_on_break="true">
         <input name="moduleUuid" var="moduleUuid"/>
         <input name="moduleVersionMajor" var="moduleVersionMajor"/>
         <input name="moduleVersionMinor" var="moduleVersionMinor"/>
         <input name="deviceIDOrNull" value="0"/>
         <input name="module" var="_modulehandle"/>
         <input name="eventtimeouttime" var="inserttimeouttime"/>
      </invoke>
      <invoke activity="PrepareDBTesting" package="be0a1330-d59e-11d8-9669-
0800200c9a66" break_on_break="true">
         <only_if>
            <same_as var1="supportDBCreation" value2="true"/>
         </only_if>
         <input name="ModuleHandle" var="_modulehandle"/>
         <input name="dbName" var="dbName"/>
         <input name="nosourcepresentsupported" var="nosourcepresentsupported"/>
      </invoke>
      <!-- Invoke the function BioSPI_DbOpen to open the specified database. -->
      <invoke function="BioSPI_DbOpen">
         <input name="ModuleHandle" var="_modulehandle"/>
         <input name="DbName" var="dbName"/>
         <input name="ReadAccessRequest" value="true"/>
         <input name="WriteAccessRequest" value="true"/>
```

```
        <output name="DbHandle" setvar="dbHandle"/>
        <output name="Cursor" setvar="cursor"/>
        <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_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>
     <!-- The BSP is ready to capture. Invoke the function BioSPI_Enroll for
the purpose of enrollment. -->
     <invoke function="BioSPI_Enroll">
        <input name="ModuleHandle" var="_modulehandle"/>
        <input name="Purpose"
var="__BioAPI_PURPOSE_ENROLL_FOR_VERIFICATION_ONLY"/>
        <input name="Timeout" var="capturetimeout"/>
        <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="ModuleHandle" var="_modulehandle"/>
        <input name="BIRToStore_Form" var="__BioAPI_BIR_HANDLE_INPUT"/>
        <input name="BIRToStore_BIRHandle" var="newtemplate_handle"/>
        <input name="DbHandle" var="dbHandle"/>
        <input name="no_Uuid" value="false"/>
        <output name="Uuid" setvar="uuid"/>
        <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_DbStoreBIR has returned BioAPI_OK.
        </description>
        <equal_to var1="return" var2="__BioAPI_OK"/>
     </assert_condition>
     <!-- Invoke the function BioSPI_DbGetBIR  -->
     <invoke function="BioSPI_DbGetBIR">
        <input name="ModuleHandle" var="_modulehandle"/>
        <input name="DbHandle" var="dbHandle"/>
        <input name="KeyValue" value="ffffffff-ffff-ffff-ffff-ffffffffffff"/>
        <output name="RetrievedBIR" setvar="retrievedBir_handle"/>
```

```
        <output name="Cursor" setvar="cursor"/>
        <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_DbGetBIR has returned
BioAPIERR_BSP_RECORD_NOT_FOUND.
        </description>
        <equal_to var1="return" var2="__BioAPIERR_BSP_RECORD_NOT_FOUND "/>
      </assert_condition>
      <!-- Invoke the function BioSPI_DbClose with valid input parameters -->
      <invoke function="BioSPI_DbClose">
        <input name="ModuleHandle" var="_modulehandle"/>
        <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 activity="CleanUpDBTesting" package="be0a1330-d59e-11d8-9669-
0800200c9a66" break_on_break="true">
        <only_if>
           <same_as var1="supportDBCreation" value2="true"/>
        </only_if>
        <input name="ModuleHandle" var="_modulehandle"/>
        <input name="dbName" var="dbName"/>
      </invoke>
      <!-- Invoke the functions BioSPI_ModuleDetach and BioSPI_ModuleUnload -->
      <invoke activity="DetachAndUnload" package="be0a1330-d59e-11d8-9669-
0800200c9a66">
        <input name="moduleUuid" var="moduleUuid"/>
        <input name="module" var="_modulehandle"/>
      </invoke>
   </activity>
</package>
```

16.51.4 Assertion 30.3 BioSPI_DbGetBIR_InvalidModuleHandle

16.51.4.1 Test Purpose:
To test BioSPI_DbGetBIR with an invalid module handle if supported by the BSP.

16.51.4.2 Test Scenario:
If supported by the BSP, call BioSPI_DbGetBIR with an invalid module handle.

16.51.4.3 Expected Results:
Return code other than BioAPI_OK.

16.51.4.4 XML:

```
<package name="01067768-09a4-1085-8ff8-0002a5d5fd2e">
   <author>Author</author>
```

```
   <description>
      This package contains the assertion "BioSPI_DbGetBIR_InvalidModuleHandle"
   </description>
   <assertion name="BioSPI_DbGetBIR_InvalidModuleHandle" model="BSPTesting">
      <description>
         This assertion invokes BioSPI_DbGetBIR with invalid module handle and
verify if the return code is BioAPIERR_H_FRAMEWORK_INVALID_MODULE_HANDLE.
         The relevant text in BioAPI 1.1 is quoted below from subclauses 2.5.4.8.
         ____________________________________________________________
         BioAPI_RETURN BioAPI BioAPI_DbGetBIR
         (BioAPI_HANDLE ModuleHandle,
         BioAPI_DB_HANDLE DbHandle,
         const BioAPI_UUID *KeyValue,
         BioAPI_BIR_HANDLE_PTR RetrievedBIR,
         BioAPI_DB_CURSOR_PTR Cursor);
         The BIR identified by the KeyValue parameter in the open database
identified by the DbHandle parameter is
                       retrieved.
         ____________________________________________________________
         and text from subclause 2.3.2.3
         ____________________________________________________________
         #define BioAPIERR_H_FRAMEWORK_INVALID_MODULE_HANDLE \
                       (BioAPI_H_FRAMEWORK_BASE_ERROR +
BioAPI_ERRORCODE_COMMON_EXTENT + 1)
         The given service provider handle is not valid
         ____________________________________________________________

         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_Enroll to generate a BIR.
            5) Invoke function BioSPI_DbStoreBIR to store the enrolled BIR into
database.
            6) Invoke function BioSPI_DbGetBIR to retrieve the stored BIR.
            7) Verify the return code, it is expected to be BioAPI_OK.
            8) Invoke function BioSPI_DbDelete to delete the database.
            9) Detach and unload the BSP.

         If any of the intermediate operations fail, an UNDECIDED conformity
response is issued.
      </description>
      <!-- UUID of the BSP under test -->
      <input name="_moduleUuid"/>
      <!-- Major version number of the BSP under test -->
      <input name="_moduleVersionMajor"/>
      <!-- Minor version number of the BSP under test -->
      <input name="_moduleVersionMinor"/>
      <!-- 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 capture -->
      <input name="_capturetimeout"/>
      <!-- Database Name to be opened -->
      <input name="_dbName"/>
      <!-- Indicates if the BSP support DB creation-->
      <input name="_supportDBCreation"/>
```

