BS ISO/IEC 24709-1:2017

BSI Standards Publication

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

Part 1: Methods and procedures

Documents
Document Sharing Hub

Full document details: BS ISO/IEC 24709-1:2017

National foreword

This British Standard is the UK implementation of ISO/IEC 24709-1:2017. It supersedes BS ISO/IEC 24709-1:2007, which is withdrawn.

The UK participation in its preparation was entrusted to Technical Committee IST/44, Biometrics.

A list of organizations represented on this committee can be obtained on request to its secretary.

This publication does not purport to include all the necessary provisions of a contract. Users are responsible for its correct application.

© The British Standards Institution 2017
Published by BSI Standards Limited 2017

ISBN 978 0 580 81672 7

ICS 35.240.15

Compliance with a British Standard cannot confer immunity from legal obligations.

This British Standard was published under the authority of the Standards Policy and Strategy Committee on 31 December 2017.

Amendments/corrigenda issued since publication

Date	Text affected

Documents
Document Sharing Hub

Full document details: BS ISO/IEC 24709-1:2017

INTERNATIONAL STANDARD

ISO/IEC 24709-1

Second edition
2017-11

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

Part 1: Methods and procedures

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

Partie 1: Méthodes et procédures

Reference number
ISO/IEC 24709-1:2017(E)

Documents
Document Sharing Hub

© ISO/IEC 2017

Full document details: BS ISO/IEC 24709-1:2017

COPYRIGHT PROTECTED DOCUMENT

© ISO/IEC 2017, Published in Switzerland

All rights reserved. Unless otherwise specified, no part of this publication may be reproduced or utilized otherwise in any form or by any means, electronic or mechanical, including photocopying, or posting on the internet or an intranet, without prior written permission. Permission can be requested from either ISO at the address below or ISO's member body in the country of the requester.

ISO copyright office
Ch. de Blandonnet 8 • CP 401
CH-1214 Vernier, Geneva, Switzerland
Tel. +41 22 749 01 11
Fax +41 22 749 09 47
copyright@iso.org
www.iso.org

© ISO/IEC 2017 – All rights reserved

Documents
Document Sharing Hub

Full document details: BS ISO/IEC 24709-1:2017

Contents

Page

Foreword ... **xix**

Introduction ... **xx**

1 Scope ... **1**

2 Normative references ... **1**

3 Terms and definitions ... **1**

4 Abbreviated terms ... **3**

5 Conformance ... **3**

6 Conformance testing methodology ... **4**
- 6.1 General ... 4
  - 6.1.1 Implementation under test ... 4
  - 6.1.2 Test method ... 5
  - 6.1.3 Standard BioAPI components and standard BioAPI interfaces ... 6
  - 6.1.4 Physical architectures ... 7
- 6.2 Conformance testing models ... 8
- 6.3 Abstract test engine ... 13

7 General properties of the assertion language ... **14**
- 7.1 General ... 14
- 7.2 Variables ... 17
- 7.3 Built-in variables ... 18
- 7.4 Representation of integers ... 18
- 7.5 Representation of Booleans ... 19
- 7.6 Representation of universally unique identifiers (UUIDs) ... 19
- 7.7 Representation of octet strings ... 19
- 7.8 XML documents ... 20

8 Elements of the assertion language ... **20**
- 8.1 Element `<package>` ... 20
  - 8.1.1 Syntax ... 20
  - 8.1.2 Semantics ... 20
  - 8.1.3 Example (non-normative) ... 21
- 8.2 Element `<assertion>` (child of `<package>`) ... 21
  - 8.2.1 Syntax ... 21
  - 8.2.2 Semantics ... 22
  - 8.2.3 Example (non-normative) ... 22
- 8.3 Element `<input>` (child of `<assertion>`) ... 23
  - 8.3.1 Syntax ... 23
  - 8.3.2 Semantics ... 23
  - 8.3.3 Example (non-normative) ... 23
- 8.4 Element `<invoke>` (child of `<assertion>`) ... 23
  - 8.4.1 Syntax ... 23
  - 8.4.2 Semantics ... 24
  - 8.4.3 Example (non-normative) ... 24
- 8.5 Element `<input>` (child of `<invoke>`) ... 24
  - 8.5.1 Syntax ... 24
  - 8.5.2 Semantics ... 25
  - 8.5.3 Example (non-normative) ... 25
- 8.6 Element `<output>` (child of `<invoke>`) ... 25
  - 8.6.1 Syntax ... 25
  - 8.6.2 Semantics ... 26
- 8.7 Element `<return>` (child of `<invoke>`) ... 26
  - 8.7.1 Syntax ... 26
  - 8.7.2 Semantics ... 27

© ISO/IEC 2017 – All rights reserved

AVA Documents
Document Sharing Hub

Full document details: BS ISO/IEC 24709-1:2017

8.8 Element <bind> (child of <assertion>) 27
8.8.1 Syntax 27
8.8.2 Semantics 27
8.9 Element <activity> (child of <package>) 28
8.9.1 Syntax 28
8.9.2 Semantics 29
8.9.3 Example (non-normative) 32
8.10 Element <input> (child of <activity>) 35
8.10.1 Syntax 35
8.10.2 Semantics 35
8.11 Element <output> (child of <activity>) 35
8.11.1 Syntax 35
8.11.2 Semantics 35
8.12 Element <set> 36
8.12.1 Syntax 36
8.12.2 Semantics 36
8.13 Element <add> 37
8.13.1 Syntax 37
8.13.2 Semantics 37
8.14 Element <subtract> 38
8.14.1 Syntax 38
8.14.2 Semantics 38
8.15 Element <invoke> (child of <activity>) 39
8.15.1 Syntax 39
8.15.2 Semantics 39
8.15.3 Example (non-normative) 42
8.16 Element <only_if> 42
8.16.1 Syntax 42
8.16.2 Semantics 43
8.17 Element <wait_until> 43
8.17.1 Syntax 43
8.17.2 Semantics 44
8.18 Element <assert_condition> 45
8.18.1 Syntax 45
8.18.2 Semantics 45
8.19 Element <and> 46
8.19.1 Syntax 46
8.19.2 Semantics 46
8.20 Element <or> 47
8.20.1 Syntax 47
8.20.2 Semantics 47
8.21 Element <xor> 47
8.21.1 Syntax 47
8.21.2 Semantics 48
8.22 Element <not> 48
8.22.1 Syntax 48
8.22.2 Semantics 48
8.23 Element <equal_to> 49
8.23.1 Syntax 49
8.23.2 Semantics 49
8.24 Element <not_equal_to> 50
8.24.1 Syntax 50
8.24.2 Semantics 50
8.25 Element <greater_than> 50
8.25.1 Syntax 50
8.25.2 Semantics 50
8.26 Element <greater_than_or_equal_to> 50
8.26.1 Syntax 50

AVA Documents
Document Sharing Hub

© ISO/IEC 2017 – All rights reserved

Full document details: BS ISO/IEC 24709-1:2017

8.26.2 Semantics ... 50
8.27 Element <less_than> ... 50
8.27.1 Syntax ... 50
8.27.2 Semantics ... 50
8.28 Element <less_than_or_equal_to> ... 51
8.28.1 Syntax ... 51
8.28.2 Semantics ... 51
8.29 Element <same_as> ... 51
8.29.1 Syntax ... 51
8.29.2 Semantics ... 51
8.30 Element <not_same_as> ... 52
8.30.1 Syntax ... 52
8.30.2 Semantics ... 52
8.31 Element <existing> ... 52
8.31.1 Syntax ... 52
8.31.2 Semantics ... 52
8.32 Element <not_existing> ... 53
8.32.1 Syntax ... 53
8.32.2 Semantics ... 53

9 Standard interface functions ... 53
9.1 General ... 53
9.2 Parameter groups ... 60
9.2.1 General ... 60
9.2.2 Parameter group “Biometric type” ... 60
9.2.3 Parameter group “Operations” ... 62
9.2.4 Parameter group “Options” ... 64
9.2.5 Parameter group “Events” ... 66
9.2.6 Parameter group “Biometric data type” ... 67
9.2.7 Parameter group “Biometric subtype” ... 68
9.2.8 Parameter group “Date” ... 69
9.2.9 Parameter group “Date and time” ... 70
9.2.10 Parameter group “Framework schema” ... 71
9.2.11 Parameter group “BSP schema” ... 72
9.2.12 Parameter group “BFP schema” ... 76
9.2.13 Parameter group “Unit schema” ... 78
9.2.14 Parameter group “BIR header” ... 81
9.2.15 Parameter group “BIR” ... 84
9.2.16 Parameter group “Input BIR” ... 85
9.2.17 Parameter group “Identify population” ... 86
9.2.18 Parameter group “Candidate” ... 88
9.2.19 Parameter group “GUI state” ... 90
9.2.20 Parameter group “Access type” ... 90
9.2.21 Parameter group “GUI Event Subscription” ... 91
9.2.22 Parameter group “Key Information” ... 92
9.2.23 Parameter group “Key Transport” ... 93
9.2.24 Parameter group “MAC Algorithm Information” ... 93
9.2.25 Parameter group “Security Options Mask” ... 94
9.2.26 Parameter group “Security Profile” ... 95
9.2.27 Parameter group “ACBio parameters” ... 97
9.2.28 Parameter group “BFP List Element parameters” ... 98
9.2.29 Parameter group “Biometric subtype mask” ... 98
9.2.30 Parameter group “ENCRYPTION INFO” ... 100
9.3 BioAPI_Init ... 101
9.3.1 Function invocation scheme ... 101
9.3.2 Constraints on the parameters ... 101
9.3.3 Function invocation input ... 101
9.3.4 Function invocation output ... 101
9.3.5 Bound activity invocation input ... 101

