AS ISO/IEC 19784.1:2019
ISO/IEC 19784-1:2018

AS ISO/IEC 19784.1:2019

Australian STANDARD®

Information technology — Biometric application programming interface

Part 1: BioAPI specification

STANDARDS Australia

Documents
Document Sharing Hub

Full document details: AS ISO IEC 19784.1-2019

AS ISO/IEC 19784.1:2019

This Australian Standard® was prepared by IT-032, Biometrics and Identification. It was approved on behalf of the Council of Standards Australia on 26 November 2019.

This Standard was published on 13 December 2019.

The following are represented on Committee IT-032:

Australian Federal Police
Australian Information Industry Association
Australian Retailers Association
Centrelink
Defence Science & Technology Group
Department of Foreign Affairs and Trade (Australian Government)
Department of Immigration and Border Protection (Australian Government)
Engineers Australia
NSW Business Chamber
Photo Marketing Association

This Standard was issued in draft form for comment as DR2 AS ISO/IEC 19784.1:2019.

Keeping Standards up-to-date

Ensure you have the latest versions of our publications and keep up-to-date about Amendments, Rulings, Withdrawals, and new projects by visiting:
www.standards.org.au

ISBN 978 1 76072 660 7

Documents
Document Sharing Hub

Full document details: AS ISO IEC 19784.1-2019

AS ISO/IEC 19784.1:2019
ISO/IEC 19784-1:2018

Australian STANDARD®

Information technology — Biometric application programming interface

Part 1: BioAPI specification

Originated as AS ISO/IEC 19784.1—2007.
Second edition 2019.

COPYRIGHT

© ISO/IEC 2019 — All rights reserved
© Standards Australia Limited 2019

All rights are reserved. No part of this work may be reproduced or copied in any form or by any means, electronic or mechanical, including photocopying, without the written permission of the publisher, unless otherwise permitted under the Copyright Act 1968 (Cth).

Documents
Document Sharing Hub

Full document details: AS ISO IEC 19784.1-2019

Preface

This Standard was prepared by the Standards Australia Committee IT-032, Biometrics and Identification, to supersede AS ISO/IEC 19784.1—2007.

The objective of this Standard is to define the Application Programming Interface (API) and Service Provider Interface (SPI) for standard interfaces within a biometric system that support the provision of that biometric system using components from multiple vendors. It provides interworking between such components through adherence to this and to other International Standards.

For use in a system that does not include a BioAPI Framework (called a framework-free BioAPI system), only the SPI interface is applicable, with applications interfacing directly to that in a platform-specific manner.

This document does not define security requirements for biometric applications and biometric service providers.

The performance of biometric systems (particularly in relation to searches of a large population to provide the biometric identification capability) is not in the scope of this document. Trade-offs between interoperability and performance are not in the scope of this document.

This Standard is identical with, and has been reproduced from, ISO/IEC 19784-1:2018, *Information technology — Biometric application programming interface — Part 1: BioAPI specification*.

As this document has been reproduced from an International Standard, a full point substitutes for a comma when referring to a decimal marker.

Australian or Australian/New Zealand Standards that are identical adoptions of international normative references may be used interchangeably. Refer to the online catalogue for information on specific Standards.

The terms “normative” and “informative” are used in Standards to define the application of the appendices or annexes to which they apply. A “normative” appendix or annex is an integral part of a Standard, whereas an “informative” appendix or annex is only for information and guidance.