```
     <!-- Invocation of the primary activity of this assertion with input
parameter values assigned from the assertion's parameters. -->
     <invoke activity="BioSPI_DbGetBIR">
        <input name="moduleUuid" var="_moduleUuid"/>
        <input name="moduleVersionMajor" var="_moduleVersionMajor"/>
        <input name="moduleVersionMinor" var="_moduleVersionMinor"/>
        <input name="inserttimeouttime" var="_inserttimeout"/>
        <input name="nosourcepresentsupported" var="_noSourcePresentSupported"/>
        <input name="sourcepresenttimeouttime" var="_sourcepresenttimeout"/>
        <input name="capturetimeout" var="_capturetimeout"/>
        <input name="dbName" var="_dbName"/>
        <input name="supportDBCreation" var="_supportDBCreation"/>
     </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="be0a1330-d59e-11d8-9669-
0800200c9a66" function="BioSPI_ModuleEventHandler"/>
   </assertion>
   <activity name="BioSPI_DbGetBIR">
     <input name="moduleUuid"/>
     <input name="moduleVersionMajor"/>
     <input name="moduleVersionMinor"/>
     <input name="inserttimeouttime"/>
     <input name="nosourcepresentsupported"/>
     <input name="sourcepresenttimeouttime"/>
     <input name="capturetimeout"/>
     <input name="dbName"/>
     <input name="supportDBCreation"/>
     <!-- This assertion will use ModuleHandle "1" for all BioSPI calls that
require it -->
     <set name="_modulehandle" value="1"/>
     <!-- Invoke the functions BioSPI_ModuleLoad and BioSPI_ModuleAttach
exposed by the BSP under test.
          The input value for the parameter "deviceIDOrNull" is "0", therefore
the assertion will test the BSP's default device. -->
     <invoke activity="LoadAndAttach" package="be0a1330-d59e-11d8-9669-
0800200c9a66" break_on_break="true">
        <input name="moduleUuid" var="moduleUuid"/>
        <input name="moduleVersionMajor" var="moduleVersionMajor"/>
        <input name="moduleVersionMinor" var="moduleVersionMinor"/>
        <input name="deviceIDOrNull" value="0"/>
        <input name="module" var="_modulehandle"/>
        <input name="eventtimeouttime" var="inserttimeouttime"/>
     </invoke>
     <invoke activity="PrepareDBTesting" package="be0a1330-d59e-11d8-9669-
0800200c9a66" break_on_break="true">
        <only_if>
           <same_as var1="supportDBCreation" value2="true"/>
        </only_if>
        <input name="ModuleHandle" var="_modulehandle"/>
        <input name="dbName" var="dbName"/>
        <input name="nosourcepresentsupported" var="nosourcepresentsupported"/>
     </invoke>
     <!-- Invoke the function BioSPI_DbOpen to open the specified database. -->
     <invoke function="BioSPI_DbOpen">
        <input name="ModuleHandle" var="_modulehandle"/>
        <input name="DbName" var="dbName"/>
        <input name="ReadAccessRequest" value="true"/>
        <input name="WriteAccessRequest" value="true"/>
        <output name="DbHandle" setvar="dbHandle"/>
        <output name="Cursor" setvar="cursor"/>
        <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_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>
     <!-- The BSP is ready to capture. Invoke the function BioSPI_Enroll for
the purpose of enrollment. -->
     <invoke function="BioSPI_Enroll">
        <input name="ModuleHandle" var="_modulehandle"/>
        <input name="Purpose"
var="__BioAPI_PURPOSE_ENROLL_FOR_VERIFICATION_ONLY"/>
        <input name="Timeout" var="capturetimeout"/>
        <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="ModuleHandle" var="_modulehandle"/>
        <input name="BIRToStore_Form" var="__BioAPI_BIR_HANDLE_INPUT"/>
        <input name="BIRToStore_BIRHandle" var="newtemplate_handle"/>
        <input name="DbHandle" var="dbHandle"/>
        <input name="no_Uuid" value="false"/>
        <output name="Uuid" setvar="uuid"/>
        <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_DbStoreBIR has returned BioAPI_OK.
        </description>
        <equal_to var1="return" var2="__BioAPI_OK"/>
     </assert_condition>
     <!-- Invoke the function BioSPI_DbGetBIR with valid input parameters -->
     <invoke function="BioSPI_DbGetBIR">
        <input name="ModuleHandle" value="0"/>
        <input name="DbHandle" var="dbHandle"/>
        <input name="KeyValue" var="uuid"/>
        <output name="RetrievedBIR" setvar="retrievedBir_handle"/>
        <output name="Cursor" setvar="cursor"/>
        <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_DbGetBIR has returned
BioAPIERR_H_FRAMEWORK_INVALID_MODULE_HANDLE.
        </description>
        <equal_to var1="return"
var2="__BioAPIERR_H_FRAMEWORK_INVALID_MODULE_HANDLE "/>
     </assert_condition>
     <!-- Invoke the function BioSPI_DbClose with valid input parameters -->
     <invoke function="BioSPI_DbClose">
        <input name="ModuleHandle" var="_modulehandle"/>
        <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 activity="CleanUpDBTesting" package="be0a1330-d59e-11d8-9669-
0800200c9a66" break_on_break="true">
        <only_if>
          <same_as var1="supportDBCreation" value2="true"/>
        </only_if>
        <input name="ModuleHandle" var="_modulehandle"/>
        <input name="dbName" var="dbName"/>
     </invoke>
     <!-- Invoke the functions BioSPI_ModuleDetach and BioSPI_ModuleUnload -->
     <invoke activity="DetachAndUnload" package="be0a1330-d59e-11d8-9669-
0800200c9a66">
        <input name="moduleUuid" var="moduleUuid"/>
        <input name="module" var="_modulehandle"/>
     </invoke>
   </activity>
</package>
```

16.51.5 Assertion 30.4 BioSPI_DbGetBIR_InvalidDBHandle

16.51.5.1 Test Purpose:
To test BioSPI_DbGetBIR with an invalid DB handle if supported by the BSP.

16.51.5.2 Test Scenario:
If supported by the BSP, call BioSPI_DbGetBIR with an invalid DB handle.

16.51.5.3 Expected Results:
Return code other than BioAPI_OK.

16.51.5.4 XML:

16.51.6 Assertion 30.5 BioSPI_DbGetBIR_InvalidKeyValue

16.51.6.1 Test Purpose:
To test BioSPI_DbGetBIR with an invalid key value if supported by the BSP.

16.51.6.2 Test Scenario:
If supported by the BSP, call BioSPI_DbGetBIR with an invalid key value.

16.51.6.3 Expected Results:
Return code other than BioAPI_OK.

16.51.6.4 XML:

16.52 Feature 31. BioSPI Db Get Next BIR

16.52.1 BioAPI Specification References:
3.3.5.9

16.52.2 Assertion 31.1 BioSPI_DbGetNextBIR_ValidParam

16.52.2.1 Test Purpose:
To test BioSPI_DbGetNextBIR with valid parameters if supported by the BSP.

16.52.2.2 Test Scenario:
If supported by the BSP, call BioSPI_DbGetNextBIR with valid parameters.

16.52.2.3 Expected Results:
Return code BioAPI_OK.

16.52.2.4 XML:

```
<package name="038f3d58-09ac-1085-a5f6-0002a5d5fd2e">
   <author>Author</author>
   <description>
      This package contains the assertion "BioSPI_DbGetNextBIR_ValidParam"
   </description>
   <assertion name="BioSPI_DbGetNextBIR_ValidParam" model="BSPTesting">
      <description>
         This assertion invokes BioSPI_DbGetNextBIR with valid input parameters
and verify if the return code is BioAPI_OK.
         The relevant text in BioAPI 1.1 is quoted below from subclauses 2.5.4.9.
         ____________________________________________________________
         BioAPI_RETURN BioAPI BioAPI_DbGetNextBIR
         (BioAPI_HANDLE ModuleHandle,
         BioAPI_DB_CURSOR_PTR Cursor,
         BioAPI_BIR_HANDLE_PTR RetievedBIR,
         BioAPI_UUID_PTR Uuid);
         The BIR identified by the Cursor parameter is retrieved. The
         Cursor is updated to the next record in the database, or to
         the first when the end of the database is reached.
         ____________________________________________________________
         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 activity PrepareDBTesting to create a DB and store a
BIR
          into it.
          4) Invoke the function BioSPI_Enroll to generate a BIR.
          5) Invoke the function BioSPI_DbStoreBIR to store the enrolled BIR
into database.
          6) Invoke the function BioSPI_DbSetCursor to set the cursor to the
first record.
          7) Invoke the function BioSPI_DbGetNextBIR to retrieve the stored
BIR.
          8) Verify the return code, it is expected to be BioAPI_OK or
BioAPIERR_BSP_END_OF_DATABASE.
          9)  Invoke function BioSPI_DbGetNextBIR to retrieve the next BIR, it
is expected to be BioAPI_OK or BioAPIERR_BSP_END_OF_DATABASE.
          10)  Invoke the activity CleanUpDBTesting to delete the database.
          11)  Detach and unload the BSP.

        If any of the intermediate operations fail, an UNDECIDED conformity
response is issued.
     </description>
     <!-- UUID of the BSP under test -->
     <input name="_moduleUuid"/>
     <!-- Major version number of the BSP under test -->
     <input name="_moduleVersionMajor"/>
     <!-- Minor version number of the BSP under test -->
     <input name="_moduleVersionMinor"/>
     <!-- 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 capture -->
     <input name="_capturetimeout"/>
     <!-- Database Name to be opened -->
     <input name="_dbName"/>
     <!-- Indicates if the BSP support DB creation-->
     <input name="_supportDBCreation"/>
     <!-- Invocation of the primary activity of this assertion with input
parameter values assigned from the assertion's parameters. -->
     <invoke activity="BioSPI_DbGetNextBIR">
        <input name="moduleUuid" var="_moduleUuid"/>
        <input name="moduleVersionMajor" var="_moduleVersionMajor"/>
        <input name="moduleVersionMinor" var="_moduleVersionMinor"/>
        <input name="inserttimeouttime" var="_inserttimeout"/>
        <input name="nosourcepresentsupported" var="_noSourcePresentSupported"/>
        <input name="sourcepresenttimeouttime" var="_sourcepresenttimeout"/>
        <input name="capturetimeout" var="_capturetimeout"/>
        <input name="dbName" var="_dbName"/>
        <input name="supportDBCreation" var="_supportDBCreation"/>
     </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="be0a1330-d59e-11d8-9669-
0800200c9a66" function="BioSPI_ModuleEventHandler"/>
   </assertion>
   <activity name="BioSPI_DbGetNextBIR">
     <input name="moduleUuid"/>
     <input name="moduleVersionMajor"/>
     <input name="moduleVersionMinor"/>
     <input name="inserttimeouttime"/>
```

```
     <input name="nosourcepresentsupported"/>
     <input name="sourcepresenttimeouttime"/>
     <input name="capturetimeout"/>
     <input name="dbName"/>
     <input name="supportDBCreation"/>
     <!-- This assertion will use ModuleHandle "1" for all BioSPI calls that
require it -->
     <set name="_modulehandle" value="1"/>
     <!-- Invoke the functions BioSPI_ModuleLoad and BioSPI_ModuleAttach
exposed by the BSP under test.
          The input value for the parameter "deviceIDOrNull" is "0", therefore
the assertion will test the BSP's default device. -->
     <invoke activity="LoadAndAttach" package="be0a1330-d59e-11d8-9669-
0800200c9a66" break_on_break="true">
        <input name="moduleUuid" var="moduleUuid"/>
        <input name="moduleVersionMajor" var="moduleVersionMajor"/>
        <input name="moduleVersionMinor" var="moduleVersionMinor"/>
        <input name="deviceIDOrNull" value="0"/>
        <input name="module" var="_modulehandle"/>
        <input name="eventtimeouttime" var="inserttimeouttime"/>
     </invoke>
     <invoke activity="PrepareDBTesting" package="be0a1330-d59e-11d8-9669-
0800200c9a66" break_on_break="true">
        <only_if>
           <same_as var1="supportDBCreation" value2="true"/>
        </only_if>
        <input name="ModuleHandle" var="_modulehandle"/>
        <input name="dbName" var="dbName"/>
        <input name="nosourcepresentsupported" var="nosourcepresentsupported"/>
        <output name="biruuid" setvar="biruuid"/>
     </invoke>
     <!-- Invoke the function BioSPI_DbOpen to open the specified database. -->
     <invoke function="BioSPI_DbOpen">
        <input name="ModuleHandle" var="_modulehandle"/>
        <input name="DbName" var="dbName"/>
        <input name="ReadAccessRequest" value="true"/>
        <input name="WriteAccessRequest" value="true"/>
        <output name="DbHandle" setvar="dbHandle"/>
        <output name="Cursor" setvar="cursor"/>
        <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_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>
     <!-- The BSP is ready to capture. Invoke the function BioSPI_Enroll for
the purpose of enrollment. -->
     <invoke function="BioSPI_Enroll">
        <input name="ModuleHandle" var="_modulehandle"/>
        <input name="Purpose"
var="__BioAPI_PURPOSE_ENROLL_FOR_VERIFICATION_ONLY"/>
        <input name="Timeout" var="capturetimeout"/>
        <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="ModuleHandle" var="_modulehandle"/>
        <input name="BIRToStore_Form" var="__BioAPI_BIR_HANDLE_INPUT"/>
        <input name="BIRToStore_BIRHandle" var="newtemplate_handle"/>
        <input name="DbHandle" var="dbHandle"/>
        <input name="no_Uuid" value="false"/>
        <output name="Uuid" setvar="uuid1"/>
        <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_DbStoreBIR has returned BioAPI_OK.
        </description>
        <equal_to var1="return" var2="__BioAPI_OK"/>
     </assert_condition>
     <!-- Invoke function BioSPI_DbSetCursor with valid input parameters -->
     <invoke function="BioSPI_DbSetCursor">
        <input name="ModuleHandle" var="_modulehandle"/>
        <input name="DbHandle" var="dbHandle"/>
        <input name="KeyValue" var="uuid1"/>
        <output name="Cursor" setvar="cursor"/>
        <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_DbSetCursor has returned BioAPI_OK
        </description>
        <equal_to var1="return" var2="__BioAPI_OK"/>
     </assert_condition>
     <!-- Invoke the function BioSPI_DbGetNextBIR with valid input parameters -
->
     <invoke function="BioSPI_DbGetNextBIR">
        <input name="ModuleHandle" var="_modulehandle"/>
        <input name="Cursor" var="cursor"/>
        <output name="RetrievedBIR" setvar="retrievedBir_handle"/>
        <output name="Uuid" setvar="uuid"/>
        <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_DbGetNextBIR has returned either BioAPI_OK or
BioAPIERR_BSP_END_OF_DATABASE depending on the order in which the BIRs are
stored and the output Uuid is the same as the Uuid of the BIR pointed by the
cursor.
        </description>
        <and>
           <or>
              <equal_to var1="return" var2="__BioAPI_OK"/>
              <equal_to var1="return" var2="__BioAPIERR_BSP_END_OF_DATABASE"/>
           </or>
           <same_as var1="uuid" var2="uuid1"/>
        </and>
     </assert_condition>
     <invoke function="BioSPI_DbGetNextBIR">
        <input name="ModuleHandle" var="_modulehandle"/>
        <input name="Cursor" var="cursor"/>
        <output name="RetrievedBIR" setvar="retrievedBir_handle"/>
        <output name="Uuid" setvar="uuid"/>
        <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_DbGetNextBIR has returned either BioAPI_OK or
BioAPIERR_BSP_END_OF_DATABASE depending on the order in which the BIRs are
stored and the output Uuid is the same as the Uuid of the BIR pointed by the
cursor.
        </description>
        <and>
           <or>
              <equal_to var1="return" var2="__BioAPI_OK"/>
              <equal_to var1="return" var2="__BioAPIERR_BSP_END_OF_DATABASE"/>
           </or>
           <same_as var1="uuid" var2="biruuid"/>
        </and>
     </assert_condition>
     <!-- Invoke the function BioSPI_DbClose with valid input parameters -->
     <invoke function="BioSPI_DbClose">
        <input name="ModuleHandle" var="_modulehandle"/>
        <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 activity="CleanUpDBTesting" package="be0a1330-d59e-11d8-9669-
0800200c9a66" break_on_break="true">
        <only_if>
           <same_as var1="supportDBCreation" value2="true"/>
        </only_if>
        <input name="ModuleHandle" var="_modulehandle"/>
        <input name="dbName" var="dbName"/>
```