© ISO/IEC 2017 – All rights reserved

AVA Documents
Document Sharing Hub
Full document details: BS ISO/IEC 24709-1:2017

9.3.6 Bound activity invocation output 102
9.4 BioAPI_Terminate 102
9.4.1 Function invocation scheme 102
9.4.2 Constraints on the parameters 102
9.4.3 Function invocation input 102
9.4.4 Function invocation output 102
9.4.5 Bound activity invocation input 102
9.4.6 Bound activity invocation output 102
9.5 BioAPI_GetFrameworkInfo 103
9.5.1 Function invocation scheme 103
9.5.2 Constraints on the parameters 103
9.5.3 Function invocation input 103
9.5.4 Function invocation output 103
9.5.5 Bound activity invocation input 104
9.5.6 Bound activity invocation output 104
9.6 BioAPI_EnumBSPs 104
9.6.1 Function invocation scheme 104
9.6.2 Constraints on the parameters 105
9.6.3 Function invocation input 105
9.6.4 Function invocation output 105
9.6.5 Bound activity invocation input 105
9.6.6 Bound activity invocation output 106
9.7 BioAPI_BSPLoad 106
9.7.1 Function invocation scheme 106
9.7.2 Constraints on the parameters 106
9.7.3 Function invocation input 106
9.7.4 Function invocation output 107
9.7.5 Bound activity invocation input 107
9.7.6 Bound activity invocation output 107
9.8 BioAPI_BSPUnload 107
9.8.1 Function invocation scheme 107
9.8.2 Constraints on the parameters 108
9.8.3 Function invocation input 108
9.8.4 Function invocation output 108
9.8.5 Bound activity invocation input 108
9.8.6 Bound activity invocation output 109
9.9 BioAPI_BSPAttach 109
9.9.1 Function invocation scheme 109
9.9.2 Constraints on the parameters 109
9.9.3 Function invocation input 110
9.9.4 Function invocation output 110
9.9.5 Bound activity invocation input 110
9.9.6 Bound activity invocation output 111
9.10 BioAPI_BSPDetach 111
9.10.1 Function invocation scheme 111
9.10.2 Constraints on the parameters 111
9.10.3 Function invocation input 111
9.10.4 Function invocation output 112
9.10.5 Bound activity invocation input 112
9.10.6 Bound activity invocation output 112
9.11 BioAPI_QueryUnits 112
9.11.1 Function invocation scheme 112
9.11.2 Constraints on the parameters 113
9.11.3 Function invocation input 113
9.11.4 Function invocation output 114
9.11.5 Bound activity invocation input 114
9.11.6 Bound activity invocation output 114
9.12 BioAPI_EnumBFPs 114

© ISO/IEC 2017 – All rights reserved
Documents
Document Sharing Hub
Full document details: BS ISO/IEC 24709-1:2017

9.12.1 Function invocation scheme 114
9.12.2 Constraints on the parameters 115
9.12.3 Function invocation input 115
9.12.4 Function invocation output 116
9.12.5 Bound activity invocation input 116
9.12.6 Bound activity invocation output 116
9.13 BioAPI_QueryBFPs 116
9.13.1 Function invocation scheme 116
9.13.2 Constraints on the parameters 117
9.13.3 Function invocation input 117
9.13.4 Function invocation output 118
9.13.5 Bound activity invocation input 118
9.13.6 Bound activity invocation output 118
9.14 BioAPI_ControlUnit 118
9.14.1 Function invocation scheme 118
9.14.2 Constraints on the parameters 119
9.14.3 Function invocation input 119
9.14.4 Function invocation output 119
9.14.5 Bound activity invocation input 120
9.14.6 Bound activity invocation output 120
9.15 BioAPI_LinkToEndpoint 120
9.15.1 Function invocation scheme 120
9.15.2 Constraints on the parameter 120
9.15.3 Function invocation input 121
9.15.4 Function invocation output 121
9.15.5 Bound activity invocation input 121
9.15.6 Bound activity invocation output 121
9.16 BioAPI_UnlinkFromEndpoint 121
9.16.1 Function invocation scheme 121
9.16.2 Constraints on the parameter 121
9.16.3 Function invocation input 122
9.16.4 Function invocation output 122
9.16.5 Bound activity invocation input 122
9.16.6 Bound activity invocation output 122
9.17 BioAPI_EnumFrameworks 122
9.17.1 Function invocation scheme 122
9.17.2 Constraints on the parameters 123
9.17.3 Function invocation input 123
9.17.4 Function invocation output 123
9.17.5 Bound activity invocation input 123
9.17.6 Bound activity invocation output 123
9.18 BioAPI_FreeBIRHandle 124
9.18.1 Function invocation scheme 124
9.18.2 Constraints on the parameters 124
9.18.3 Function invocation input 124
9.18.4 Function invocation output 124
9.18.5 Bound activity invocation input 124
9.18.6 Bound activity invocation output 125
9.19 BioAPI_GetBIRFromHandle 125
9.19.1 Function invocation scheme 125
9.19.2 Constraints on the parameters 125
9.19.3 Function invocation input 126
9.19.4 Function invocation output 126
9.19.5 Bound activity invocation input 126
9.19.6 Bound activity invocation output 126
9.20 BioAPI_GetHeaderFromHandle 126
9.20.1 Function invocation scheme 126
9.20.2 Constraints on the parameters 127

© ISO/IEC 2017 – All rights reserved

Documents
Document Sharing Hub
Full document details: BS ISO/IEC 24709-1:2017

9.20.3 Function invocation input 127
9.20.4 Function invocation output 127
9.20.5 Bound activity invocation input 128
9.20.6 Bound activity invocation output 128
9.21 BioAPI_EnableEvents 128
9.21.1 Function invocation scheme 128
9.21.2 Constraints on the parameters 128
9.21.3 Function invocation input 129
9.21.4 Function invocation output 129
9.21.5 Bound activity invocation input 129
9.21.6 Bound activity invocation output 129
9.22 BioAPI_NotifyGUIProgressEvent 130
9.22.1 Function invocation scheme 130
9.22.2 Constraints on the parameters 131
9.22.3 Function invocation input 131
9.22.4 Function invocation output 131
9.22.5 Bound activity invocation input 132
9.22.6 Bound activity invocation output 132
9.23 BioAPI_NotifyGUISelectEvent 132
9.23.1 Function invocation scheme 132
9.23.2 Constraints on the parameters 133
9.23.3 Function invocation input 134
9.23.4 Function invocation output 134
9.23.5 Bound activity invocation input 134
9.23.6 Bound activity invocation output 135
9.24 BioAPI_NotifyGUIStateEvent 135
9.24.1 Function invocation scheme 135
9.24.2 Constraints on the parameters 136
9.24.3 Function invocation input 137
9.24.4 Function invocation output 137
9.24.5 Bound activity invocation input 137
9.24.6 Bound activity invocation output 138
9.25 BioAPI_QueryGUIEventSubscriptions 138
9.25.1 Function invocation scheme 138
9.25.2 Constraints on the parameters 139
9.25.3 Function invocation input 139
9.25.4 Function invocation output 139
9.25.5 Bound activity invocation input 139
9.25.6 Bound activity invocation output 139
9.26 BioAPI_RedirectGUIEvents 140
9.26.1 Function invocation scheme 140
9.26.2 Constraints on the parameters 140
9.26.3 Function invocation input 141
9.26.4 Function invocation output 141
9.26.5 Bound activity invocation input 141
9.26.6 Bound activity invocation output 141
9.27 BioAPI_SubscribeToGUIEvents 142
9.27.1 Function invocation scheme 142
9.27.2 Constraints on the parameters 142
9.27.3 Function invocation input 143
9.27.4 Function invocation output 143
9.27.5 Bound activity invocation input 143
9.27.6 Bound activity invocation output 144
9.28 BioAPI_UnredirctGUIEvents 144
9.28.1 Function invocation scheme 144
9.28.2 Constraints on the parameters 144
9.28.3 Function invocation input 145
9.28.4 Function invocation output 145

© ISO/IEC 2017 – All rights reserved
Documents
Document Sharing Hub
Full document details: BS ISO/IEC 24709-1:2017