Documents
Document Sharing Hub

© Standards Australia Limited 2019

Full document details: AS ISO IEC 19784.1-2019

Contents

Preface ii
Foreword viii
Introduction ix
1 Scope 1
2 Normative references 2
3 Terms and definitions 2
4 Symbols and abbreviated terms 8
5 Conformance 8
6 The BioAPI architecture 9
6.1 The full BioAPI API/SPI Architectural Model 9
6.2 The framework-free BioAPI Architectural Model 10
6.3 The BioAPI BSP Architectural Model 10
6.4 The component registry 12
6.5 BSP and BFP Installation and De-installation 13
6.6 BSP Load and BioAPI Unit Attachment 14
6.7 Controlling BioAPI Units 15
6.8 BIR Structure and Handling 15
6.8.1 BIR Structure 15
6.8.2 BIR Data Handling 16
7 BioAPI types and macros 17
7.1 BioAPI 17
7.2 BioAPI_ACBio_PARAMETERS (BioAPI 2.2) 17
7.3 BioAPI_ASN1_BIR (BioAPI 2.2) 17
7.4 BioAPI_ASN1_ENCODED (BioAPI 2.2) 18
7.5 BioAPI_BFP_LIST_ELEMENT 18
7.6 BioAPI_BFP_SCHEMA 18
7.7 BioAPI_BIR 19
7.8 BioAPI_BIR_ARRAY_POPULATION 19
7.9 BioAPI_BIR_BIOMETRIC_DATA_FORMAT 20
7.10 BioAPI_BIR_BIOMETRIC_PRODUCT_ID 20
7.11 BioAPI_BIR_BIOMETRIC_TYPE (BioAPI 2.0) 20
7.12 BioAPI_BIR_BIOMETRIC_TYPE (BioAPI 2.1) 21
7.13 BioAPI_BIR_DATA_TYPE 22
7.14 BioAPI_BIR_HANDLE 23
7.15 BioAPI_BIR_HEADER 23
7.16 BioAPI_BIR_PURPOSE 24
7.17 BioAPI_BIR_SECURITY_BLOCK_FORMAT 25
7.18 BioAPI_BIR_SUBTYPE (BioAPI 2.0) 25
7.19 BioAPI_BIR_SUBTYPE (BioAPI 2.1) 26
7.20 BioAPI_BIR_SUBTYPE_MASK (BioAPI 2.1) 27
7.21 BioAPI_BOOL 28
7.22 BioAPI_BSP_SCHEMA (BioAPI 2.0) 28
7.23 BioAPI_BSP_SCHEMA (BioAPI 2.1) 30
7.24 BioAPI_CANDIDATE 32
7.25 BioAPI_CATEGORY 32
7.26 BioAPI_DATA 32
7.27 BioAPI_DATE 32
7.28 BioAPI_DB_ACCESS_TYPE 33
7.29 BioAPI_DB_MARKER_HANDLE 33
7.30 BioAPI_DB_HANDLE 33
7.31 BioAPI_DBBIR_ID 33

Documents
Document Sharing Hub
Full document details: AS ISO IEC 19784.1-2019

7.32 BioAPI_DTG ... 34
7.33 BioAPI_ENCRYPTION_ALG (BioAPI 2.2) ... 34
7.34 BioAPI_ENCRYPTION_INFO (BioAPI 2.2) ... 34
7.35 BioAPI_ERROR_INFO (BioAPI 2.1) ... 34
7.36 BioAPI_EVENT ... 35
7.37 BioAPI_EVENT_MASK ... 35
7.38 BioAPI_EventHandler ... 35
7.39 BioAPI_FMR ... 36
7.40 BioAPI_FRAMEWORK_SCHEMA ... 36
7.41 BioAPI_GUI_BITMAP (BioAPI 2.0) ... 37
7.42 BioAPI_GUI_BITMAP (BioAPI 2.1) ... 37
7.43 BioAPI_GUI_BITMAP_ARRAY (BioAPI 2.1) ... 38
7.44 BioAPI_GUI_ENROLL_TYPE (BioAPI 2.1) ... 38
7.45 BioAPI_GUI_EVENT_SUBSCRIPTION (BioAPI 2.1) ... 39
7.46 BioAPI_GUI_MESSAGE (BioAPI 2.0) ... 39
7.47 BioAPI_GUI_MOMENT (BioAPI 2.1) ... 40
7.48 BioAPI_GUI_OPERATION (BioAPI 2.1) ... 41
7.49 BioAPI_GUI_PROGRESS (BioAPI 2.0) ... 41
7.50 BioAPI_GUI_PROGRESS (BioAPI 2.1) ... 42
7.51 BioAPI_GUI_RESPONSE (BioAPI 2.0) ... 42
7.52 BioAPI_GUI_RESPONSE (BioAPI 2.1) ... 42
7.53 BioAPI_GUI_STATE (BioAPI 2.0) ... 44
7.54 BioAPI_GUI_STATE_CALLBACK (BioAPI 2.0) ... 44
7.55 BioAPI_GUI_STREAMING_CALLBACK (BioAPI 2.0) ... 45
7.56 BioAPI_GUI_SUBOPERATION (BioAPI 2.1) ... 46
7.57 BioAPI_HANDLE ... 47
7.58 BioAPI_HASH_ALG (BioAPI 2.2) ... 47
7.59 BioAPI_IDENTIFY_POPULATION ... 47
7.60 BioAPI_IDENTIFY_POPULATION_TYPE ... 47
7.61 BioAPI_INDICATOR_STATUS ... 48
7.62 BioAPI_INPUT_BIR ... 48
7.63 BioAPI_INPUT_BIR_FORM ... 48
7.64 BioAPI_INSTALL_ACTION ... 48
7.65 BioAPI_INSTALL_ERROR ... 48
7.66 BioAPI_KEY_INFO (BioAPI 2.2) ... 49
7.67 BioAPI_KEY_TRANSPORT (BioAPI 2.2) ... 49
7.68 BioAPI_MAC_ALG (BioAPI 2.2) ... 49
7.69 BioAPI_MAC_INFO (BioAPI 2.2) ... 49
7.70 BioAPI_OPERATIONS_MASK ... 50
7.71 BioAPI_OPTIONS_MASK ... 50
7.72 BioAPI_POWER_MODE ... 52
7.73 BioAPI_QUALITY ... 52
7.74 BioAPI_RETURN ... 53
7.75 BioAPI_SECURITY_OPTIONS_MASK (BioAPI 2.2) ... 53
7.76 BioAPI_SECURITY_PROFILE (BioAPI 2.2) ... 53
7.77 BioAPI_DIGITAL_SIGNATURE_ALG (BioAPI 2.2) ... 54
7.78 BioAPI_STRING ... 55
7.79 BioAPI_TIME ... 55
7.80 BioAPI_UNIT_ID ... 55
7.81 BioAPI_UNIT_LIST_ELEMENT ... 55
7.82 BioAPI_UNIT_SCHEMA ... 55
7.83 BioAPI_UNIT_SCHEMA (BioAPI 2.2) ... 57
7.84 BioAPI_UUID ... 58
7.85 BioAPI_VERSION ... 58
7.86 GUI Events ... 58
7.86.1 BioAPI_GUI_SELECT_EVENT_HANDLER (BioAPI 2.1) ... 58
7.86.2 BioAPI_GUI_STATE_EVENT_HANDLER (BioAPI 2.1) ... 61
7.86.3 BioAPI_GUI_PROGRESS_EVENT_HANDLER (BioAPI 2.1) ... 63