```
     </invoke>
     <!-- Invoke the functions BioSPI_ModuleDetach and BioSPI_ModuleUnload -->
     <invoke activity="DetachAndUnload" package="be0a1330-d59e-11d8-9669-
0800200c9a66">
        <input name="moduleUuid" var="moduleUuid"/>
        <input name="module" var="_modulehandle"/>
     </invoke>
   </activity>
</package>
```

16.52.3 Assertion 31.2 BioSPI_DbGetNextBIR_InvalidModuleHandle

16.52.3.1 Test Purpose:
To test BioSPI_DbGetNextBIR with an nvalid module handle if supported by the BSP.

16.52.3.2 Test Scenario:
If supported by the BSP, call BioSPI_DbGetNextBIR with an invalid module handle.

16.52.3.3 Expected Results:
Return code other than BioAPI_OK.

16.52.3.4 XML:

```
<package name="05c91440-09ac-1085-97e9-0002a5d5fd2e">
   <author>Author</author>
   <description>
      This package contains the assertion
"BioSPI_DbGetNextBIR_InvalidModuleHandle"
   </description>
   <assertion name="BioSPI_DbGetNextBIR_InvalidModuleHandle" model="BSPTesting">
      <description>
         This assertion invokes BioSPI_DbGetNextBIR with invalid module handle
and verify if the return code is BioAPIERR_H_FRAMEWORK_INVALID_MODULE_HANDLE.
         The relevant text in BioAPI 1.1 is quoted below from subclauses 2.5.4.9.
         ____________________________________________________________
         BioAPI_RETURN BioAPI BioAPI_DbGetNextBIR
         (BioAPI_HANDLE ModuleHandle,
         BioAPI_DB_CURSOR_PTR Cursor,
         BioAPI_BIR_HANDLE_PTR RetievedBIR,
         BioAPI_UUID_PTR Uuid);
         The BIR identified by the Cursor parameter is retrieved.
         ____________________________________________________________
         and text from subclause 2.3.2.3
         ____________________________________________________________
         #define BioAPIERR_H_FRAMEWORK_INVALID_MODULE_HANDLE \
                      (BioAPI_H_FRAMEWORK_BASE_ERROR +
BioAPI_ERRORCODE_COMMON_EXTENT + 1)
         The given service provider handle is not valid
         ____________________________________________________________

         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_Enroll to generate a BIR.
            5) Invoke function BioSPI_DbStoreBIR to store the enrolled BIR into
database.
            6) Invoke function BioSPI_DbSetCursor to set the cursor to the first
record.
            7) Invoke function BioSPI_DbGetNextBIR with invalid module handle.
```

```
          8) Verify the return code, it is expected to be
BioAPIERR_H_FRAMEWORK_INVALID_MODULE_HANDLE.
          9) Invoke function BioSPI_DbDelete to delete the database.
          10)  Detach and unload the BSP.

        If any of the intermediate operations fail, an UNDECIDED conformity
response is issued.
     </description>
     <!-- UUID of the BSP under test -->
     <input name="_moduleUuid"/>
     <!-- Major version number of the BSP under test -->
     <input name="_moduleVersionMajor"/>
     <!-- Minor version number of the BSP under test -->
     <input name="_moduleVersionMinor"/>
     <!-- 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 capture -->
     <input name="_capturetimeout"/>
     <!-- Database Name to be opened -->
     <input name="_dbName"/>
     <!-- Indicates if the BSP support DB creation-->
     <input name="_supportDBCreation"/>
     <!-- Invocation of the primary activity of this assertion with input
parameter values assigned from the assertion's parameters. -->
     <invoke activity="BioSPI_DbGetNextBIR">
        <input name="moduleUuid" var="_moduleUuid"/>
        <input name="moduleVersionMajor" var="_moduleVersionMajor"/>
        <input name="moduleVersionMinor" var="_moduleVersionMinor"/>
        <input name="inserttimeouttime" var="_inserttimeout"/>
        <input name="nosourcepresentsupported" var="_noSourcePresentSupported"/>
        <input name="sourcepresenttimeouttime" var="_sourcepresenttimeout"/>
        <input name="capturetimeout" var="_capturetimeout"/>
        <input name="dbName" var="_dbName"/>
        <input name="supportDBCreation" var="_supportDBCreation"/>
     </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="be0a1330-d59e-11d8-9669-
0800200c9a66" function="BioSPI_ModuleEventHandler"/>
   </assertion>
   <activity name="BioSPI_DbGetNextBIR">
     <input name="moduleUuid"/>
     <input name="moduleVersionMajor"/>
     <input name="moduleVersionMinor"/>
     <input name="inserttimeouttime"/>
     <input name="nosourcepresentsupported"/>
     <input name="sourcepresenttimeouttime"/>
     <input name="capturetimeout"/>
     <input name="dbName"/>
     <input name="supportDBCreation"/>
     <!-- This assertion will use ModuleHandle "1" for all BioSPI calls that
require it -->
     <set name="_modulehandle" value="1"/>
     <!-- Invoke the functions BioSPI_ModuleLoad and BioSPI_ModuleAttach
exposed by the BSP under test.
          The input value for the parameter "deviceIDOrNull" is "0", therefore
the assertion will test the BSP's default device. -->
```