9.28.5 Bound activity invocation input 145
9.28.6 Bound activity invocation output 145
9.29 BioAPI_UnsubscribeFromGUIEvents 146
9.29.1 Function invocation scheme 146
9.29.2 Constraints on the parameters 146
9.29.3 Function invocation input 147
9.29.4 Function invocation output 147
9.29.5 Bound activity invocation input 147
9.29.6 Bound activity invocation output 147
9.30 BioAPI_SetGUICallbacks 148
9.30.1 Function invocation scheme 148
9.30.2 Constraints on the parameters 148
9.30.3 Function invocation input 148
9.30.4 Function invocation output 149
9.30.5 Bound activity invocation input 149
9.30.6 Bound activity invocation output 149
9.31 BioAPI_Capture 150
9.31.1 Function invocation scheme 150
9.31.2 Constraints on the parameters 150
9.31.3 Function invocation input 151
9.31.4 Function invocation output 151
9.31.5 Bound activity invocation input 151
9.31.6 Bound activity invocation output 152
9.32 BioAPI_CreateTemplate 152
9.32.1 Function invocation scheme 152
9.32.2 Constraints on the parameters 153
9.32.3 Function invocation input 154
9.32.4 Function invocation output 154
9.32.5 Bound activity invocation input 155
9.32.6 Bound activity invocation output 155
9.33 BioAPI_Process 155
9.33.1 Function invocation scheme 155
9.33.2 Constraints on the parameters 156
9.33.3 Function invocation input 156
9.33.4 Function invocation output 157
9.33.5 Bound activity invocation input 157
9.33.6 Bound activity invocation output 157
9.34 BioAPI_ProcessUsingAuxBIRs 158
9.34.1 Function invocation scheme 158
9.34.2 Constraints on the parameters 158
9.34.3 Function invocation input 159
9.34.4 Function invocation output 159
9.34.5 Bound activity invocation input 159
9.34.6 Bound activity invocation output 160
9.35 BioAPI_VerifyMatch 160
9.35.1 Function invocation scheme 160
9.35.2 Constraints on the parameters 161
9.35.3 Function invocation input 162
9.35.4 Function invocation output 162
9.35.5 Bound activity invocation input 163
9.35.6 Bound activity invocation output 163
9.36 BioAPI_VerifyMatchUsingAuxBIRs 163
9.36.1 Function invocation scheme 163
9.36.2 Constraints on the parameters 164
9.36.3 Function invocation input 165
9.36.4 Function invocation output 165
9.36.5 Bound activity invocation input 166
9.36.6 Bound activity invocation output 166

AVA Documents
Document Sharing Hub

© ISO/IEC 2017 – All rights reserved

Full document details: BS ISO/IEC 24709-1:2017

9.37 BioAPI_IdentifyMatch 167
9.37.1 Function invocation scheme 167
9.37.2 Constraints on the parameters 168
9.37.3 Function invocation input 169
9.37.4 Function invocation output 169
9.37.5 Bound activity invocation input 170
9.37.6 Bound activity invocation output 170
9.38 BioAPI_Decide 170
9.38.1 Function invocation scheme 170
9.38.2 Constraints on the parameters 171
9.38.3 Function invocation input 171
9.38.4 Function invocation output 171
9.38.5 Bound activity invocation input 172
9.38.6 Bound activity invocation output 172
9.39 BioAPI_Fuse 172
9.39.1 Function invocation scheme 172
9.39.2 Constraints on the parameters 173
9.39.3 Function invocation input 173
9.39.4 Function invocation output 173
9.39.5 Bound activity invocation input 173
9.39.6 Bound activity invocation output 174
9.40 BioAPI_Enroll 174
9.40.1 Function invocation scheme 174
9.40.2 Constraints on the parameters 175
9.40.3 Function invocation input 176
9.40.4 Function invocation output 176
9.40.5 Bound activity invocation input 177
9.40.6 Bound activity invocation output 177
9.41 BioAPI_Verify 177
9.41.1 Function invocation scheme 177
9.41.2 Constraints on the parameters 178
9.41.3 Function invocation input 179
9.41.4 Function invocation output 179
9.41.5 Bound activity invocation input 180
9.41.6 Bound activity invocation output 181
9.42 BioAPI_Identify 181
9.42.1 Function invocation scheme 181
9.42.2 Constraints on the parameters 182
9.42.3 Function invocation input 183
9.42.4 Function invocation output 183
9.42.5 Bound activity invocation input 184
9.42.6 Bound activity invocation output 184
9.43 BioAPI_Import 184
9.43.1 Function invocation scheme 184
9.43.2 Constraints on the parameters 185
9.43.3 Function invocation input 185
9.43.4 Function invocation output 186
9.43.5 Bound activity invocation input 186
9.43.6 Bound activity invocation output 187
9.44 BioAPI_Export 187
9.44.1 Function invocation scheme 187
9.44.2 Constraints on the parameters 187
9.44.3 Function invocation input 188
9.44.4 Function invocation output 188
9.44.5 Bound activity invocation input 188
9.44.6 Bound activity invocation output 189
9.45 BioAPI_PresetIdentifyPopulation 189
9.45.1 Function invocation scheme 189

Documents
Document Sharing Hub

© ISO/IEC 2017 – All rights reserved

Full document details: BS ISO/IEC 24709-1:2017

9.45.2 Constraints on the parameters 189
9.45.3 Function invocation input 189
9.45.4 Function invocation output 190
9.45.5 Bound activity invocation input 190
9.45.6 Bound activity invocation output 190
9.46 BioAPI_Transform 190
9.46.1 Function invocation scheme 190
9.46.2 Constraints on the parameters 191
9.46.3 Function invocation input 191
9.46.4 Function invocation output 192
9.46.5 Bound activity invocation input 192
9.46.6 Bound activity invocation output 192
9.47 BioAPI_DbOpen 192
9.47.1 Function invocation scheme 192
9.47.2 Constraints on the parameters 193
9.47.3 Function invocation input 193
9.47.4 Function invocation output 194
9.47.5 Bound activity invocation input 194
9.47.6 Bound activity invocation output 194
9.48 BioAPI_DbClose 195
9.48.1 Function invocation scheme 195
9.48.2 Constraints on the parameters 195
9.48.3 Function invocation input 195
9.48.4 Function invocation output 195
9.48.5 Bound activity invocation input 195
9.48.6 Bound activity invocation output 196
9.49 BioAPI_DbCreate 196
9.49.1 Function invocation scheme 196
9.49.2 Constraints on the parameters 196
9.49.3 Function invocation input 197
9.49.4 Function invocation output 197
9.49.5 Bound activity invocation input 197
9.49.6 Bound activity invocation output 197
9.50 BioAPI_DbDelete 198
9.50.1 Function invocation scheme 198
9.50.2 Constraints on the parameters 198
9.50.3 Function invocation input 198
9.50.4 Function invocation output 198
9.50.5 Bound activity invocation input 198
9.50.6 Bound activity invocation output 199
9.51 BioAPI_DbSetMarker 199
9.51.1 Function invocation scheme 199
9.51.2 Constraints on the parameters 199
9.51.3 Function invocation input 200
9.51.4 Function invocation output 200
9.51.5 Bound activity invocation input 200
9.51.6 Bound activity invocation output 200
9.52 BioAPI_DbFreeMarker 200
9.52.1 Function invocation scheme 200
9.52.2 Constraints on the parameters 201
9.52.3 Function invocation input 201
9.52.4 Function invocation output 201
9.52.5 Bound activity invocation input 201
9.52.6 Bound activity invocation output 201
9.53 BioAPI_DbStoreBIR 202
9.53.1 Function invocation scheme 202
9.53.2 Constraints on the parameters 202
9.53.3 Function invocation input 203

Documents
Document Sharing Hub

© ISO/IEC 2017 – All rights reserved

Full document details: BS ISO/IEC 24709-1:2017

9.53.4 Function invocation output 203
9.53.5 Bound activity invocation input 203
9.53.6 Bound activity invocation output 203
9.54 BioAPI_DbGetBIR 204
9.54.1 Function invocation scheme 204
9.54.2 Constraints on the parameters 204
9.54.3 Function invocation input 205
9.54.4 Function invocation output 205
9.54.5 Bound activity invocation input 205
9.54.6 Bound activity invocation output 206
9.55 BioAPI_DbGetNextBIR 206
9.55.1 Function invocation scheme 206
9.55.2 Constraints on the parameters 206
9.55.3 Function invocation input 207
9.55.4 Function invocation output 207
9.55.5 Bound activity invocation input 207
9.55.6 Bound activity invocation output 208
9.56 BioAPI_DbDeleteBIR 208
9.56.1 Function invocation scheme 208
9.56.2 Constraints on the parameters 208
9.56.3 Function invocation input 209
9.56.4 Function invocation output 209
9.56.5 Bound activity invocation input 209
9.56.6 Bound activity invocation output 209
9.57 BioAPI_SetPowerMode 209
9.57.1 Function invocation scheme 209
9.57.2 Constraints on the parameters 210
9.57.3 Function invocation input 210
9.57.4 Function invocation output 210
9.57.5 Bound activity invocation input 210
9.57.6 Bound activity invocation output 210
9.58 BioAPI_SetIndicatorStatus 211
9.58.1 Function invocation scheme 211
9.58.2 Constraints on the parameters 211
9.58.3 Function invocation input 211
9.58.4 Function invocation output 211
9.58.5 Bound activity invocation input 211
9.58.6 Bound activity invocation output 212
9.59 BioAPI_GetIndicatorStatus 212
9.59.1 Function invocation scheme 212
9.59.2 Constraints on the parameters 212
9.59.3 Function invocation input 213
9.59.4 Function invocation output 213
9.59.5 Bound activity invocation input 213
9.59.6 Bound activity invocation output 213
9.60 BioAPI_CalibrateSensor 213
9.60.1 Function invocation scheme 213
9.60.2 Constraints on the parameters 214
9.60.3 Function invocation input 214
9.60.4 Function invocation output 214
9.60.5 Bound activity invocation input 214
9.60.6 Bound activity invocation output 214
9.61 BioAPI_Cancel 215
9.61.1 Function invocation scheme 215
9.61.2 Constraints on the parameters 215
9.61.3 Function invocation input 215
9.61.4 Function invocation output 215
9.61.5 Bound activity invocation input 215