Documents
Document Sharing Hub
Full document details: AS ISO IEC 19784.1-2019

8 BioAPI functions **65**
8.1 Component Management Functions 65
8.1.1 BioAPI_Init 65
8.1.2 BioAPI_Terminate 65
8.1.3 BioAPI_GetFrameworkInfo 66
8.1.4 BioAPI_EnumBSPs 67
8.1.5 BioAPI_BSPLoad 67
8.1.6 BioAPI_BSPUnload 69
8.1.7 BioAPI_BSPAttach 70
8.1.8 BioAPI_BSPAttachSecure (BioAPI 2.2) 71
8.1.9 BioAPI_BSPDetach 73
8.1.10 BioAPI_QueryUnits 73
8.1.11 BioAPI_EnumBFPs 75
8.1.12 BioAPI_QueryBFPs 75
8.1.13 BioAPI_ControlUnit 76
8.1.14 BioAPI_Control (BioAPI 2.1) 77
8.1.15 BioAPI_Transform (BioAPI 2.1) 78
8.1.16 BioAPI_LinkToEndpoint (BioAPI 2.1) 79
8.1.17 BioAPI_UnlinkFromEndpoint (BioAPI 2.1) 80
8.1.18 BioAPI_EnumFrameworks (BioAPI 2.1) 80
8.2 Data Handle Operations 81
8.2.1 BioAPI_FreeBIRHandle 81
8.2.2 BioAPI_GetBIRFromHandle 82
8.2.3 BioAPI_GetHeaderFromHandle 83
8.3 Callback and Event Operations 83
8.3.1 BioAPI_EnableEvents 83
8.3.2 BioAPI_SetGUICallbacks (BioAPI 2.0) 84
8.3.3 BioAPI_NotifyGUIProgressEvent (BioAPI 2.1) 85
8.3.4 BioAPI_NotifyGUISelectEvent (BioAPI 2.1) 86
8.3.5 BioAPI_NotifyGUIStateEvent (BioAPI 2.1) 87
8.3.6 BioAPI_QueryGUIEventSubscriptions (BioAPI 2.1) 88
8.3.7 BioAPI_RedirectGUIEvents (BioAPI 2.1) 89
8.3.8 BioAPI_SubscribeToGUIEvents (BioAPI 2.1) 91
8.3.9 BioAPI_UnredirectGUIEvents (BioAPI 2.1) 93
8.3.10 BioAPI_UnsubscribeFromGUIEvents (BioAPI 2.1) 94
8.3.11 BioAPI_EnableEventNotifications (BioAPI 2.1) 95
8.4 Biometric Operations 95
8.4.1 BioAPI_Capture 95
8.4.2 BioAPI_CreateTemplate 97
8.4.3 BioAPI_Process 98
8.4.4 BioAPI_ProcessWithAuxBIR (BioAPI2.0 and BioAPI2.1) 100
8.4.5 BioAPI_ProcessUsingAuxBIRs (BioAPI 2.2) 101
8.4.6 BioAPI_VerifyMatch 102
8.4.7 BioAPI_VerifyMatchUsingAuxBIRs (BioAPI 2.2) 104
8.4.8 BioAPI_IdentifyMatch 106
8.4.9 BioAPI_Decide (BioAPI 2.2) 108
8.4.10 BioAPI_Fuse (BioAPI 2.2) 110
8.4.11 BioAPI_Enroll 111
8.4.12 BioAPI_Verify 113
8.4.13 BioAPI_Identify 115
8.4.14 BioAPI_Import 118
8.4.15 BioAPI_Export (BioAPI 2.2) 119
8.4.16 BioAPI_PresetIdentifyPopulation 120
8.5 Database Operations 121
8.5.1 BioAPI_DbOpen 121
8.5.2 BioAPI_DbClose 122
8.5.3 BioAPI_DbCreate 122
8.5.4 BioAPI_DbDelete 123