```
     <invoke activity="LoadAndAttach" package="be0a1330-d59e-11d8-9669-
0800200c9a66" break_on_break="true">
        <input name="moduleUuid" var="moduleUuid"/>
        <input name="moduleVersionMajor" var="moduleVersionMajor"/>
        <input name="moduleVersionMinor" var="moduleVersionMinor"/>
        <input name="deviceIDOrNull" value="0"/>
        <input name="module" var="_modulehandle"/>
        <input name="eventtimeouttime" var="inserttimeouttime"/>
     </invoke>
     <invoke activity="PrepareDBTesting" package="be0a1330-d59e-11d8-9669-
0800200c9a66" break_on_break="true">
        <only_if>
           <same_as var1="supportDBCreation" value2="true"/>
        </only_if>
        <input name="ModuleHandle" var="_modulehandle"/>
        <input name="dbName" var="dbName"/>
        <input name="nosourcepresentsupported" var="nosourcepresentsupported"/>
     </invoke>
     <!-- Invoke the function BioSPI_DbOpen to open the specified database. -->
     <invoke function="BioSPI_DbOpen">
        <input name="ModuleHandle" var="_modulehandle"/>
        <input name="DbName" var="dbName"/>
        <input name="ReadAccessRequest" value="true"/>
        <input name="WriteAccessRequest" value="true"/>
        <output name="DbHandle" setvar="dbHandle"/>
        <output name="Cursor" setvar="cursor"/>
        <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_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>
     <!-- The BSP is ready to capture. Invoke the function BioSPI_Enroll for
the purpose of enrollment. -->
     <invoke function="BioSPI_Enroll">
        <input name="ModuleHandle" var="_modulehandle"/>
        <input name="Purpose"
var="__BioAPI_PURPOSE_ENROLL_FOR_VERIFICATION_ONLY"/>
        <input name="Timeout" var="capturetimeout"/>
        <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="ModuleHandle" var="_modulehandle"/>
        <input name="BIRToStore_Form" var="__BioAPI_BIR_HANDLE_INPUT"/>
        <input name="BIRToStore_BIRHandle" var="newtemplate_handle"/>
        <input name="DbHandle" var="dbHandle"/>
        <input name="no_Uuid" value="false"/>
        <output name="Uuid" setvar="uuid"/>
        <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_DbStoreBIR has returned BioAPI_OK.
        </description>
        <equal_to var1="return" var2="__BioAPI_OK"/>
     </assert_condition>
     <!-- Invoke function BioSPI_DbSetCursor with valid input parameters -->
     <invoke function="BioSPI_DbSetCursor">
        <input name="ModuleHandle" var="_modulehandle"/>
        <input name="DbHandle" var="dbHandle"/>
        <input name="KeyValue" var="uuid"/>
        <output name="Cursor" setvar="cursor"/>
        <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_DbSetCursor has returned BioAPI_OK
        </description>
        <equal_to var1="return" var2="__BioAPI_OK"/>
     </assert_condition>
     <!-- Invoke the function BioSPI_DbGetNextBIR with valid input parameters -
->
     <invoke function="BioSPI_DbGetNextBIR">
        <input name="ModuleHandle" value="0"/>
        <input name="Cursor" var="cursor"/>
        <output name="RetrievedBIR" setvar="retrievedBir_handle"/>
        <output name="Uuid" setvar="uuid"/>
        <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_DbGetNextBIR has returned
BioAPIERR_H_FRAMEWORK_INVALID_MODULE_HANDLE.
        </description>
        <equal_to var1="return"
var2="__BioAPIERR_H_FRAMEWORK_INVALID_MODULE_HANDLE "/>
     </assert_condition>
     <!-- Invoke the function BioSPI_DbClose with valid input parameters -->
     <invoke function="BioSPI_DbClose">
        <input name="ModuleHandle" var="_modulehandle"/>
        <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 activity="CleanUpDBTesting" package="be0a1330-d59e-11d8-9669-
0800200c9a66" break_on_break="true">
        <only_if>
           <same_as var1="supportDBCreation" value2="true"/>
        </only_if>
        <input name="ModuleHandle" var="_modulehandle"/>
        <input name="dbName" var="dbName"/>
     </invoke>
     <!-- Invoke the functions BioSPI_ModuleDetach and BioSPI_ModuleUnload -->
     <invoke activity="DetachAndUnload" package="be0a1330-d59e-11d8-9669-
0800200c9a66">
        <input name="moduleUuid" var="moduleUuid"/>
        <input name="module" var="_modulehandle"/>
     </invoke>
   </activity>
</package>
```

16.52.4 Assertion 31.3 BioSPI_DbGetNextBIR_InvalidCursor

16.52.4.1 Test Purpose:
To test BioSPI_DbGetNextBIR with an nvalid cursor if supported by the BSP.

16.52.4.2 Test Scenario:
If supported by the BSP, call BioSPI_DbGetNextBIR with an invalid cursor.

16.52.4.3 Expected Results:
Return code other than BioAPI_OK.

16.52.4.4 XML:

16.52.5 Assertion 31.4 BioSPI_DbGetNextBIR_EndOfDatabase

16.52.5.1 Test Purpose:
To test BioSPI_DbGetNextBIR returns the correct end of database indicator.

16.52.5.2 Test Scenario:
Repeatedly call BioSPI_DbGetNextBIR until there are no more records.

16.52.5.3 Expected Results:
Return code BioAPIERR_BSP_END_OF_DATABASE.

16.52.5.4 XML:

16.53 Feature 32. BioSPI Db Query BIR

16.53.1 BioAPI Specification References:

3.3.5.10

16.53.2 Assertion 32.1 BioSPI_DbQueryBIR_ValidParam

16.53.2.1 Test Purpose:
To test BioSPI_DbQueryBIR with valid parameters if supported by the BSP.

16.53.2.2 Test Scenario:
If supported by the BSP, call BioSPI_DbQueryBIR with valid parameters.

16.53.2.3 Expected Results:
Return code BioAPI_OK.

16.53.2.4 XML:

```
<package name="03a18120-09b5-1085-9cbf-0002a5d5fd2e">
   <author>Author</author>
   <description>
      This package contains the assertion "BioSPI_DbQueryBIR_ValidParam"
   </description>
   <assertion name="BioSPI_DbQueryBIR_ValidParam" model="BSPTesting">
      <description>
         This assertion invokes BioSPI_DbQueryBIR with valid input parameters and
verify if the return code is BioAPI_OK.
         The relevant text in BioAPI 1.1 is quoted below from subclauses
2.5.4.10.
         ____________________________________________________________
         BioAPI_RETURN BioAPI BioAPI_DbQueryBIR
         (BioAPI_HANDLE ModuleHandle,
         BioAPI_DB_HANDLE DbHandle,
         const BioAPI_INPUT_BIR *BIRToQuery,
         BioAPI_UUID_PTR Uuid);

         If the BIR identified by the BIRToQuery parameter is in the open
database identified by the DbHandle parameter, a pointer to its UUID is
returned. Otherwise, BioAPIERR_BSP_RECORD_NOT_FOUND is returned.
         ____________________________________________________________
         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_Enroll to generate a BIR.
            5) Invoke function BioSPI_DbStoreBIR to store the enrolled BIR into
database.
            6) Invoke function BioSPI_DbQueryBIR to query the stored BIR.
            7) Verify the return code, it is expected to be BioAPI_OK, and the
UUID.
            8) Invoke function BioSPI_DbDelete to delete the database.
            9) Detach and unload the BSP.

         If any of the intermediate operations fail, an UNDECIDED conformity
response is issued.
      </description>
      <!-- UUID of the BSP under test -->
      <input name="_moduleUuid"/>
      <!-- Major version number of the BSP under test -->
      <input name="_moduleVersionMajor"/>
      <!-- Minor version number of the BSP under test -->
```

```
      <input name="_moduleVersionMinor"/>
      <!-- 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 capture -->
      <input name="_capturetimeout"/>
      <!-- Database Name to be opened -->
      <input name="_dbName"/>
      <!-- Indicates if the BSP support DB creation-->
      <input name="_supportDBCreation"/>
      <!-- Invocation of the primary activity of this assertion with input
parameter values assigned from the assertion's parameters. -->
      <invoke activity="BioSPI_DbQueryBIR">
         <input name="moduleUuid" var="_moduleUuid"/>
         <input name="moduleVersionMajor" var="_moduleVersionMajor"/>
         <input name="moduleVersionMinor" var="_moduleVersionMinor"/>
         <input name="inserttimeouttime" var="_inserttimeout"/>
         <input name="nosourcepresentsupported" var="_noSourcePresentSupported"/>
         <input name="sourcepresenttimeouttime" var="_sourcepresenttimeout"/>
         <input name="capturetimeout" var="_capturetimeout"/>
         <input name="dbName" var="_dbName"/>
         <input name="supportDBCreation" var="_supportDBCreation"/>
      </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="be0a1330-d59e-11d8-9669-
0800200c9a66" function="BioSPI_ModuleEventHandler"/>
   </assertion>
   <activity name="BioSPI_DbQueryBIR">
      <input name="moduleUuid"/>
      <input name="moduleVersionMajor"/>
      <input name="moduleVersionMinor"/>
      <input name="inserttimeouttime"/>
      <input name="nosourcepresentsupported"/>
      <input name="sourcepresenttimeouttime"/>
      <input name="capturetimeout"/>
      <input name="dbName"/>
      <input name="supportDBCreation"/>
      <!-- This assertion will use ModuleHandle "1" for all BioSPI calls that
require it -->
      <set name="_modulehandle" value="1"/>
      <!-- Invoke the functions BioSPI_ModuleLoad and BioSPI_ModuleAttach
exposed by the BSP under test.
            The input value for the parameter "deviceIDOrNull" is "0", therefore
the assertion will test the BSP's default device. -->
      <invoke activity="LoadAndAttach" package="be0a1330-d59e-11d8-9669-
0800200c9a66" break_on_break="true">
         <input name="moduleUuid" var="moduleUuid"/>
         <input name="moduleVersionMajor" var="moduleVersionMajor"/>
         <input name="moduleVersionMinor" var="moduleVersionMinor"/>
         <input name="deviceIDOrNull" value="0"/>
         <input name="module" var="_modulehandle"/>
         <input name="eventtimeouttime" var="inserttimeouttime"/>
      </invoke>
      <invoke activity="PrepareDBTesting" package="be0a1330-d59e-11d8-9669-
0800200c9a66" break_on_break="true">
         <only_if>
            <same_as var1="supportDBCreation" value2="true"/>
```

```
        </only_if>
        <input name="ModuleHandle" var="_modulehandle"/>
        <input name="dbName" var="dbName"/>
        <input name="nosourcepresentsupported" var="nosourcepresentsupported"/>
      </invoke>
      <!-- Invoke the function BioSPI_DbOpen to open the specified database. -->
      <invoke function="BioSPI_DbOpen">
        <input name="ModuleHandle" var="_modulehandle"/>
        <input name="DbName" var="dbName"/>
        <input name="ReadAccessRequest" value="true"/>
        <input name="WriteAccessRequest" value="true"/>
        <output name="DbHandle" setvar="dbHandle"/>
        <output name="Cursor" setvar="cursor"/>
        <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_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>
      <!-- The BSP is ready to capture. Invoke the function BioSPI_Enroll for
the purpose of enrollment. -->
      <invoke function="BioSPI_Enroll">
        <input name="ModuleHandle" var="_modulehandle"/>
        <input name="Purpose"
var="__BioAPI_PURPOSE_ENROLL_FOR_VERIFICATION_ONLY"/>
        <input name="Timeout" var="capturetimeout"/>
        <output name="NewTemplate" setvar="newtemplate_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 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="ModuleHandle" var="_modulehandle"/>
        <input name="BIRToStore_Form" var="__BioAPI_BIR_HANDLE_INPUT"/>
        <input name="BIRToStore_BIRHandle" var="newtemplate_handle1"/>
        <input name="DbHandle" var="dbHandle"/>
        <input name="no_Uuid" value="false"/>
        <output name="Uuid" setvar="uuid1"/>
        <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_DbStoreBIR has returned BioAPI_OK.
        </description>
        <equal_to var1="return" var2="__BioAPI_OK"/>
     </assert_condition>
     <!-- Enroll another BIR. -->
     <invoke function="BioSPI_Enroll">
        <input name="ModuleHandle" var="_modulehandle"/>
        <input name="Purpose"
var="__BioAPI_PURPOSE_ENROLL_FOR_VERIFICATION_ONLY"/>
        <input name="Timeout" var="capturetimeout"/>
        <output name="NewTemplate" setvar="newtemplate_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 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="ModuleHandle" var="_modulehandle"/>
        <input name="BIRToStore_Form" var="__BioAPI_BIR_HANDLE_INPUT"/>
        <input name="BIRToStore_BIRHandle" var="newtemplate_handle2"/>
        <input name="DbHandle" var="dbHandle"/>
        <input name="no_Uuid" value="false"/>
        <output name="Uuid" setvar="uuid2"/>
        <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_DbStoreBIR has returned BioAPI_OK.
        </description>
        <equal_to var1="return" var2="__BioAPI_OK"/>
     </assert_condition>
     <!-- Invoke the function BioSPI_DbQueryBIR with valid input parameters -->
     <invoke function="BioSPI_DbQueryBIR">
        <input name="ModuleHandle" var="_modulehandle"/>
        <input name="DbHandle" var="dbHandle"/>
        <input name="BIRToQuery_Form" value="1"/>
        <input name="BIRToQuery_DbHandle" var="dbHandle"/>
        <input name="BIRToQuery_KeyValue" var="uuid1"/>
        <output name="Uuid" setvar="uuid3"/>
        <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_DbQueryBIR has returned BioAPI_OK and the output
UUID points to the expected BIR.
```