© ISO/IEC 2017 – All rights reserved
Documents
Document Sharing Hub
Full document details: BS ISO/IEC 24709-1:2017

9.61.6 Bound activity invocation output 215
9.62 BioAPI_Free 216
9.62.1 Function invocation scheme 216
9.62.2 Constraints on the parameters 216
9.62.3 Function invocation input 216
9.62.4 Function invocation output 216
9.62.5 Bound activity invocation input 216
9.62.6 Bound activity invocation output 216
9.63 BioAPI_Util_InstallBSP 217
9.63.1 Function invocation scheme 217
9.63.2 Constraints on the parameters 217
9.63.3 Function invocation input 218
9.63.4 Function invocation output 218
9.63.5 Bound activity invocation input 218
9.63.6 Bound activity invocation output 218
9.64 BioAPI_Util_InstallBFP 219
9.64.1 Function invocation scheme 219
9.64.2 Constraints on the parameters 219
9.64.3 Function invocation input 220
9.64.4 Function invocation output 220
9.64.5 Bound activity invocation input 220
9.64.6 Bound activity invocation output 220
9.65 BioSPI_BSPLoad 220
9.65.1 Function invocation scheme 220
9.65.2 Constraints on the parameters 221
9.65.3 Function invocation input 221
9.65.4 Function invocation output 222
9.65.5 Bound activity invocation input 222
9.65.6 Bound activity invocation output 222
9.65.7 Default output 222
9.66 BioSPI_BSPUnload 222
9.66.1 Function invocation scheme 222
9.66.2 Constraints on the parameters 222
9.66.3 Function invocation input 223
9.66.4 Function invocation output 223
9.66.5 Bound activity invocation input 223
9.66.6 Bound activity invocation output 223
9.66.7 Default output 223
9.67 BioSPI_BSPAttach 223
9.67.1 Function invocation scheme 223
9.67.2 Constraints on the parameters 224
9.67.3 Function invocation input 224
9.67.4 Function invocation output 225
9.67.5 Bound activity invocation input 225
9.67.6 Bound activity invocation output 225
9.67.7 Default output 225
9.68 BioSPI_BSPDetach 225
9.68.1 Function invocation scheme 225
9.68.2 Constraints on the parameters 226
9.68.3 Function invocation input 226
9.68.4 Function invocation output 226
9.68.5 Bound activity invocation input 226
9.68.6 Bound activity invocation output 226
9.68.7 Default output 226
9.69 BioSPI_QueryUnits 226
9.69.1 Function invocation scheme 226
9.69.2 Bound activity invocation output 227
9.69.3 Default output 228

Documents
Document Sharing Hub

© ISO/IEC 2017 – All rights reserved

Full document details: BS ISO/IEC 24709-1:2017

9.70 BioSPI_QueryBFPs 228
9.70.1 Function invocation scheme 228
9.70.2 Bound activity invocation output 229
9.70.3 Default output 229
9.71 BioSPI_ControlUnit 229
9.71.1 Function invocation scheme 229
9.71.2 Bound activity invocation output 230
9.71.3 Default output 230
9.72 BioSPI_SetConfiguration 231
9.72.1 Function invocation scheme 231
9.72.2 Bound activity invocation output 231
9.72.3 Default output 231
9.73 BioSPI_FreeBIRHandle 232
9.73.1 Function invocation scheme 232
9.73.2 Bound activity invocation output 232
9.73.3 Default output 232
9.74 BioSPI_GetBIRFromHandle 232
9.74.1 Function invocation scheme 232
9.74.2 Bound activity invocation output 233
9.74.3 Default output 233
9.75 BioSPI_GetHeaderFromHandle 233
9.75.1 Function invocation scheme 233
9.75.2 Bound activity invocation output 234
9.75.3 Default output 234
9.76 BioSPI_SubscribeToGUIEvents 234
9.76.1 Function invocation scheme 234
9.76.2 Bound activity invocation output 235
9.76.3 Default output 235
9.77 BioSPI_UnsubscribeFromGUIEvents 235
9.77.1 Function invocation scheme 235
9.77.2 Bound activity invocation output 235
9.77.3 Default output 236
9.78 BioSPI_EnableEvents 236
9.78.1 Function invocation scheme 236
9.78.2 Bound activity invocation output 236
9.78.3 Default output 236
9.79 BioSPI_SetGUICallbacks 237
9.79.1 Function invocation scheme 237
9.79.2 Bound activity invocation output 237
9.79.3 Default output 237
9.80 BioSPI_Capture 238
9.80.1 Function invocation scheme 238
9.80.2 Bound activity invocation output 238
9.80.3 Default output 239
9.81 BioSPI_CreateTemplate 239
9.81.1 Function invocation scheme 239
9.81.2 Bound activity invocation output 240
9.81.3 Default output 240
9.82 BioSPI_Process 241
9.82.1 Function invocation scheme 241
9.82.2 Bound activity invocation output 241
9.82.3 Default output 242
9.83 BioSPI_ProcessUsingAuxBIRs 242
9.83.1 Function invocation scheme 242
9.83.2 Bound activity invocation output 243
9.83.3 Default output 243
9.84 BioSPI_VerifyMatch 243
9.84.1 Function invocation scheme 243

AVA Documents
Document Sharing Hub

© ISO/IEC 2017 – All rights reserved

Full document details: BS ISO/IEC 24709-1:2017

9.84.2 Bound activity invocation output ... 244
9.84.3 Default output ... 245
9.85 BioSPI_VerifyMatchUsingAuxBIRs ... 245
9.85.1 Function invocation scheme ... 245
9.85.2 Bound activity invocation output ... 246
9.85.3 Default output ... 247
9.86 BioSPI_IdentifyMatch ... 247
9.86.1 Function invocation scheme ... 247
9.86.2 Bound activity invocation output ... 248
9.86.3 Default output ... 249
9.87 BioSPI_Decide ... 249
9.87.1 Function invocation scheme ... 249
9.87.2 Bound activity invocation output ... 250
9.87.3 Default output ... 250
9.88 BioSPI_Fuse ... 251
9.88.1 Function invocation scheme ... 251
9.88.2 Bound activity invocation output ... 251
9.88.3 Default output ... 252
9.89 BioSPI_Enroll ... 252
9.89.1 Function invocation scheme ... 252
9.89.2 Bound activity invocation output ... 253
9.89.3 Default output ... 254
9.90 BioSPI_Verify ... 254
9.90.1 Function invocation scheme ... 254
9.90.2 Bound activity invocation output ... 255
9.90.3 Default output ... 256
9.91 BioSPI_Identify ... 256
9.91.1 Function invocation scheme ... 256
9.91.2 Bound activity invocation output ... 257
9.91.3 Default output ... 258
9.92 BioSPI_Import ... 258
9.92.1 Function invocation scheme ... 258
9.92.2 Bound activity invocation output ... 259
9.92.3 Default output ... 259
9.93 BioSPI_Export ... 260
9.93.1 Function invocation scheme ... 260
9.93.2 Bound activity invocation output ... 260
9.93.3 Default output ... 261
9.94 BioSPI_PresetIdentifyPopulation ... 261
9.94.1 Function invocation scheme ... 261
9.94.2 Bound activity invocation output ... 261
9.94.3 Default output ... 262
9.95 BioSPI_Transform ... 262
9.95.1 Function invocation scheme ... 262
9.95.2 Bound activity invocation output ... 262
9.95.3 Default output ... 263
9.96 BioSPI_DbOpen ... 263
9.96.1 Function invocation scheme ... 263
9.96.2 Bound activity invocation output ... 264
9.96.3 Default output ... 264
9.97 BioSPI_DbClose ... 265
9.97.1 Function invocation scheme ... 265
9.97.2 Bound activity invocation output ... 265
9.97.3 Default output ... 265
9.98 BioSPI_DbCreate ... 265
9.98.1 Function invocation scheme ... 265
9.98.2 Bound activity invocation output ... 266
9.98.3 Default output ... 266