Documents
Document Sharing Hub
Full document details: AS ISO IEC 19784.1-2019

8.5.5 BioAPI_DbSetMarker 124
8.5.6 BioAPI_DbFreeMarker 124
8.5.7 BioAPI_DbStoreBIR 125
8.5.8 BioAPI_DbGetBIR 126
8.5.9 BioAPI_DbGetNextBIR 127
8.5.10 BioAPI_DbDeleteBIR 128
8.6 BioAPI Unit operations 128
8.6.1 BioAPI_SetPowerMode 128
8.6.2 BioAPI_SetIndicatorStatus 129
8.6.3 BioAPI_GetIndicatorStatus 130
8.6.4 BioAPI_CalibrateSensor 130
8.7 Utility Functions 131
8.7.1 BioAPI_Cancel 131
8.7.2 BioAPI_Free 131

9 BioAPI Service Provider Interface 132
9.1 Summary 132
9.2 Type Definitions for Biometric Service Providers 132
9.2.1 BioSPI_EventHandler 132
9.2.2 BioSPI_BFP_ENUMERATION_HANDLER 133
9.2.3 BioSPI_MEMORY_FREE_HANDLER 134
9.2.4 BioSPI_GUI_PROGRESS_EVENT_HANDLER (BioAPI 2.1) 135
9.2.5 BioSPI_GUI_SELECT_EVENT_HANDLER (BioAPI 2.1) 136
9.2.6 BioSPI_GUI_STATE_EVENT_HANDLER (BioAPI 2.1) 137
9.3 Biometric Service Provider Operations 137
9.3.1 SPI Component Management Operations 137
9.3.2 SPI Data Handle Operations 143
9.3.3 SPI Callback and Event Operations 143
9.3.4 SPI Biometric Operations 145
9.3.5 SPI Database Operations 148
9.3.6 SPI BioAPI Unit operations 150
9.3.7 SPI Utility Functions 151

10 Component registry interface 151
10.1 BioAPI Registry Schema 151
10.1.1 Framework Schema 152
10.1.2 BSP Schema 152
10.1.3 BFP Schema 153
10.2 Component registry functions 154
10.2.1 BioAPI_Util_InstallBSP 154
10.2.2 BioAPI_Util_InstallBFP 155
10.2.3 BioAPI_RegisterBSP (BioAPI 2.1) 156
10.2.4 BioAPI_UnregisterBSP (BioAPI 2.1) 156
10.2.5 ioAPI_RegisterBFP (BioAPI 2.1) 157
10.2.6 BioAPI_UnregisterBFP (BioAPI 2.1) 158
10.2.7 BioAPI_GetLastErrorInfo (BioAPI 2.1) 159

11 BioAPI error handling 159
11.1 Error Values and Error Codes Scheme 159
11.2 Error Codes and Error Value Enumeration 160
11.2.1 BioAPI Error Value Constants 160
11.2.2 Implementation-Specific Error Codes 160
11.2.3 General Error Codes 160
11.2.4 Component Management Error Codes 162
11.2.5 Database Error Values 162
11.2.6 Location Error Values 163
11.2.7 Quality Error Codes 164
11.2.8 Security Error Codes (BioAPI 2.2) 165

Annex A (normative) **Conformance 167**

AVA Documents
Document Sharing Hub

Full document details: AS ISO IEC 19784.1-2019

Annex B (normative) **CBEFF Patron Format Specification: BioAPI patron format** **184**
Annex C (informative) **Specification overview** **189**
Annex D (informative) **Calling sequence examples and sample code** **213**
Annex E (normative) **ASN.1 specification of BioAPI_BIR (BioAPI2.2)** **231**
Bibliography **233**

Documents
Document Sharing Hub

Full document details: AS ISO IEC 19784.1-2019

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 ISO/IEC JTC 1, *Information technology*, SC 37, *Biometrics*.

This second edition cancels and replaces the first edition (ISO/IEC 19784-1:2006). It also incorporates the Amendments ISO/IEC 19784-1:2006/Amd 1:2007, ISO/IEC 19784-1:2006/Amd 2:2009 and ISO/IEC 19784-1:2006/Amd 3:2010.

A list of all the parts in the ISO 19784 series, can be found on the ISO website.