```
        </description>
        <equal_to var1="return" var2="__BioAPI_OK"/>
        <same_as var1="uuid3" var2="uuid1"/>
      </assert_condition>
      <!-- Invoke the function BioSPI_DbClose with valid input parameters -->
      <invoke function="BioSPI_DbClose">
        <input name="ModuleHandle" var="_modulehandle"/>
        <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 activity="CleanUpDBTesting" package="be0a1330-d59e-11d8-9669-
0800200c9a66" break_on_break="true">
        <only_if>
           <same_as var1="supportDBCreation" value2="true"/>
        </only_if>
        <input name="ModuleHandle" var="_modulehandle"/>
        <input name="dbName" var="dbName"/>
      </invoke>
      <!-- Invoke the functions BioSPI_ModuleDetach and BioSPI_ModuleUnload -->
      <invoke activity="DetachAndUnload" package="be0a1330-d59e-11d8-9669-
0800200c9a66">
        <input name="moduleUuid" var="moduleUuid"/>
        <input name="module" var="_modulehandle"/>
      </invoke>
   </activity>
</package>
```

16.53.3 Assertion 32.2 BioSPI_DbQueryBIR_InvalidModuleHandle.

16.53.3.1 Test Purpose:
To test BioSPI_DbQueryBIR with ian nvalid module handle if supported by the BSP.

16.53.3.2 Test Scenario:
If supported by the BSP, call BioSPI_DbQueryBIR with an invalid module handle.

16.53.3.3 Expected Results:
Return code other than BioAPI_OK.

16.53.3.4 XML:

```
<package name="014bdd08-09bc-1085-91a2-0002a5d5fd2e">
   <author>Author</author>
   <description>
      This package contains the assertion
"BioSPI_DbQueryBIR_InvalidModuleHandle"
   </description>
   <assertion name="BioSPI_DbQueryBIR_InvalidModuleHandle" model="BSPTesting">
      <description>
         This assertion invokes BioSPI_DbQueryBIR with invalid module handle and
verify if the return code is BioAPIERR_H_FRAMEWORK_INVALID_MODULE_HANDLE.
         The relevant text in BioAPI 1.1 is quoted below from subclauses
2.5.4.10.
```

```
        ____________________________________________________________
        BioAPI_RETURN BioAPI BioAPI_DbQueryBIR
        (BioAPI_HANDLE ModuleHandle,
        BioAPI_DB_HANDLE DbHandle,
        const BioAPI_INPUT_BIR *BIRToQuery,
        BioAPI_UUID_PTR Uuid);

        If the BIR identified by the BIRToQuery parameter is in the open
database identified by the DbHandle parameter, a pointer to its UUID is
returned. Otherwise, BioAPIERR_BSP_RECORD_NOT_FOUND is returned.
        ____________________________________________________________
        and text from subclause 2.3.2.3
        ____________________________________________________________
        #define BioAPIERR_H_FRAMEWORK_INVALID_MODULE_HANDLE \
                     (BioAPI_H_FRAMEWORK_BASE_ERROR +
BioAPI_ERRORCODE_COMMON_EXTENT + 1)
        The given service provider handle is not valid
        ____________________________________________________________

        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_Enroll to generate a BIR.
           5) Invoke function BioSPI_DbStoreBIR to store the enrolled BIR into
database.
           6) Invoke function BioSPI_DbQueryBIR with invalid module handle.
           7) Verify the return code, it is expected to be
BioAPIERR_H_FRAMEWORK_INVALID_MODULE_HANDLE.
           8) Invoke function BioSPI_DbDelete to delete the database.
           9) Detach and unload the BSP.

        If any of the intermediate operations fail, an UNDECIDED conformity
response is issued.
     </description>
     <!-- UUID of the BSP under test -->
     <input name="_moduleUuid"/>
     <!-- Major version number of the BSP under test -->
     <input name="_moduleVersionMajor"/>
     <!-- Minor version number of the BSP under test -->
     <input name="_moduleVersionMinor"/>
     <!-- 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 capture -->
     <input name="_capturetimeout"/>
     <!-- Database Name to be opened -->
     <input name="_dbName"/>
     <!-- Indicates if the BSP support DB creation-->
     <input name="_supportDBCreation"/>
     <!-- Invocation of the primary activity of this assertion with input
parameter values assigned from the assertion's parameters. -->
     <invoke activity="BioSPI_DbQueryBIR">
        <input name="moduleUuid" var="_moduleUuid"/>
        <input name="moduleVersionMajor" var="_moduleVersionMajor"/>
        <input name="moduleVersionMinor" var="_moduleVersionMinor"/>
        <input name="inserttimeouttime" var="_inserttimeout"/>
```

```
        <input name="nosourcepresentsupported" var="_noSourcePresentSupported"/>
        <input name="sourcepresenttimeouttime" var="_sourcepresenttimeout"/>
        <input name="capturetimeout" var="_capturetimeout"/>
        <input name="dbName" var="_dbName"/>
        <input name="supportDBCreation" var="_supportDBCreation"/>
     </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="be0a1330-d59e-11d8-9669-
0800200c9a66" function="BioSPI_ModuleEventHandler"/>
   </assertion>
   <activity name="BioSPI_DbQueryBIR">
     <input name="moduleUuid"/>
     <input name="moduleVersionMajor"/>
     <input name="moduleVersionMinor"/>
     <input name="inserttimeouttime"/>
     <input name="nosourcepresentsupported"/>
     <input name="sourcepresenttimeouttime"/>
     <input name="capturetimeout"/>
     <input name="dbName"/>
     <input name="supportDBCreation"/>
     <!-- This assertion will use ModuleHandle "1" for all BioSPI calls that
require it -->
     <set name="_modulehandle" value="1"/>
     <!-- Invoke the functions BioSPI_ModuleLoad and BioSPI_ModuleAttach
exposed by the BSP under test.
          The input value for the parameter "deviceIDOrNull" is "0", therefore
the assertion will test the BSP's default device. -->
     <invoke activity="LoadAndAttach" package="be0a1330-d59e-11d8-9669-
0800200c9a66" break_on_break="true">
        <input name="moduleUuid" var="moduleUuid"/>
        <input name="moduleVersionMajor" var="moduleVersionMajor"/>
        <input name="moduleVersionMinor" var="moduleVersionMinor"/>
        <input name="deviceIDOrNull" value="0"/>
        <input name="module" var="_modulehandle"/>
        <input name="eventtimeouttime" var="inserttimeouttime"/>
     </invoke>
     <invoke activity="PrepareDBTesting" package="be0a1330-d59e-11d8-9669-
0800200c9a66" break_on_break="true">
        <only_if>
           <same_as var1="supportDBCreation" value2="true"/>
        </only_if>
        <input name="ModuleHandle" var="_modulehandle"/>
        <input name="dbName" var="dbName"/>
        <input name="nosourcepresentsupported" var="nosourcepresentsupported"/>
     </invoke>
     <!-- Invoke the function BioSPI_DbOpen to open the specified database. -->
     <invoke function="BioSPI_DbOpen">
        <input name="ModuleHandle" var="_modulehandle"/>
        <input name="DbName" var="dbName"/>
        <input name="ReadAccessRequest" value="true"/>
        <input name="WriteAccessRequest" value="true"/>
        <output name="DbHandle" setvar="dbHandle"/>
        <output name="Cursor" setvar="cursor"/>
        <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_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>
     <!-- The BSP is ready to capture. Invoke the function BioSPI_Enroll for
the purpose of enrollment. -->
     <invoke function="BioSPI_Enroll">
        <input name="ModuleHandle" var="_modulehandle"/>
        <input name="Purpose"
var="__BioAPI_PURPOSE_ENROLL_FOR_VERIFICATION_ONLY"/>
        <input name="Timeout" var="capturetimeout"/>
        <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="ModuleHandle" var="_modulehandle"/>
        <input name="BIRToStore_Form" var="__BioAPI_BIR_HANDLE_INPUT"/>
        <input name="BIRToStore_BIRHandle" var="newtemplate_handle"/>
        <input name="DbHandle" var="dbHandle"/>
        <input name="no_Uuid" value="false"/>
        <output name="Uuid" setvar="uuid1"/>
        <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_DbStoreBIR has returned BioAPI_OK.
        </description>
        <equal_to var1="return" var2="__BioAPI_OK"/>
     </assert_condition>
     <!-- Enroll another BIR. -->
     <invoke function="BioSPI_Enroll">
        <input name="ModuleHandle" var="_modulehandle"/>
        <input name="Purpose"
var="__BioAPI_PURPOSE_ENROLL_FOR_VERIFICATION_ONLY"/>
        <input name="Timeout" var="capturetimeout"/>
        <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="ModuleHandle" var="_modulehandle"/>
        <input name="BIRToStore_Form" var="__BioAPI_BIR_HANDLE_INPUT"/>
        <input name="BIRToStore_BIRHandle" var="newtemplate_handle"/>
        <input name="DbHandle" var="dbHandle"/>
        <input name="no_Uuid" value="false"/>
        <output name="Uuid" setvar="uuid2"/>
        <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_DbStoreBIR has returned BioAPI_OK.
        </description>
        <equal_to var1="return" var2="__BioAPI_OK"/>
      </assert_condition>
      <!-- Invoke the function BioSPI_DbQueryBIR with valid input parameters -->
      <invoke function="BioSPI_DbQueryBIR">
        <input name="ModuleHandle" value="0"/>
        <input name="DbHandle" var="dbHandle"/>
        <input name="BIRToQuery_Form" value="1"/>
        <input name="BIRToQuery_DbHandle" var="dbHandle"/>
        <input name="BIRToQuery_KeyValue" var="uuid1"/>
        <output name="Uuid" setvar="uuid3"/>
        <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_DbQueryBIR has returned
BioAPIERR_H_FRAMEWORK_INVALID_MODULE_HANDLE.
        </description>
        <equal_to var1="return"
var2="__BioAPIERR_H_FRAMEWORK_INVALID_MODULE_HANDLE "/>
      </assert_condition>
      <!-- Invoke the function BioSPI_DbClose with valid input parameters -->
      <invoke function="BioSPI_DbClose">
        <input name="ModuleHandle" var="_modulehandle"/>
        <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 activity="CleanUpDBTesting" package="be0a1330-d59e-11d8-9669-
```