AVA Documents
Document Sharing Hub

© ISO/IEC 2017 – All rights reserved

Full document details: BS ISO/IEC 24709-1:2017

9.99 BioSPI_DbDelete 267
9.99.1 Function invocation scheme 267
9.99.2 Bound activity invocation output 267
9.99.3 Default output 267
9.100 BioSPI_DbSetMarker 267
9.100.1 Function invocation scheme 267
9.100.2 Bound activity invocation output 268
9.100.3 Default output 268
9.101 BioSPI_DbFreeMarker 268
9.101.1 Function invocation scheme 268
9.101.2 Bound activity invocation output 269
9.101.3 Default output 269
9.102 BioSPI_DbStoreBIR 269
9.102.1 Function invocation scheme 269
9.102.2 Bound activity invocation output 269
9.102.3 Default output 270
9.103 BioSPI_DbGetBIR 270
9.103.1 Function invocation scheme 270
9.103.2 Bound activity invocation output 271
9.103.3 Default output 271
9.104 BioSPI_DbGetNextBIR 271
9.104.1 Function invocation scheme 271
9.104.2 Bound activity invocation output 272
9.104.3 Default output 272
9.105 BioSPI_DbDeleteBIR 273
9.105.1 Function invocation scheme 273
9.105.2 Bound activity invocation output 273
9.105.3 Default output 273
9.106 BioSPI_SetPowerMode 273
9.106.1 Function invocation scheme 273
9.106.2 Bound activity invocation output 274
9.106.3 Default output 274
9.107 BioSPI_SetIndicatorStatus 274
9.107.1 Function invocation scheme 274
9.107.2 Bound activity invocation output 275
9.107.3 Default output 275
9.108 BioSPI_GetIndicatorStatus 275
9.108.1 Function invocation scheme 275
9.108.2 Bound activity invocation output 275
9.108.3 Default output 276
9.109 BioSPI_CalibrateSensor 276
9.109.1 Function invocation scheme 276
9.109.2 Bound activity invocation output 276
9.109.3 Default output 276
9.110 BioSPI_Cancel 277
9.110.1 Function invocation scheme 277
9.110.2 Bound activity invocation output 277
9.110.3 Default output 277
9.111 BioSPI_Free 277
9.111.1 Function invocation scheme 277
9.111.2 Bound activity invocation output 278
9.111.3 Default output 278
9.112 BioAPI_GUI_STATE_EVENT_HANDLER 278
9.112.1 Function invocation scheme 278
9.112.2 Bound activity invocation output 279
9.112.3 Default output 280
9.113 BioSPI_EventHandler 280
9.113.1 Function invocation scheme 280

© ISO/IEC 2017 – All rights reserved
Documents
Document Sharing Hub
Full document details: BS ISO/IEC 24709-1:2017

9.113.2 Constraints on the parameters 281
9.113.3 Function invocation input 281
9.113.4 Function invocation output 281
9.113.5 Bound activity invocation input 282
9.113.6 Bound activity invocation output 282
9.113.7 Default output 282
9.114 BioAPI_GUI_STATE_CALLBACK 282
9.114.1 Function invocation scheme 282
9.114.2 Constraints on the parameters 283
9.114.3 Bound activity invocation input 284
9.114.4 Bound activity invocation output 284
9.114.5 Default output 284
9.115 BioAPI_GUI_STREAMING_CALLBACK 285
9.115.1 Function invocation scheme 285
9.115.2 Constraints on the parameters 285
9.115.3 Bound activity invocation input 285
9.115.4 Bound activity invocation output 286
9.115.5 Default output 286
9.116 BioSPI_EventHandler 286
9.116.1 Function invocation scheme 286
9.116.2 Constraints on the parameters 286
9.116.3 Function invocation input 287
9.116.4 Function invocation output 287
9.116.5 Bound activity invocation input 287
9.116.6 Bound activity invocation output 287
9.116.7 Default output 287
9.117 BioSPI_GUI_STATE_CALLBACK 288
9.117.1 Function invocation scheme 288
9.117.2 Function invocation input 288
9.117.3 Function invocation output 289
9.118 BioSPI_GUI_STREAMING_CALLBACK 289
9.118.1 Function invocation scheme 289
9.118.2 Function invocation input 290
9.118.3 Function invocation output 290
9.119 BioSPI_BFP_ENUMERATION_HANDLER 290
9.119.1 Function invocation scheme 290
9.119.2 Constraints on the parameters 291
9.119.3 Function invocation input 291
9.119.4 Function invocation output 291
9.119.5 Bound activity invocation input 292
9.119.6 Bound activity invocation output 292
9.119.7 Default output 292
9.120 BioSPI_MEMORY_FREE_HANDLER 292
9.120.1 Function invocation scheme 292
9.120.2 Constraints on the parameters 293
9.120.3 Function invocation input 293
9.120.4 Function invocation output 293
9.120.5 Bound activity invocation input 293
9.120.6 Bound activity invocation output 293
9.120.7 Default output 293
9.121 BioSPI_GUI_PROGRESS_EVENT_HANDLER 293
9.121.1 Function invocation scheme 293
9.121.2 Bound activity invocation output 294
9.121.3 Default output 294
9.122 BioSPI_GUI_SELECT_EVENT_HANDLER 295
9.122.1 Function invocation scheme 295
9.122.2 Bound activity invocation output 296
9.122.3 Default output 296

Documents
Document Sharing Hub

© ISO/IEC 2017 – All rights reserved

Full document details: BS ISO/IEC 24709-1:2017

9.123 BioSPI_GUI_STATE_EVENT_HANDLER ... 296
9.123.1 Function invocation scheme ... 296
9.123.2 Bound activity invocation output ... 297
9.123.3 Default output ... 297

10 Built-in variables ... 297
10.1 Variables whose value never changes ... 297
10.2 Variables whose value may change ... 309

11 Test log ... 310

12 Test report ... 312

13 BioAPI conformance test suite ... 313
13.1 General concepts ... 313
13.2 BioAPI conformance test suite structure ... 314

Annex A (normative) **XML schema of the assertion language ... 316**

Annex B (normative) **ASN.1 schema of the assertion language ... 320**

Annex C (normative) **XML schema for the test log ... 323**

Annex D (informative) **Test method implementation guidelines ... 325**

Annex E (informative) **XML diagrams of the assertion language ... 331**

Bibliography ... 352

© ISO/IEC 2017 – All rights reserved
Documents
Document Sharing Hub
Full document details: BS ISO/IEC 24709-1:2017

Foreword

ISO (the International Organization for Standardization) and IEC (the International Electrotechnical Commission) form the specialized system for worldwide standardization. National bodies that are members of ISO or IEC participate in the development of International Standards through technical committees established by the respective organization to deal with particular fields of technical activity. ISO and IEC technical committees collaborate in fields of mutual interest. Other international organizations, governmental and non-governmental, in liaison with ISO and IEC, also take part in the work. In the field of information technology, ISO and IEC have established a joint technical committee, ISO/IEC JTC 1.

The procedures used to develop this document and those intended for its further maintenance are described in the ISO/IEC Directives, Part 1. In particular the different approval criteria needed for the different types of document should be noted. This document was drafted in accordance with the editorial rules of the ISO/IEC Directives, Part 2 (see www.iso.org/directives).

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. ISO and IEC shall not be held responsible for identifying any or all such patent rights. Details of any patent rights identified during the development of the document will be in the Introduction and/or on the ISO list of patent declarations received (see www.iso.org/patents).

Any trade name used in this document is information given for the convenience of users and does not constitute an endorsement.

For an explanation on the voluntary nature of standards, the meaning of ISO specific terms and expressions related to conformity assessment, as well as information about ISO's adherence to the World Trade Organization (WTO) principles in the Technical Barriers to Trade (TBT) see the following URL: www.iso.org/iso/foreword.html.

This document was prepared by Technical Committee ISO/IEC JTC 1, *Information technology*, Subcommittee SC 37, *Biometrics*.

This second edition cancels and replaces the first edition (ISO/IEC 24709-1:2007), which has been technically revised.

A list of all parts in the ISO/IEC 24709 series can be found on the ISO website.

© ISO/IEC 2017 – All rights reserved

Documents
Document Sharing Hub
Full document details: BS ISO/IEC 24709-1:2017

Introduction

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

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

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

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

Clause 8 defines the elements of the assertion language.

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

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

Clause 11 defines the test log using XML syntax.

Clause 12 defines the test report using XML syntax.

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

Annex A defines the XML schema of the assertion language.

Annex B defines the ASN.1 schema of the assertion language.

Annex C defines the XML schema for the test log.

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

Annex E provides the relationship diagrams for the assertion language.

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

NOTE Rather than make normative references to these documents, this document incorporates appropriate excerpts of their text, in some cases paraphrasing the text or adapting the provisions to the specific circumstances. Therefore, these documents are listed in the Bibliography or are referred in the body text explicitly as appropriate.