© Standards Australia Limited 2019

Documents
Document Sharing Hub

Full document details: AS ISO IEC 19784.1-2019

Introduction

This document provides a high-level generic biometric authentication model suited to most forms of biometric technology. An architectural model is described which enables components of a biometric system to be provided by different vendors, and to interwork through fully-defined Application Programming Interfaces (APIs).

A key feature of the architecture is the BioAPI Framework, which supports calls by one or more application components (provided by different vendors, and potentially running concurrently) using the BioAPI API specification. The BioAPI Framework provides this support by invoking (through a Service Provider Interface, SPI) one or more Biometric Service Provider (BSP) components (provided by different vendors, and potentially running concurrently) which can be dynamically loaded and invoked as required by an application component.

However, this document can also be applied where a system is to be built from conforming BSP components (without a BioAPI Framework module), using platform-specific system-integration mechanisms - see Clause 6.

This document specifies the behaviour of the BioAPI Framework when applications and BSPs are in the same system. Other interworking standards (see 4.29) specify modifications of that behaviour that enable both BSPs and Graphical User Interfaces to be remote from the system containing an application.

NOTE 1 ISO/IEC 24708 BioAPI Interworking Protocol (BIP)[6] is an example of an interworking standard.

At the lowest level there is hardware or software that performs biometric functions such as capture, matching, or archiving. These parts of the architecture are called BioAPI Units, and can be integral to a BSP or can be supplied as part of a separate BioAPI Function Provider (BFP) component.

Interactions (through the BioAPI Framework) can occur between BSPs from different vendors provided data structures used to record information from the BioAPI Units they access conform to other International Standards, and in particular to ISO/IEC 19794[5].

The final component of the BioAPI architecture is the recognition that a BSP can provide its biometric services either:

a) by the use of BioAPI Units that are integral to (that is, directly managed by) the BSP, or

b) by invoking, through the BioAPI Function Provider Interface (FPI), one or more BFP components (provided by different vendors) that manage BioAPI Units that are integral to the BFP.

NOTE 2 A BioAPI Unit may consist of software only, or a combination of software and hardware (e.g., a biometric sensor, archive, or algorithm).

For each type of BioAPI Unit supported by a BSP (or BFP) there may be one or more BioAPI Units of that type which can be dynamically inserted and removed from the system. Insertion and removal generates events that can be signalled (through the BSP and the BioAPI Framework) to an application.

The BioAPI specification covers the basic biometric functions of Enrollment, Verification, and Identification (see Annex C), and includes a database interface to allow an application to manage the storage of biometric records through an archive BioAPI Unit managed by a BSP or BFP. This provides for optimum performance (e.g., when performing the biometric Identification function within a large population) of the archiving and biometric search processes.

The interface to the application provides primitives that allow it to manage the capture of biometric samples from a biometric sensor by accessing the corresponding BioAPI Unit, and the use of those biometric samples for Enrollment (storage in an application-controlled or BSP-controlled BIR database), and subsequent Verification or Identification against those stored records.

This document also specifies the content of a biometric component registry (information about the biometric components that have been installed on the biometric system). It also provides a component registry interface for the management and inspection of that registry.

© Standards Australia Limited 2019

Documents Document Sharing Hub

Full document details: AS ISO IEC 19784.1-2019

This documnet uses the C programming language (see ISO/IEC 9899) to specify the data structures and function calls that form the BioAPI interfaces.

Clause 6 describes the BioAPI architectural model and its components, and the interfaces that are specified between these components.

Clause 7 defines the data structures used in the BioAPI.

Clause 8 defines the function calls initiated by an application and supported by a conforming BioAPI Framework that are either handled internally by the BioAPI Framework (for example enumeration of installed BioAPI components) or mapped to a function provided by a BSP.

Clause 9 defines the function calls supported by a conforming BSP (and invoked by the BioAPI Framework in response to a call from a biometric application).

Clause 10 specifies the form of the biometric component registry and the component registry interface.

Clause 11 defines the handling of events and error returns.

Annex A is normative, and specifies details of conformance requirements and proformas that can be used by the vendor of a BioAPI Biometric Application, Framework, or BSP component to identify those functions and biometric record formats that shall be supported.

NOTE 3 A separate IS, ISO/IEC 24709, addresses conformance testing for this BioAPI specification[9].

Annex B is normative, and specifies the BioAPI Biometric Information Record (BIR) as a CBEFF Patron Format in accordance with ISO/IEC 19785-1. It provides a description of the biometric record specified in this part of ISO/IEC 19784, together with the (platform-independent) bit-pattern representation of such a record for storage and transfer.

Annex C is informative, and provides a general tutorial on a number of aspects of the BioAPI specification.

Annex D is informative, and provides example code to illustrate calling sequences and to provide implementation guidance.