```
0800200c9a66" break_on_break="true">
        <only_if>
           <same_as var1="supportDBCreation" value2="true"/>
        </only_if>
        <input name="ModuleHandle" var="_modulehandle"/>
        <input name="dbName" var="dbName"/>
     </invoke>
     <!-- Invoke the functions BioSPI_ModuleDetach and BioSPI_ModuleUnload -->
     <invoke activity="DetachAndUnload" package="be0a1330-d59e-11d8-9669-
0800200c9a66">
        <input name="moduleUuid" var="moduleUuid"/>
        <input name="module" var="_modulehandle"/>
     </invoke>
   </activity>
</package>
```

16.53.4 Assertion 32.3 BioSPI_DbQueryBIR_InvalidDBHandle

16.53.4.1 Test Purpose:
To test BioSPI_DbQueryBIR with ian nvalid DB handle if supported by the BSP.

16.53.4.2 Test Scenario:
If supported by the BSP, call BioSPI_DbQueryBIR with an invalid DB handle.

16.53.4.3 Expected Results:
Return code other than BioAPI_OK.

16.53.4.4 XML:

16.54 Feature 33. BioSPI Db Delete BIR

16.54.1 BioAPI Specification References:
3.3.5.11

16.54.2 Assertion 33.1 BioSPI_DbDeleteBIR_ValidParam

16.54.2.1 Test Purpose:
To test BioSPI_DbDeleteBIR with valid parameters if supported by the BSP.

16.54.2.2 Test Scenario:
If supported by the BSP, call BioSPI_DbDeleteBIR with valid parameters.

16.54.2.3 Expected Results:
Return code BioAPI_OK.

16.54.2.4 XML:

```
<package name="00890df0-09c5-1085-9d39-0002a5d5fd2e">
   <author>Author</author>
   <description>
      This package contains the assertion "BioSPI_DbDeleteBIR_ValidParam"
   </description>
   <assertion name="BioSPI_DbDeleteBIR_ValidParam" model="BSPTesting">
      <description>
         This assertion invokes BioSPI_DbDeleteBIR with valid parameters and
verify if the return code is BioAPI_OK.
```

```
        The relevant text in BioAPI 1.1 is quoted below from subclauses
2.5.4.11.
        ____________________________________________________________
        BioAPI_RETURN BioAPI BioAPI_DbDeleteBIR
        (BioAPI_HANDLE ModuleHandle,
        BioAPI_DB_HANDLE DbHandle,
        const BioAPI_UUID *KeyValue);

        The BIR identified by the KeyValue parameter in the open database
identified by the DbHandle parameter is
                     deleted from the 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) Invoke function BioSPI_DbCreate to create the specified database.
           4) Invoke function BioSPI_Enroll to generate a BIR.
           5) Invoke function BioSPI_DbStoreBIR to store the captured BIR into
database.
           6) Invoke function BioSPI_DbDeleteBIR to delete the stored BIR.
           7) Invoke function BioSPI_DbGetBIR to retrieve the deleted BIR, the
expected return code is BioAPIERR_BSP_RECORD_NOT_FOUND.
           8) Invoke function BioSPI_DbDelete to delete the database.
           9) Detach and unload the BSP.

        If any of the intermediate operations fail, an UNDECIDED conformity
response is issued.
      </description>
      <!-- UUID of the BSP under test -->
      <input name="_moduleUuid"/>
      <!-- Major version number of the BSP under test -->
      <input name="_moduleVersionMajor"/>
      <!-- Minor version number of the BSP under test -->
      <input name="_moduleVersionMinor"/>
      <!-- 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 capture -->
      <input name="_capturetimeout"/>
      <!-- Database Name to be opened -->
      <input name="_dbName"/>
      <!-- Indicates if the BSP support database creation -->
      <input name="_supportDBCreation"/>
      <!-- Invocation of the primary activity of this assertion with input
parameter values assigned from the assertion's parameters. -->
      <invoke activity="BioSPI_DbDeleteBIR">
        <input name="moduleUuid" var="_moduleUuid"/>
        <input name="moduleVersionMajor" var="_moduleVersionMajor"/>
        <input name="moduleVersionMinor" var="_moduleVersionMinor"/>
        <input name="inserttimeouttime" var="_inserttimeout"/>
        <input name="nosourcepresentsupported" var="_noSourcePresentSupported"/>
        <input name="sourcepresenttimeouttime" var="_sourcepresenttimeout"/>
        <input name="capturetimeout" var="_capturetimeout"/>
        <input name="dbName" var="_dbName"/>
        <input name="supportDBCreation" var="_supportDBCreation"/>
      </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="be0a1330-d59e-11d8-9669-
0800200c9a66" function="BioSPI_ModuleEventHandler"/>
   </assertion>
   <activity name="BioSPI_DbDeleteBIR">
      <input name="moduleUuid"/>
      <input name="moduleVersionMajor"/>
      <input name="moduleVersionMinor"/>
      <input name="inserttimeouttime"/>
      <input name="nosourcepresentsupported"/>
      <input name="sourcepresenttimeouttime"/>
      <input name="capturetimeout"/>
      <input name="dbName"/>
      <input name="supportDBCreation"/>
      <!-- This assertion will use ModuleHandle "1" for all BioSPI calls that
require it -->
      <set name="_modulehandle" value="1"/>
      <!-- Invoke the functions BioSPI_ModuleLoad and BioSPI_ModuleAttach
exposed by the BSP under test.
           The input value for the parameter "deviceIDOrNull" is "0", therefore
the assertion will test the BSP's default device. -->
      <invoke activity="LoadAndAttach" package="be0a1330-d59e-11d8-9669-
0800200c9a66" break_on_break="true">
         <input name="moduleUuid" var="moduleUuid"/>
         <input name="moduleVersionMajor" var="moduleVersionMajor"/>
         <input name="moduleVersionMinor" var="moduleVersionMinor"/>
         <input name="deviceIDOrNull" value="0"/>
         <input name="module" var="_modulehandle"/>
         <input name="eventtimeouttime" var="inserttimeouttime"/>
      </invoke>
      <invoke activity="PrepareDBTesting" package="be0a1330-d59e-11d8-9669-
0800200c9a66" break_on_break="true">
         <only_if>
            <same_as var1="supportDBCreation" value2="true"/>
         </only_if>
         <input name="ModuleHandle" var="_modulehandle"/>
         <input name="dbName" var="dbName"/>
         <input name="nosourcepresentsupported" var="nosourcepresentsupported"/>
         <output name="biruuid" setvar="biruuid"/>
      </invoke>
      <!-- Invoke the function BioSPI_DbOpen to open the specified database. -->
      <invoke function="BioSPI_DbOpen">
         <input name="ModuleHandle" var="_modulehandle"/>
         <input name="DbName" var="dbName"/>
         <input name="ReadAccessRequest" value="true"/>
         <input name="WriteAccessRequest" value="true"/>
         <output name="DbHandle" setvar="dbHandle"/>
         <output name="Cursor" setvar="cursor"/>
         <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_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_DbDeleteBIR with valid input parameters --
>
     <invoke function="BioSPI_DbDeleteBIR">
        <input name="ModuleHandle" var="_modulehandle"/>
        <input name="DbHandle" var="dbHandle"/>
        <input name="KeyValue" var="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>
        <description>
          The function BioSPI_DbDeleteBIR has returned BioAPI_OK.
        </description>
        <equal_to var1="return" var2="__BioAPI_OK"/>
     </assert_condition>
     <!-- Invoke the function BioSPI_DbGetBIR with valid input parameters -->
     <invoke function="BioSPI_DbGetBIR">
        <input name="ModuleHandle" var="_modulehandle"/>
        <input name="DbHandle" var="dbHandle"/>
        <input name="KeyValue" var="biruuid"/>
        <output name="RetrievedBIR" setvar="retrievedBir_handle"/>
        <output name="Cursor" setvar="cursor"/>
        <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_DbGetBIR has returned
BioAPIERR_BSP_RECORD_NOT_FOUND.
        </description>
        <equal_to var1="return" var2="__BioAPIERR_BSP_RECORD_NOT_FOUND "/>
     </assert_condition>
     <!-- Invoke the function BioSPI_DbClose with valid input parameters -->
     <invoke function="BioSPI_DbClose">
        <input name="ModuleHandle" var="_modulehandle"/>
        <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="ModuleHandle" var="_modulehandle"/>
        <input name="DbName" var="dbName"/>
        <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>
        <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="be0a1330-d59e-11d8-9669-
0800200c9a66">
        <input name="moduleUuid" var="moduleUuid"/>
        <input name="module" var="_modulehandle"/>
     </invoke>
   </activity>
</package>
```

16.54.3 Assertion 33.2 BioSPI_DbDeleteBIR_InvalidModuleHandle

16.54.3.1 Test Purpose:
(?OMJ?) If supported by the BSP, call BioSPI_DbQueryBIR withan invalid module handle.

16.54.3.2 Test Scenario:
If supported by the BSP, call BioSPI_DbDeleteBIR with an invalid module handle.

16.54.3.3 Expected Results:
Return code other than BioAPI_OK.

16.54.3.4 XML:

```
<package name="03aab498-09c6-1085-b751-0002a5d5fd2e">
   <author>Author</author>
   <description>
      This package contains the assertion
"BioSPI_DbDeleteBIR_InvalidModuleHandle"
   </description>
   <assertion name="BioSPI_DbDeleteBIR_InvalidModuleHandle" model="BSPTesting">
      <description>
         This assertion invokes BioSPI_DbDeleteBIR with an invalid module handle
and verify if the return code is BioAPIERR_H_FRAMEWORK_INVALID_MODULE_HANDLE.
         The relevant text in BioAPI 1.1 is quoted below from subclauses
2.5.4.11.
         ____________________________________________________________
         BioAPI_RETURN BioAPI BioAPI_DbDeleteBIR
         (BioAPI_HANDLE ModuleHandle,
         BioAPI_DB_HANDLE DbHandle,
         const BioAPI_UUID *KeyValue);

         The BIR identified by the KeyValue parameter in the open database
identified by the DbHandle parameter is
                       deleted from the database.
         ____________________________________________________________
         and text from subclause 2.3.2.3
         ____________________________________________________________
         #define BioAPIERR_H_FRAMEWORK_INVALID_MODULE_HANDLE \
                       (BioAPI_H_FRAMEWORK_BASE_ERROR +
BioAPI_ERRORCODE_COMMON_EXTENT + 1)
         The given service provider handle is not valid
         ____________________________________________________________
         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_Enroll to generate a BIR.
          5) Invoke function BioSPI_DbStoreBIR to store the captured BIR into
database.
          6) Invoke function BioSPI_DbDeleteBIR to delete the stored BIR.
          7) Invoke function BioSPI_DbDelete to delete the database.
          8) Detach and unload the BSP.