© ISO/IEC 2017 – All rights reserved
Documents
Document Sharing Hub
Full document details: BS ISO/IEC 24709-1:2017

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

Part 1: Methods and procedures

1 Scope

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

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

This document is applicable to the development and use of conformance test method specifications, BioAPI conformance test suites, and conformance testing programs for BioAPI-conformant products. It is intended primarily for use by testing organizations, but can be applied by developers and users of test assertions and test method implementations.

2 Normative references

The following documents are referred to in the text in such a way that some or all of their content constitutes requirements of this document. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

ISO 8601, *Data elements and interchange formats — Information interchange — Representation of dates and times*

ISO/IEC 19784-1, *Information technology — Biometric application program interface — Part 1: BioAPI specification*

3 Terms and definitions

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

ISO and IEC maintain terminological databases for use in standardization at the following addresses:

— IEC Electropedia: available at http://www.electropedia.org/

— ISO Online browsing platform: available at http://www.iso.org/obp

3.1
abstract test engine
conceptual machine capable of performing conformance tests on an instance of a standard BioAPI component

© ISO/IEC 2017 – All rights reserved

Documents
Document Sharing Hub
Full document details: BS ISO/IEC 24709-1:2017

3.2
base standard
standard for which a *test method specification* (3.13) is written and/or a *test method implementation* (3.12) is developed

3.3
BioAPI conformity statement
statement that describes conformity of an IUT against relevant BioAPI requirements

3.4
BioAPI conformance test suite
test software used to ascertain *conformance* (3.6) to a specification or a standard that is in conformance to ISO/IEC 19784-1

3.5
certification
acknowledgement that a *validation* (3.17) has been completed and the criteria established by the certifying organization have been met

3.6
conformance
fulfillment by a product, process, or service of all relevant specified conformance requirements

3.7
conformity assessment
any activity concerned with determining directly or indirectly that relevant requirements are fulfilled

Note 1 to entry: See ISO/IEC Guide 2.

3.8
implementation under test
software and hardware, located on one or more system, which implement the standard(s) being tested

3.9
standard BioAPI interface
one of the interfaces specified in ISO/IEC 19784-1, which one standard BioAPI component exposes to another standard BioAPI component

Note 1 to entry: See 6.1.3.8.

3.10
test assertion
assertion
specification for testing a conformance requirement in an *implementation under test* (3.8) in the form of software or procedural methods that generate the test results (also named test outcomes or test verdicts) used for assessment of the conformance requirement

3.11
test case
specification of the actions required to achieve a specific *test purpose* (3.14) or combination of test purposes

[SOURCE: ISO/IEC 9646-1:1994, 3.3.3, modified — The definition has been revised.]

3.12
test method implementation
software, procedures, or other means used to measure *conformance* (3.6)

© ISO/IEC 2017 – All rights reserved
Documents
Document Sharing Hub
Full document details: BS ISO/IEC 24709-1:2017

3.13
test method specification
document that expresses the required functionality and behaviour of a *base standard* (3.2) as *assertions* (3.10) and provides the complete set of conforming *test result codes* (3.15)

[SOURCE: ISO/IEC 13210:1999, 2.2.2.30]

3.14
test purpose
prose description of a narrowly defined objective of testing, focusing on a single conformance requirement, as specified in the appropriate product specification

[SOURCE: ISO/IEC 13210:1999, 2.2.2.32, modified — The definition has been revised.]

3.15
test result code
value that describes the result of a test

3.16
test report
document that presents test results and other information relevant to the execution of the test methods against an *implementation under test* (3.8)

3.17
validation
process of testing software for *conformance* (3.6) to a specific specification

4 Abbreviated terms

API Application Programming Interface

BCS BioAPI Conformity Statement

BIR Biometric Information Record

BSP Biometric Service Provider

CBEFF Common Biometric Exchange Formats Framework

CTS BioAPI Conformance Test Suite

IUT Implementation Under Test

SPI Service Provider Interface

UUID Universally Unique Indentifier

5 Conformance

5.1 A BioAPI conformance test suite conforming to this document shall support one or more conformance testing models (see 6.2) and shall be able to execute any valid test assertion for the testing model(s) that it supports, and that are written in the assertion language specified in Clause 7 to Clause 10.

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

© ISO/IEC 2017 – All rights reserved

Documents
Document Sharing Hub
Full document details: BS ISO/IEC 24709-1:2017

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

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

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

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

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

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

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

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

6 Conformance testing methodology

6.1 General

6.1.1 Implementation under test

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

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

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

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

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

6.1.1.3 To evaluate the conformity of a biometric product, it is necessary to have a statement of the capabilities that have been implemented in conformance with the BioAPI specification, so that the implementation can be tested for conformity against relevant requirements and against those

© ISO/IEC 2017 – All rights reserved
AVA Documents Document Sharing Hub
Full document details: BS ISO/IEC 24709-1:2017

requirements only. Such a statement is called a BioAPI Conformity Statement (BCS, and shall be prepared by the IUT supplier prior to the beginning of the conformance testing. At a minimum, the BCS shall contain an itemized list of all mandatory, optional, and conditional conformity requirements of the BioAPI specification included in the IUT.

6.1.2 Test method

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

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

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

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

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

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

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

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

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

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

b) documented test cases, which will sufficiently assure conformity to this document. The test cases shall be formally represented in the form of test assertions using the assertion language, which can be submitted to the BioAPI conformance test suite;

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

© ISO/IEC 2017 – All rights reserved

Documents
Document Sharing Hub
Full document details: BS ISO/IEC 24709-1:2017

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

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

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

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

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

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

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

6.1.3 Standard BioAPI components and standard BioAPI interfaces

6.1.3.1 This document specifies a methodology for assessing conformance of implementations of ISO/IEC 19784-1. Three types of implementations of ISO/IEC 19784-1 are distinguished (see Figure 1):

a) BioAPI application;

b) BioAPI framework;

c) BioAPI biometric service provider (BSP).

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

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

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

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

© ISO/IEC 2017 – All rights reserved
Documents
Document Sharing Hub
Full document details: BS ISO/IEC 24709-1:2017

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

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

a) notifications of events related to BioAPI units;

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

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

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

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

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

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

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

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

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

6.1.3.6.5 In addition, BioAPI frameworks include functions that support installation and deinstallation of BSPs, and installation and deinstallaton of BFPs. In this document, those functions are included in the BioAPI interface.

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

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

6.1.3.9 This document specifies a BioAPI conformance testing methodology in terms of the three standard BioAPI components and of the four standard BioAPI interfaces.

6.1.4 Physical architectures

6.1.4.1 ISO/IEC 19784-1 does not specify the details of loading in memory for any of the standard BioAPI components, nor does it specify a physical association between standard BioAPI components and

© ISO/IEC 2017 – All rights reserved

AVA Documents Document Sharing Hub
Full document details: BS ISO/IEC 24709-1:2017

nodes, nor does it limit the number of instances of each standard BioAPI component within a node. All this information is therefore platform-specific or implementation-specific.

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

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

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

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

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

6.2 Conformance testing models

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

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

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

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

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

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

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

© ISO/IEC 2017 – All rights reserved
Documents
Document Sharing Hub
Full document details: BS ISO/IEC 24709-1:2017