Annex E provides an ASN.1 specification of the BioAPI Biometric Information Record (BIR).

This revision is a merged document of ISO/IEC 19784-1: 2006, ISO/IEC 19784-1: 2006/Amd 1: 2007, ISO/IEC 19784-1: 2006/Amd 2:2009, and ISO/IEC 19784-1: 2006/Amd 3 2010. But in this document, the defects in ISO/IEC 19784-1: 2006/Amd 3 are corrected. BioAPI 2.0 means specification from ISO/IEC 19784-1: 2006/Amd 2:2009, BioAPI 2.1 from ISO/IEC 19784-1: 2006/Amd 1: 2007, and BioAPI 2.2 from ISO/IEC 19784-1: 2006/Amd 3 2010.

© Standards Australia Limited 2019

Documents
Document Sharing Hub

Full document details: AS ISO IEC 19784.1-2019

Australian Standard®

Information technology — Biometric application programming interface

Part 1: BioAPI specification

1 Scope

This document defines the Application Programming Interface (API) and Service Provider Interface (SPI) for standard interfaces within a biometric system that support the provision of that biometric system using components from multiple vendors. It provides interworking between such components through adherence to this and to other International Standards.

For use in a system that does not include a BioAPI Framework (called a framework-free BioAPI system), only the SPI interface is applicable, with applications interfacing directly to that in a platform-specific manner.

NOTE 1 Many clauses and/or sub-clauses of this document are not applicable for implementation of a framework-free BioAPI system. These are identified at the head of the clause of sub-clause.

The BioAPI specification is applicable to a broad range of biometric technology types. It is also applicable to a wide variety of biometrically enabled applications, from personal devices, through network security applications, to large complex identification systems.

This document supports an architecture in which a BioAPI Framework supports multiple simultaneous biometric applications (provided by different vendors), using multiple dynamically installed and loaded (or unloaded) Biometric Service Provider (BSP) components and BioAPI Units (provided by other different vendors), possibly using one of an alternative set of BioAPI Function Provider (BFP) components (provided by other vendors) or by direct management of BioAPI Units.

NOTE 2 Where BioAPI Units are provided by a different vendor fom a BSP, a standardised BioAPI Function Provider Interface (FPI) may be needed. This is outside the scope of this document, but is specified by later parts for the different categories of FPI.

NOTE 3 Where a BioAPI Framework is not used in a system, the ability to support multiple applications and multiple BSPs is platform-dependent and depends on the nature of the system-integration techniques employed.

This document is not required (and should normally not be referenced) when a complete biometric system is being procured from a single vendor, particularly if the addition or interchange of biometric hardware, services, or applications is not a feature of that biometric system. (Such systems are sometimes referred to as "embedded systems".) Standardisation of such systems is not in the scope of this document.

This document does not define security requirements for biometric applications and biometric service providers.

NOTE 4 ISO 19092 provides guidelines on security aspects of biometric systems[3].

The performance of biometric systems (particularly in relation to searches of a large population to provide the biometric identification capability) is not in the scope of this document. Trade-offs between interoperability and performance are not in the scope of this document.

This document specifies a version of the BioAPI specification that is defined to have a version number described as Major 2, Minor 0, or version 2.0. It also specifies a version number described as Major 2, Minor 1, or version 2.1 that provides an enhanced Graphical User Interface. It also specifies a version number described as Major 2, Minor 2, or version 2.2 that provides features supporting fusion and security. Some clauses and sub-clauses apply only to one of these versions, some to two or more. This is identified at the head of the relevant clauses and sub-clauses.

NOTE 5 Earlier versions of the BioAPI specification were not International Standards.

Documents
Document Sharing Hub
© Standards Australia Limited 2019

Full document details: AS ISO IEC 19784.1-2019

NOTE 6 The differences between the requirements of the 2.0 specification and the 2.1 specification for framework-free operation relate only to the biometric type values and encodings.

Conformance requirements are specified in Clause 5.

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/IEC 10646:2017, *Information technology — Universal Coded Character Set (UCS)*

ISO/IEC 19785-1, *Information technology — Common Biometric Exchange Formats Framework — Part 1: Data element specification*

ISO/IEC 19785-4:2010, *Information technology — Common Biometric Exchange Formats Framework — Part 4: Security block format specifications*

3 Terms and definitions

For the purposes of this document, the following terms and definitions 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

NOTE 1 Function names and data element names are not included here, but are defined within the body of this document.

NOTE 2 Many of these terms and definitions are not relevant for the use of this document for framework-free BioAPI systems.

4.1
ACBio instance
report generated by a biometric processing unit (BPU) compliant to ISO/IEC 24761 to show the validity of the result of one or more subprocesses executed in the BPU

[SOURCE: ISO/IEC 24761]

4.2
adaptation
template adaptation
use of a BIR produced from a newly captured and verified biometric sample to automatically update or refresh a reference template

Note 1 to entry: This procedure is used to minimize the effects of template aging.