        If any of the intermediate operations fail, an UNDECIDED conformity
response is issued.
     </description>
     <!-- UUID of the BSP under test -->
     <input name="_moduleUuid"/>
     <!-- Major version number of the BSP under test -->
     <input name="_moduleVersionMajor"/>
     <!-- Minor version number of the BSP under test -->
     <input name="_moduleVersionMinor"/>
     <!-- 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 capture -->
     <input name="_capturetimeout"/>
     <!-- Database Name to be opened -->
     <input name="_dbName"/>
     <!-- Indicates if the BSP support database creation -->
     <input name="_supportDBCreation"/>
     <!-- Invocation of the primary activity of this assertion with input
parameter values assigned from the assertion's parameters. -->
     <invoke activity="BioSPI_DbDeleteBIR">
        <input name="moduleUuid" var="_moduleUuid"/>
        <input name="moduleVersionMajor" var="_moduleVersionMajor"/>
        <input name="moduleVersionMinor" var="_moduleVersionMinor"/>
        <input name="inserttimeouttime" var="_inserttimeout"/>
        <input name="nosourcepresentsupported" var="_noSourcePresentSupported"/>
        <input name="sourcepresenttimeouttime" var="_sourcepresenttimeout"/>
        <input name="capturetimeout" var="_capturetimeout"/>
        <input name="dbName" var="_dbName"/>
        <input name="supportDBCreation" var="_supportDBCreation"/>
     </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="be0a1330-d59e-11d8-9669-
0800200c9a66" function="BioSPI_ModuleEventHandler"/>
   </assertion>
   <activity name="BioSPI_DbDeleteBIR">
     <input name="moduleUuid"/>
     <input name="moduleVersionMajor"/>
     <input name="moduleVersionMinor"/>
     <input name="inserttimeouttime"/>
     <input name="nosourcepresentsupported"/>
     <input name="sourcepresenttimeouttime"/>
     <input name="capturetimeout"/>
     <input name="dbName"/>
     <input name="supportDBCreation"/>
     <!-- This assertion will use ModuleHandle "1" for all BioSPI calls that
require it -->
     <set name="_modulehandle" value="1"/>
```

```
     <!-- Invoke the functions BioSPI_ModuleLoad and BioSPI_ModuleAttach
exposed by the BSP under test.
          The input value for the parameter "deviceIDOrNull" is "0", therefore
the assertion will test the BSP's default device. -->
     <invoke activity="LoadAndAttach" package="be0a1330-d59e-11d8-9669-
0800200c9a66" break_on_break="true">
        <input name="moduleUuid" var="moduleUuid"/>
        <input name="moduleVersionMajor" var="moduleVersionMajor"/>
        <input name="moduleVersionMinor" var="moduleVersionMinor"/>
        <input name="deviceIDOrNull" value="0"/>
        <input name="module" var="_modulehandle"/>
        <input name="eventtimeouttime" var="inserttimeouttime"/>
     </invoke>
     <invoke activity="PrepareDBTesting" package="be0a1330-d59e-11d8-9669-
0800200c9a66" break_on_break="true">
        <only_if>
          <same_as var1="supportDBCreation" value2="true"/>
        </only_if>
        <input name="ModuleHandle" var="_modulehandle"/>
        <input name="dbName" var="dbName"/>
        <input name="nosourcepresentsupported" var="nosourcepresentsupported"/>
        <output name="biruuid" setvar="biruuid"/>
     </invoke>
     <!-- Invoke the function BioSPI_DbOpen to open the specified database. -->
     <invoke function="BioSPI_DbOpen">
        <input name="ModuleHandle" var="_modulehandle"/>
        <input name="DbName" var="dbName"/>
        <input name="ReadAccessRequest" value="true"/>
        <input name="WriteAccessRequest" value="true"/>
        <output name="DbHandle" setvar="dbHandle"/>
        <output name="Cursor" setvar="cursor"/>
        <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_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_DbDeleteBIR with valid input parameters --
>
     <invoke function="BioSPI_DbDeleteBIR">
        <input name="ModuleHandle" value="0"/>
        <input name="DbHandle" var="dbHandle"/>
        <input name="KeyValue" var="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>
        <description>
          The function BioSPI_DbDeleteBIR has returned
BioAPIERR_H_FRAMEWORK_INVALID_MODULE_HANDLE.
        </description>
        <equal_to var1="return"
```

```
var2="__BioAPIERR_H_FRAMEWORK_INVALID_MODULE_HANDLE "/>
     </assert_condition>
     <!-- Invoke the function BioSPI_DbClose with valid input parameters -->
     <invoke function="BioSPI_DbClose">
        <input name="ModuleHandle" var="_modulehandle"/>
        <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="ModuleHandle" var="_modulehandle"/>
        <input name="DbName" var="dbName"/>
        <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>
        <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="be0a1330-d59e-11d8-9669-
0800200c9a66">
        <input name="moduleUuid" var="moduleUuid"/>
        <input name="module" var="_modulehandle"/>
     </invoke>
   </activity>
</package>
```

16.54.4 Assertion 33.3 BioSPI_DbDeleteBIR_InvalidDBHandle

16.54.4.1 Test Purpose:
(?OMJ?) If supported by the BSP, call BioSPI_DbQueryBIR withan invalid DB handle.

16.54.4.2 Test Scenario:
If supported by the BSP, call BioSPI_DbDeleteBIR with an invalid DB handle.

16.54.4.3 Expected Results:
Return code other than BioAPI_OK.

16.54.4.4 XML:

16.55 Feature 101. BSP must implement all mandatory functions IAW SPI

16.55.1 BioAPI Specification References:
A.2

16.55.2 Assertion 101.1 BSP_MandatoryFunctionsImplemented

16.55.2.1 Test Purpose:
To test that the BSP implements all mandatory functions: BioSPI_ModuleLoad, BioSPI_ModuleUnload, BioSPI_ModuleAttach, BioSPI_ModuleDetach, BioSPI_FreeBIRHandle, BioSPI_GetBIRFromHandle, BioSPI_GetHeaderFromHandle, BioSPI_Enroll and BioSPI_Verify. Identification BSPs must also implement BioSPI_Identify. If the BSP is a Client/Server BSP, these functions can be executed Locally or Remotely: BioSPI_Enroll, BioSPI_Verify, and (if an Identification BSP) BioSPI_Identify.

16.55.2.2 Test Scenario:
N.A.

16.55.2.3 Expected Results:
The BSP implements all mandatory functions.

This assertion cannot be tested by calling BioSPI functions. The test lab will use some other means to determine if a BSP passes this assertion.

16.55.2.4 XML:
N.A.

16.56 Feature 102. BSP accepts all valid input parameters and returns valid outputs

16.56.1 BioAPI Specification References:
A.2

16.56.2 Assertion 102.1 BSP_ValidInOut

16.56.2.1 Test Purpose:
To test that the BSP accepts all valid inputs to all mandatory functions and implemented optional functions, and returns valid output from all mandatory functions and implemented optional functions.

16.56.2.2 Test Scenario:
N.A.

16.56.2.3 Expected Results:
The BSP accepts all valid inputs to all mandatory functions and implemented optional functions, and returns valid output from all mandatory functions and implemented optional functions.

This assertion cannot be tested by calling BioSPI functions. The test lab will use some other means to determine if a BSP passes this assertion.

16.56.2.4 XML:
N.A.

16.57 Feature 103. Options implemented must be in accordance to spec (as shown in column 2 & 3 of Table)

16.57.1 BioAPI Specification References:
A.2

16.57.2 Assertion 103.1 BSP_Options_InSpec

16.57.2.1 Test Purpose:
To test that all options implemented are correct.

16.57.2.2 Test Scenario:
N.A.

16.57.2.3 Expected Results:
Each implemented option should succeed according to its Test Cases.

This assertion cannot be tested by calling BioSPI functions. The test lab will use some other means to determine if a BSP passes this assertion.

16.57.2.4 XML:
N.A.

16.58 Feature 104. BSP provides all registry entries

16.58.1 BioAPI Specification References:
A.2

16.58.2 Comments:
See also item 1, "BioAPI Registry."

16.58.3 Assertion 104.1 BSP_Registry

16.58.3.1 Test Purpose:
To test that the BSP provides all registry entries.

16.58.3.2 Test Scenario:
N.A.

16.58.3.3 Expected Results:
The BSP provides all registry entries.

This assertion cannot be tested by calling BioSPI functions. The test lab will use some other means to determine if a BSP passes this assertion.

16.58.3.4 XML:
N.A.

16.59 Feature 105. BSP has UUID according to data definition

16.59.1 BioAPI Specification References:
A.2

16.59.2 Assertion 105.1 BSP_UUID

16.59.2.1 Test Purpose:
To test the BSP's UUID.

16.59.2.2 Test Scenario:
N.A.

16.59.2.3 Expected Results:
The UUID conforms to the specification.

This assertion cannot be tested by calling BioSPI functions. The test lab will use some other means to determine if a BSP passes this assertion.

16.59.2.4 XML:
N.A.

16.60 Feature 106. Conformant data structures -- (Biometric data according to 2.1 & 3.2 including the BIR)

16.60.1 BioAPI Specification References:
A.2

16.60.2 Comments:
If function is not tested this will not be included

16.60.3 Assertion 106.1 DataConformant

16.60.3.1 Test Purpose:
To test conformance of data structures.

16.60.3.2 Test Scenario:
N.A.

16.60.3.3 Expected Results:
All data structures are conformant.

This assertion cannot be tested by calling BioSPI functions. The test lab will use some other means to determine if a BSP passes this assertion.

16.60.3.4 XML:
N.A.

16.61 Feature 107. Registered valid Format Owner and Format Type

16.61.1 BioAPI Specification References:
A.2

16.61.2 Assertion 107.1 RegisteredFormatOwnerType

16.61.2.1 Test Purpose:
To test that Format Owner and Format Type are registered and valid.

16.61.2.2 Test Scenario:
N.A.

16.61.2.3 Expected Results:
Format Owner and Format Type are registered and valid.

This assertion cannot be tested by calling BioSPI functions. The test lab will use some other means to determine if a BSP passes this assertion.

16.61.2.4 XML:
N.A.

16.62 Feature 108. Error handling according to 2.3

16.62.1 BioAPI Specification References:
A.2

16.62.2 Comments:
If function is not tested this will not be included

16.62.3 Assertion 108.1 ErrorHandlingIAW_2.3

16.62.3.1 Test Purpose:
To test that the BSP implements error handling in accordance with 2.3.

16.62.3.2 Test Scenario:
N.A.

16.62.3.3 Expected Results:
The BSP implements error handling in accordance with 2.3.

This standard includes conformance requirements related to error handling to the extent that the errors are a) generated by the BSP (not the framework), b) explicitly listed as an error return in the function being tested, and c) are feasible to be induced by a test application. Testing of all possible error conditions is beyond the scope of this standard.

This assertion cannot be tested by calling BioSPI functions. The test lab will use some other means to determine if a BSP passes this assertion. If a BSP passes all assertions items 2 through 33 the BSP can be considered to have passed this assertion.

16.62.3.4 XML:
N.A.

16.63 Feature 109. If GUI BSP must provide it

16.63.1 BioAPI Specification References:
A.2

16.63.2 Assertion 109.1 BSP_GUI_Provided

16.63.2.1 Test Purpose:
To test that the BSP provides a GUI for the capture portion of the Verify operation.

16.63.2.2 Test Scenario:
N.A.

16.63.2.3 Expected Results:
The BSP provides a GUI for the capture portion of the Verify operation.

This assertion cannot be tested by calling BioSPI functions. The test lab will use some other means to determine if a BSP passes this assertion.

16.63.2.4 XML:
N.A.

16.63.3 Assertion 109.2 BSP_GUI_Provided_Cancel

16.63.3.1 Test Purpose:
To test that if the BSP provides a GUI, an operator abort/cancel mechanism is also provided.

16.63.3.2 Test Scenario:
N.A.

16.63.3.3 Expected Results:
If the BSP provides a GUI, an operator abort/cancel mechanism is also provided.

This assertion cannot be tested by calling BioSPI functions. The test lab will use some other means to determine if a BSP passes this assertion.

16.63.3.4 XML:
N.A.

Annex B XML Schema of the test log

This Annex forms an integral part of this Standard.