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

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

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

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

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

© ISO/IEC 2017 – All rights reserved

Documents
Document Sharing Hub
Full document details: BS ISO/IEC 24709-1:2017

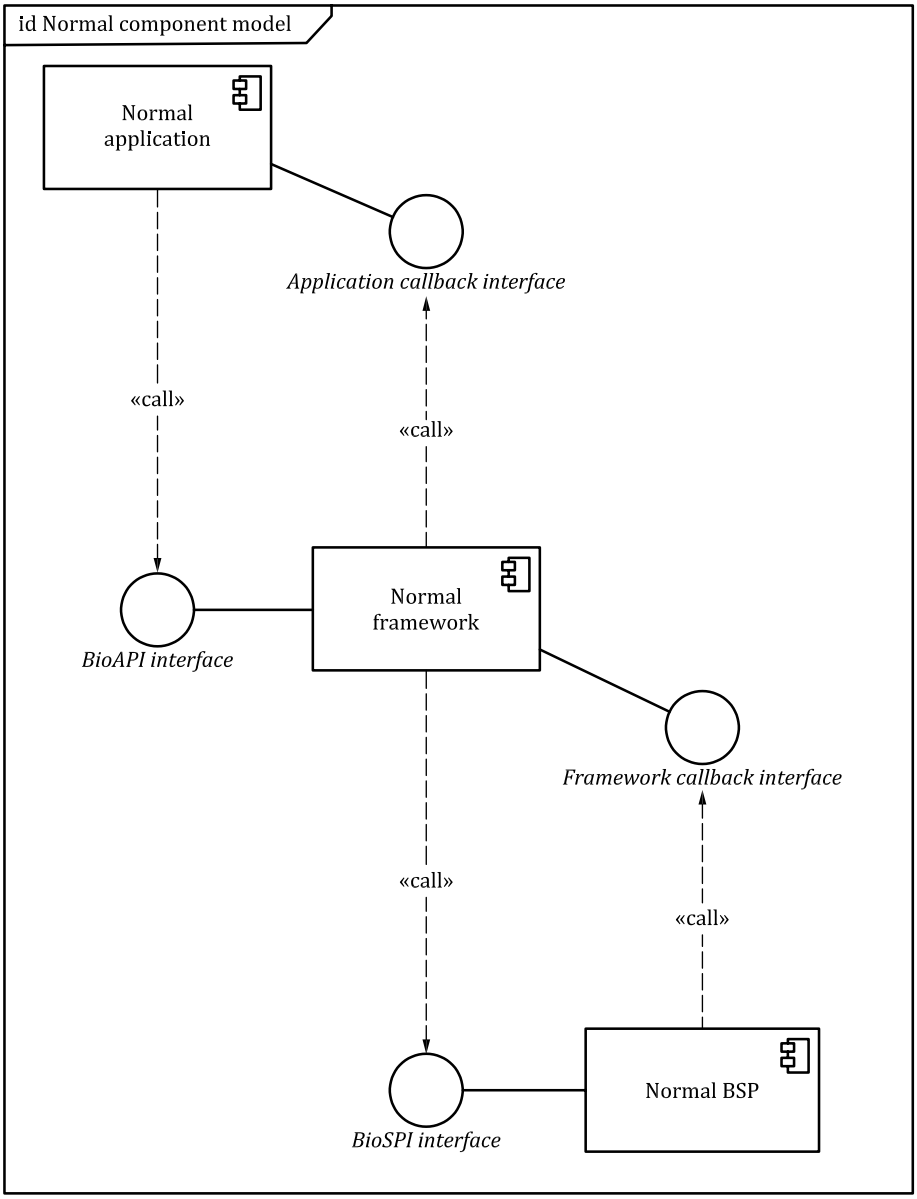

Figure 1 — Normal BioAPI component model

© ISO/IEC 2017 – All rights reserved
Documents
Document Sharing Hub
Full document details: BS ISO/IEC 24709-1:2017

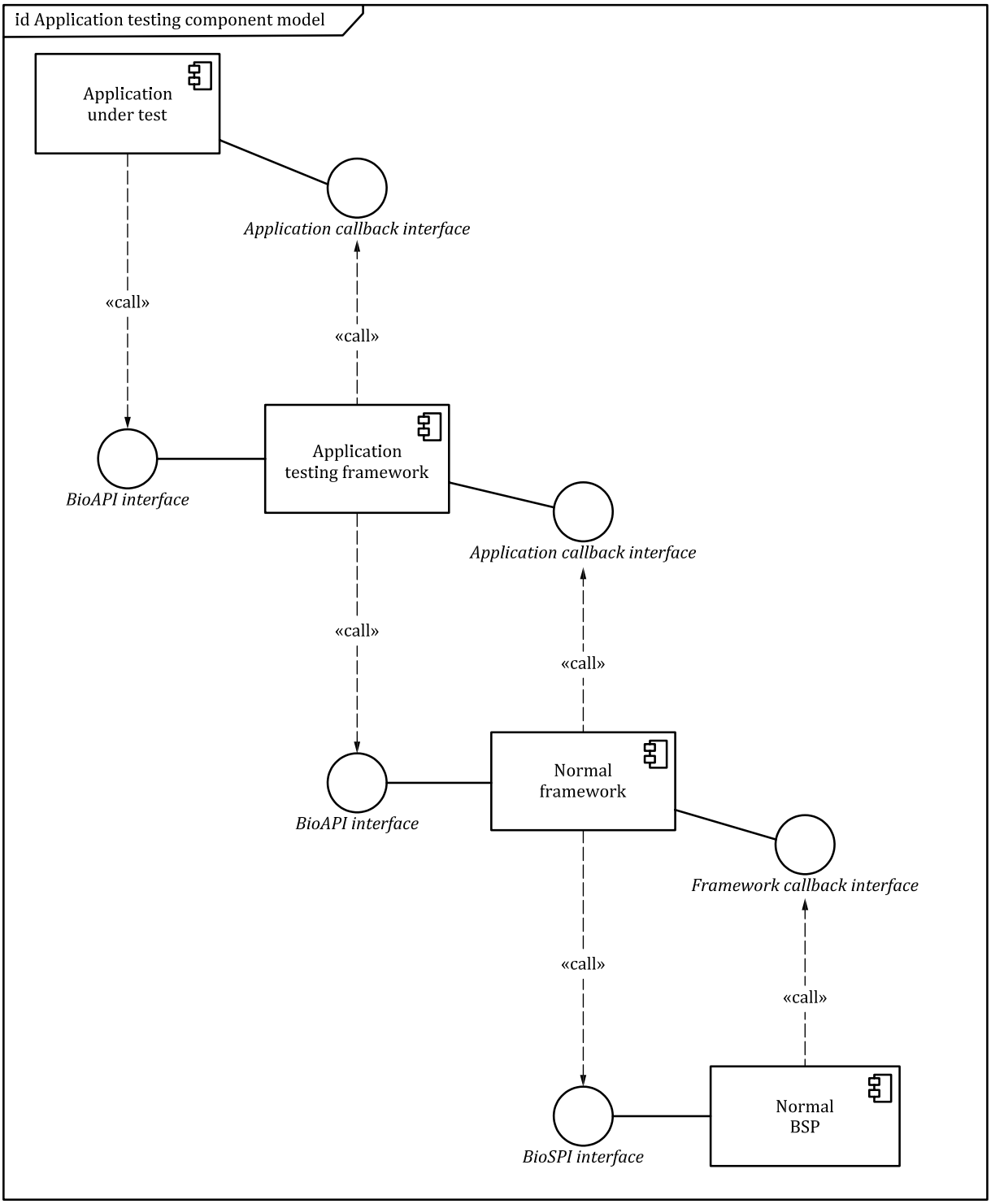

Figure 2 — Conformance testing model for BioAPI applications

© ISO/IEC 2017 – All rights reserved

Documents
Document Sharing Hub
Full document details: BS ISO/IEC 24709-1:2017

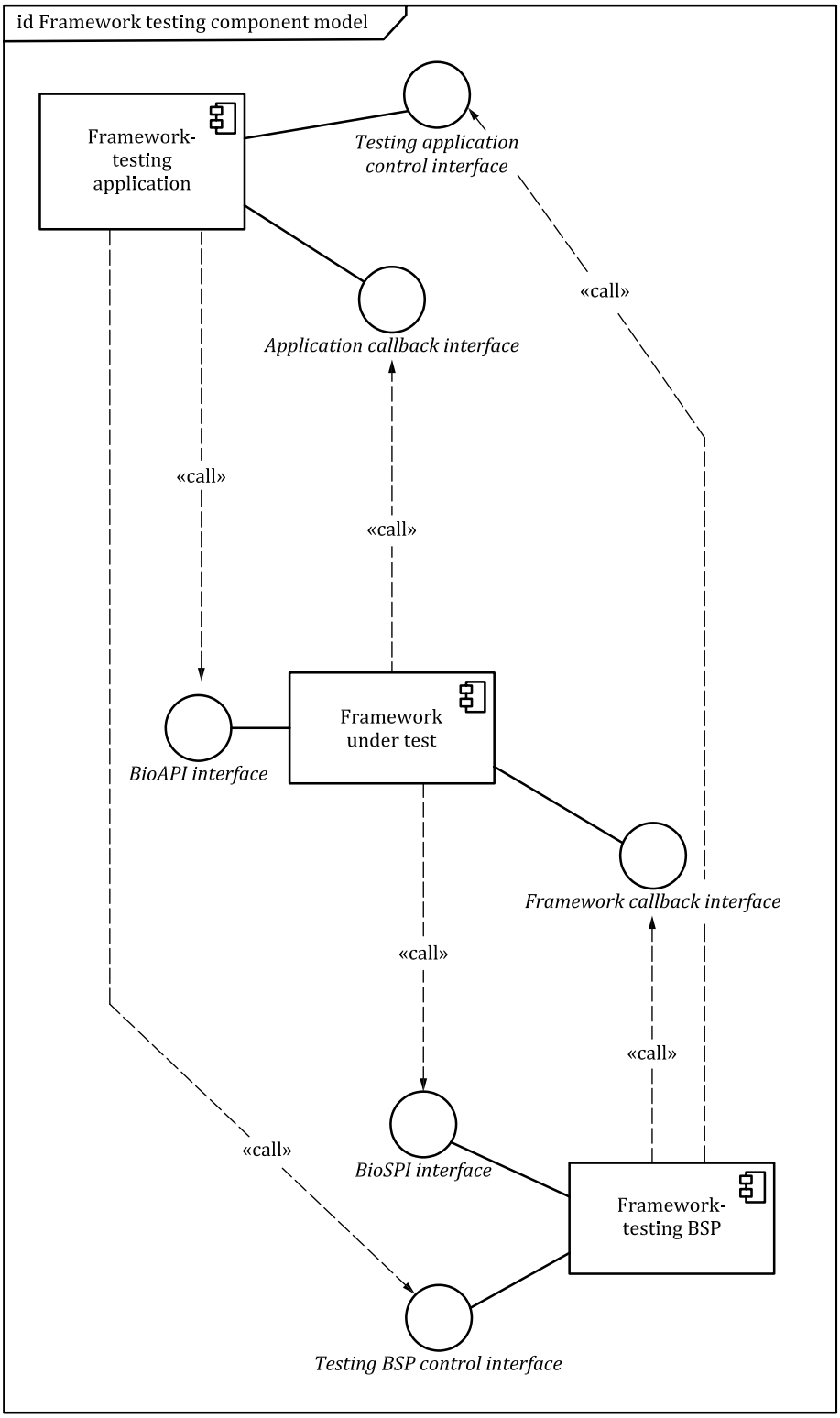

Figure 3 — Conformance testing model for BioAPI frameworks

© ISO/IEC 2017 – All rights reserved
Documents
Document Sharing Hub
Full document details: BS ISO/IEC 24709-1:2017