4.3
attach session
temporary association between an application, a single BSP, and a set of BioAPI Units that are managed either directly or indirectly by that BSP

4.4
authentication context for biometrics
ACBio
international standard that specifies the form and encoding of ACBio instances

[SOURCE: ISO/IEC 24761]

© Standards Australia Limited 2019

Documents
Document Sharing Hub

Full document details: AS ISO IEC 19784.1-2019

4.5
BioAPI component
component of the BioAPI architecture with a defined interface that can be supplied by a separate vendor and which is subject to conformance testing

Note 1 to entry: BioAPI components include BioAPI applications, the BioAPI Framework, BSPs, and BFPs.

4.6
BioAPI Function Provider
BFP
component that manages one or more BioAPI Units of a specific category

Note 1 to entry: Interfaces to BioAPI Function Providers are standardized in subsequent parts of the ISO/IEC 19784 series.

Note 2 to entry: BFPs are categorized according to the categories of BioAPI Units that they manage (see 6.3.4).

4.7
BioAPI Unit
abstraction of a hardware or software resource that is directly managed by a BSP or BFP

Note 1 to entry: Note1 to entry: BioAPI Units are categorized (see 6.3.2) and include sensor units, archive units, matching algorithm units and processing algorithm units.

4.8
biographic data (BioAPI 2.2)
non-biometric data that potentially affects a biometric operation

4.9
biographic BIR (BioAPI 2.2)
non-biometric BIR that potentially affects a biometric operation

4.10
biometric
pertaining to the field of biometrics

4.11
Biometric Data Block
BDB
block of data with a defined format that contains one or more biometric samples or biometric templates

Note 1 to entry: Note1 to entry: ISO/IEC 19784 does not support BDB formats that are not an integral multiple of eight bits.

Note 2 to entry: There is no requirement that a BDB format be self-delimiting.

Note 3 to entry: Each part of the ISO/IEC 19794 series standardises one or more BDB formats. Vendor specific formats can also be specified and identified.

Note 4 to entry: Within BioAPI, the BDB is “opaque” to the application and is therefore sometimes referred to as an opaque biometric data block.

4.12
Biometric Information Record
BIR
data structure containing one or more BDBs, together with information identifying the BDB formats, and possibly further information such as whether a BDB is signed or encrypted

Note 1 to entry: See ISO/IEC 19785-1.

Documents
Document Sharing Hub
© Standards Australia Limited 2019

Full document details: AS ISO IEC 19784.1-2019

Note 2 to entry: This document defines a BIR format (see 7.7) that supports only a single BDB. ISO/IEC 19785-1 defines a more general BIR format that supports multiple BDBs within the BIR, and the above definition is used in common by the two International Standards. When the term BIR is used in this document, it normally refers to the specific BIR format defined by BioAPI (see Annex B), not to an arbitrary BIR. The term BioAPI BIR is used where clarity is needed.

4.12.1
reference BIR
BIR whose BDB(s) contains one or more biometric templates

4.12.2
sample BIR
BIR whose BDB(s) contains only biometric samples that are not templates

4.13
biometric processing unit
BPU
entity that executes one or more subprocesses that perform a biometric verification at a uniform level of security

Note 1 to entry: See ISO/IEC 24761.

Note 2 to entry: A sensor, a smart card, and a comparison device are examples of BPUs.

4.14
biometric sample
information obtained from a biometric sensor, either directly or after further processing

Note 1 to entry: See also raw biometric sample, intermediate biometric sample, and processed biometric sample.

4.14.1
biometric template
biometric sample or combination of biometric samples that is suitable for storage as a reference for future comparison

4.14.2
intermediate biometric sample
biometric sample obtained by processing a raw biometric sample, intended for further processing

4.14.3
processed biometric sample
biometric sample suitable for comparison

4.14.4
raw biometric sample
biometric sample obtained directly from a biometric sensor

Note 1 to entry: The formats for raw biometric samples are not currently standardised, and depend on the nature of the biometric device and the vendor of that device. They may in the future be standardised as part of the standardisation of specific biometric devices.

4.14.5
reference template
biometric template that has been stored

4.15
biometric sensor
biometric hardware used to capture raw biometric samples from a subject

Note 1 to entry: The term 'biometric device' is used interchangeably with this term.

© Standards Australia Limited 2019

Documents
Document Sharing Hub

Full document details: AS ISO IEC 19784.1-2019

4.16
Biometric Service Provider
BSP
component that provides biometric services to an application through a defined interface by managing one or more BioAPI Units directly, or through interfaces to BioAPI Function Providers

4.17
biometrics
automated recognition of individuals based on their behavioural and biological characteristics

4.18
BPU IO Index
integer assigned to each biometric data stream between BPUs by the subject, such as software, which utilizes the function of the BPU so that the validator can reconstruct the data flow among BPUs

Note 1 to entry: See ISO/IEC 24761.