```
<?xml version="1.0" encoding="utf-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
elementFormDefault="qualified">
   <xs:element name="conformance_test_log" type="conformance_test_log"/>
   <xs:complexType name="conformance_test_log">
      <xs:sequence>
         <xs:element name="TestingLaboratory" type="Contact"/>
         <xs:element name="Vendor" type="Contact"/>
         <xs:element name="Biometric_Product" type="Biometric_Product"/>
         <xs:element name="CTS_ID" type="xs:string"/>
         <xs:element name="test_assertion" type="test_assertion"/>
         <xs:group ref="func_inline_response_group" minOccurs="0"
maxOccurs="unbounded"/>
      </xs:sequence>
      <xs:attribute name="date_time" type="xs:string" use="required"/>
      <xs:attribute name="standard" type="xs:string"/>
   </xs:complexType>
   <xs:group name="func_inline_response_group">
      <xs:choice>
         <xs:element name="function" type="function"/>
         <xs:element name="inline_response" type="inline_response"/>
      </xs:choice>
   </xs:group>
   <xs:complexType name="Contact">
      <xs:sequence>
         <xs:element name="Name" type="xs:string"/>
         <xs:element name="Address" type="Address"/>
         <xs:element name="Phone" type="xs:string"/>
         <xs:element name="Email" type="xs:string" minOccurs="0"/>
         <xs:element name="Url" type="xs:string" minOccurs="0"/>
      </xs:sequence>
   </xs:complexType>
   <xs:complexType name="Address">
      <xs:sequence>
         <xs:element name="Street" type="xs:string"/>
         <xs:element name="City" type="xs:string"/>
         <xs:element name="StateOrProvince" type="xs:string" minOccurs="0"/>
         <xs:element name="ZipOrPostalCode" type="xs:string"/>
         <xs:element name="Country" type="xs:string"/>
      </xs:sequence>
   </xs:complexType>
   <xs:complexType name="Biometric_Product">
      <xs:sequence>
         <xs:element name="Description" type="xs:string"/>
      </xs:sequence>
      <xs:attribute name="Name" type="xs:string" use="required"/>
      <xs:attribute name="SerialNo" type="xs:string" use="required"/>
   </xs:complexType>
   <xs:complexType name="test_assertion">
      <xs:sequence>
         <xs:element name="package_name" type="xs:string"/>
         <xs:element name="assertion_name" type="xs:string"/>
         <xs:element name="description" type="xs:string"/>
         <xs:element name="input" type="inputOutputType" minOccurs="0"
maxOccurs="unbounded"/>
      </xs:sequence>
```

```
   </xs:complexType>
   <xs:complexType name="inputOutputType">
      <xs:attribute name="name" type="xs:NCName" use="required"/>
      <xs:attribute name="value" type="xs:string" use="required"/>
   </xs:complexType>
   <xs:complexType name="function">
      <xs:sequence>
         <xs:element name="name" type="xs:string"/>
         <xs:element name="input" type="inputOutputType" minOccurs="0"
maxOccurs="unbounded"/>
         <xs:element name="output" type="inputOutputType" minOccurs="0"
maxOccurs="unbounded"/>
         <xs:element name="return" type="return"/>
      </xs:sequence>
      <xs:attribute name="dir" type="calldirection" default="outgoing"/>
   </xs:complexType>
   <xs:simpleType name="calldirection">
      <xs:restriction base="xs:token">
         <xs:enumeration value="outgoing"/>
         <xs:enumeration value="incoming"/>
      </xs:restriction>
   </xs:simpleType>
   <xs:complexType name="return">
      <xs:attribute name="value" type="xs:string" use="required"/>
   </xs:complexType>
   <xs:complexType name="inline_response">
      <xs:sequence>
         <xs:element name="asserted_condition" type="xs:string" minOccurs="0"
maxOccurs="unbounded"/>
         <xs:element name="conformance" minOccurs="0" maxOccurs="unbounded">
            <xs:simpleType>
               <xs:restriction base="xs:string">
                  <xs:enumeration value="error"/>
                  <xs:enumeration value="pass"/>
                  <xs:enumeration value="fail"/>
                  <xs:enumeration value="undecided"/>
               </xs:restriction>
            </xs:simpleType>
         </xs:element>
      </xs:sequence>
   </xs:complexType>
</xs:schema>
```

Annex C. Recommendations for Implementers

This annex forms an informative part of this Standard.

[....TO BE DONE. This annex must provide recommendations.]

Bibliography

ISO/IEC Guide 2:1996, Standardization and related activities – General vocabulary

ISO/IEC Guide 7:1994, Guidelines for drafting standards suitable for use for conformity assessment

ISO/IEC Guide 23:1982, Methods of indicating conformity with standards for third-party certification systems

ISO/IEC 17025:1999, General requirements for the competence of testing and calibration laboratories

ISO/IEC Guide 58:1993, Calibration and testing laboratory accreditation systems – General requirements for operation and recognition

ISO/IEC Guide 61:1996, General requirements for assessment and accreditation of certification/registration bodies

ISO/IEC Guide 65:1996, General requirements for bodies operating product certification systems

ISO/IEC Standard 9646:1991, Open Systems Conformity Testing Methodology and Framework

ISO/IEC TR 10183-1:????, Text and Office Systems – Office Document Architecture (ODA) and Interchange Format

ISO/IEC Standard 10641:1993, Conformity Testing of Implementations of Graphics Software

ISO/IEC Standard 13210:1999, Information Technology – Requirements and Guidelines for Test Methods Specifications and Test Method Implementations for Measuring Conformity to POSIX Standards

ISO/IEC Standard 18009:1999, Information Technology – Programming Languages –Ada Conformity Assessment of a Language Processor

M. D. Hogan, Overview of Conformity Assessment, NIST ITL, 2003

M.W. Jensen, J. F. Leathrum, Conforance Test Design Methodology and Assertion Driven Test Case Generator for the Conformance Testing of Interface Standards, Clemson University, 1991

M. Gray, A. Goldfine, L. Rosenthal, L. Carnahan, Conformity Testing, NIST

L. Rosenthal, M. Skall, L. Carnahan, Conformity Testing and Certification Framework, NIST, 2001

L. Carnahan, L. Rosenthal, M. Skall, Conformity Testing and Certification Model for Software Specifications, NIST

S.L. Stewart, NISTIR 5679 Roadmap for the Computer Integrated Manufacturing Application Framework, NIST, 1995

Rosenblum, David S. A Practical Approach to Programming with Assertions, IEEE Transactions on Software Engineering, Vol. 21, No. 1, 1995

Osterweil, Leon & Lori A. Clarke, A Proposed Testing and Research Analysis Initiative, IEEE Software, September 1992

Voas, Jeffery, How Assertions Can Increase Test Effectiveness, IEEE Software, 1997

Fong, Elizabeth, Conformance Test Framework for Government Smart Card, NIST ITL/SDCTC

Voas, J., M. Schatz, and M. Schmidt, The Testability-Based Assertion Placement Tool For Object-Oriented Software, NIST GCR 98-735