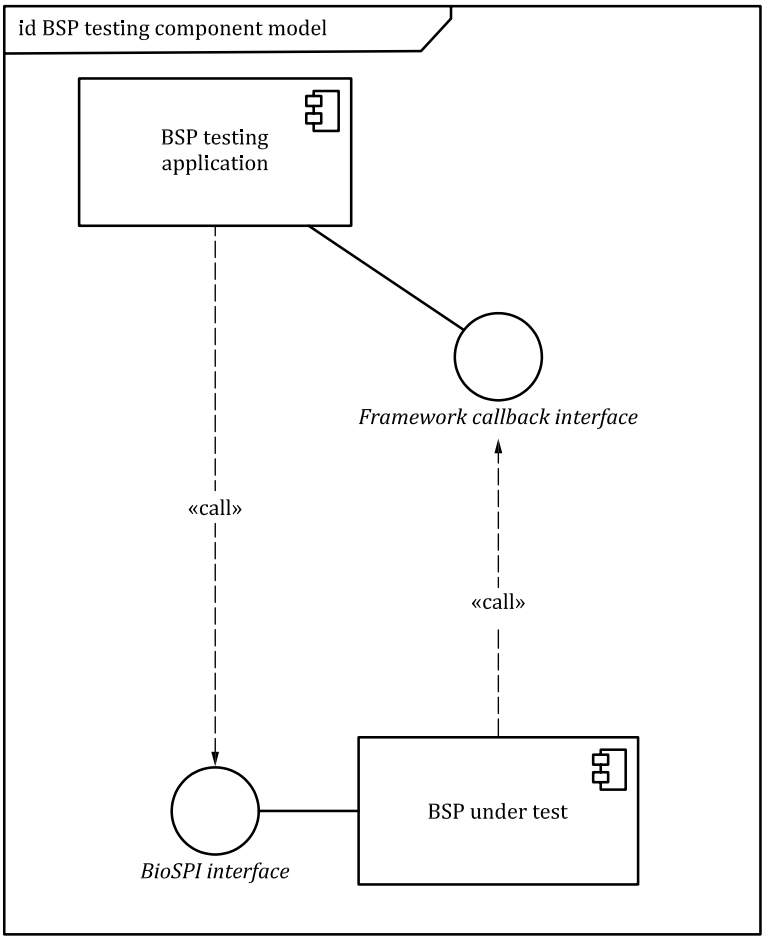

Figure 4 — Conformance testing model for BioAPI BSPs

6.3 Abstract test engine

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

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

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

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

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

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

© ISO/IEC 2017 – All rights reserved

Documents
Document Sharing Hub
Full document details: BS ISO/IEC 24709-1:2017

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

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

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

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

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

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

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

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

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

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

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

7 General properties of the assertion language

7.1 General

7.1.1 Clause 7 to Clause 10 specify a language whose purpose is to express assertions to be used in BioAPI conformance testing. The assertion language is an integral part of the BioAPI conformance testing methodology.

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

© ISO/IEC 2017 – All rights reserved
Documents
Document Sharing Hub
Full document details: BS ISO/IEC 24709-1:2017

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

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

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

EXAMPLE (non-normative)

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

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

EXAMPLE (non-normative)

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

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

© ISO/IEC 2017 – All rights reserved

Documents
Document Sharing Hub
Full document details: BS ISO/IEC 24709-1:2017

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

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

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

7.1.8 The means of assigning values to the input parameters of assertions is outside the scope of this document, but the values used in a test are documented in the standard test report (see Clause 12).

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

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

EXAMPLE (non-normative)

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

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


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


© ISO/IEC 2017 – All rights reserved
Documents
Document Sharing Hub
Full document details: BS ISO/IEC 24709-1:2017

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


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


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


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


</package>
```

7.2 Variables

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

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

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

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

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

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

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

a) when it is evaluated by a numeric operation (see 8.23 to 8.28), or

© ISO/IEC 2017 – All rights reserved

Documents
Document Sharing Hub
Full document details: BS ISO/IEC 24709-1:2017

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

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

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

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

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

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

7.3 Built-in variables

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

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

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

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

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 (e.g. to copy two or more variables to ordinary variables using `<set>`), the elements need to be placed in an activity with the attribute `atomic="true"`.

7.4 Representation of integers

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

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

© ISO/IEC 2017 – All rights reserved
Documents
Document Sharing Hub
Full document details: BS ISO/IEC 24709-1:2017

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

7.5 Representation of Booleans

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

7.5.2 This representation is canonical.

7.6 Representation of universally unique identifiers (UUIDs)

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

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

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

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

d) HYPHEN-MINUS ("-").

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

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

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

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

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

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

7.7 Representation of octet strings

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

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

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

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

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

© ISO/IEC 2017 – All rights reserved

Documents
Document Sharing Hub
Full document details: BS ISO/IEC 24709-1:2017

7.8 XML documents

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

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

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

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

8 Elements of the assertion language

8.1 Element `<package>`

8.1.1 Syntax

8.1.1.1 This element shall have the following attribute:

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

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

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

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

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

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

8.1.2 Semantics

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

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

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

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

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

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

© ISO/IEC 2017 – All rights reserved
Documents
Document Sharing Hub
Full document details: BS ISO/IEC 24709-1:2017

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

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

8.1.3 Example (non-normative)

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


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

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

8.2.1 Syntax

8.2.1.1 This element shall have the following attributes:

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

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

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

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

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

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

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

© ISO/IEC 2017 – All rights reserved

Documents
Document Sharing Hub
Full document details: BS ISO/IEC 24709-1:2017

8.2.2 Semantics

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

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

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

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

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

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

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

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

8.2.3 Example (non-normative)

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


© ISO/IEC 2017 – All rights reserved
Documents
Document Sharing Hub
Full document details: BS ISO/IEC 24709-1:2017

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

8.3.1 Syntax

8.3.1.1 This element shall have the following attribute:

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

8.3.1.2 The content of this element shall be empty.

8.3.2 Semantics

8.3.2.1 This element represents an input parameter of an assertion.

Assertions cannot have output parameters.

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

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

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

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

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

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

8.3.3 Example (non-normative)

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

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

8.4.1 Syntax

8.4.1.1 This element shall have the following attributes:

a) `activity` (required) — the value of this attriFbute shall be the name of an activity;

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

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

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

© ISO/IEC 2017 – All rights reserved

Documents
Document Sharing Hub
Full document details: BS ISO/IEC 24709-1:2017

8.4.2 Semantics

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

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

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

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

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

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

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

8.4.3 Example (non-normative)

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

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

8.5.1 Syntax

8.5.1.1 This element shall have the following attributes:

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

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

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

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

8.5.1.3 The content of this element shall be empty.

© ISO/IEC 2017 – All rights reserved
Documents
Document Sharing Hub
Full document details: BS ISO/IEC 24709-1:2017

8.5.2 Semantics

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

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

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

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

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

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

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

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

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

8.5.3 Example (non-normative)

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

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

8.6.1 Syntax

8.6.1.1 This element shall have the following attributes:

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

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

8.6.1.2 The content of this element shall be empty.

© ISO/IEC 2017 – All rights reserved

Documents
Document Sharing Hub
Full document details: BS ISO/IEC 24709-1:2017

8.6.2 Semantics

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

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

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

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

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

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

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

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

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

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

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

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

8.7.1 Syntax

8.7.1.1 This element shall have the following attribute:

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

© ISO/IEC 2017 – All rights reserved
Documents
Document Sharing Hub
Full document details: BS ISO/IEC 24709-1:2017

8.7.1.2 The content of this element shall be empty.

8.7.2 Semantics

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

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

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

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

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

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

8.8.1 Syntax

8.8.1.1 This element shall have the following attributes:

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

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

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

8.8.1.2 The content of this element shall be empty.

8.8.2 Semantics

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

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

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

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

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

© ISO/IEC 2017 – All rights reserved

Documents
Document Sharing Hub
Full document details: BS ISO/IEC 24709-1:2017

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

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

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

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

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

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

8.9.1 Syntax

8.9.1.1 This element shall have the following attributes:

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

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

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

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

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

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

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

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

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

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

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

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

© ISO/IEC 2017 – All rights reserved
Documents
Document Sharing Hub
Full document details: BS ISO/IEC 24709-1:2017