4.19
callback
mechanism by which a component that exposes an API invokes a function within a component that uses that API, where the address of that function has been previously passed as an input parameter of an API function call

Note 1 to entry: This mechanism enables a BioAPI component to communicate with another BioAPI component other than by invoking an API function, usually in response to an event or interrupt.

4.20
component registry
information maintained by the BioAPI Framework concerning the BioAPI components that are available on a biometric system

4.21
decision BIR (BioAPI 2.2)
BIR which contains decision in the BDB

4.22
encrypt
encryption
(reversible) transformation of data by a cryptographic algorithm to produce ciphertext, i.e. hide the information content (protect the confidentiality) of the data

Note 1 to entry: Encryption algorithms consist of two processes: encryption (or encipherment) which transforms plaintext into ciphertext, and decryption (or decipherment) which transforms ciphertext to plaintext.

Note 2 to entry: Encryption may be used for either security or privacy reasons.

4.23
enrollment
process of collecting one or more biometric samples from an individual, and the subsequent construction of a biometric reference template which can then be used to verify or determine the individual's identity

Note 1 to entry: The reference template would normally be stored by a biometric application, a BSP supporting an archive BioAPI Unit, or both.

4.24
False Match Rate
FMR
measure of the probability that a biometric matching process will incorrectly identify an individual or will fail to reject an impostor

Note 1 to entry: Within BioAPI, FMR is used as a means of specifying scores and thresholds (see C.4).

Documents
Document Sharing Hub
© Standards Australia Limited 2019

Full document details: AS ISO IEC 19784.1-2019

Note 2 to entry: Historically, False Acceptance Rate (FAR) has also been used with a similar definition, but FMR is the preferred term in International Standards. Similarly for False Rejection Rate (FRR), as opposed to the preferred False Non-Match Rate (FNMR).

4.25
handle
parameter returned by a BioAPI function (A say) that can be used by the BioAPI application in a subsequent function call to identify a BioAPI component or data element within the component A

Note 1 to entry: Types of handles include:

— **BIR_Handle**, generated by a BSP to select or access a BIR within that BSP;

— **BSP Attach Session Handle**, for an attach session;

— **DB_Handle**, generated by a BSP to select or access a BIR database controlled by that BSP.

4.26
identify
identification
one-to-many process of comparing a submitted biometric sample against a reference population to determine whether the submitted biometric sample matches any of the reference templates in that reference population in order to determine the identity of the enrollee whose template was matched

Note 1 to entry: This is often called an "identification match" or "identifymatch".

4.27
match
matching
one-to-one process of comparing a submitted biometric sample against a single biometric reference template and scoring the level of similarity

Note 1 to entry: An accept or reject decision would then normally be based upon whether this score exceeds a given threshold.

Note 2 to entry: Matching algorithms and their effect on False Match Rate and False Non-Match Rate scores are currently not standardised.

Note 3 to entry: See also identify (4.22) and verify (4.33).

4.28
payload
data, provided at the time of enrolment and associated with a reference template, which can be released upon a successful biometric verification

Note 1 to entry: Examples of payloads include user names, accounts, passwords, cryptographic keys, or digital certificates (see C.5).

4.29
score
score data
scoring
value indicating the degree of similarity or correlation between a biometric sample and a biometric reference template

4.30
score BIR (BioAPI 2.2)
BIR which contains score in the BDB

4.31
secure BioAPI (BioAPI 2.2)
BioAPI API and SPI interfaces that include the security features defined for version 2.2 of BioAPI

© Standards Australia Limited 2019

Documents
Document Sharing Hub

Full document details: AS ISO IEC 19784.1-2019

4.32
security block
block of data with a defined format that contains security information (for example, related to encryption or integrity) related to a BIR

Note 1 to entry: See ISO/IEC 19785-1.

4.33
self-contained device
combination device which includes a biometric sensor and all or part of the BSP functionality

Note 1 to entry: A self-contained device may include the ability to not only capture a biometric, but also to process, match, and/or store it. This functionality is typically implemented in hardware/firmware.

4.34
signature
digital signature
data appended to, or a cryptographic transformation of, a data unit that allows the recipient of the data unit to prove the origin and integrity of the data unit and protect against forgery, e.g. by the recipient

Note 1 to entry: Digital signatures may be used for purposes of authentication, data integrity, and non-repudiation.

4.35
threshold
predefined value which establishes the degree of similarity or correlation (that is, a score) necessary for a biometric sample to be deemed a match with a biometric reference template

4.36
universally unique identifier
UUID
128-bit value generated in accordance with ISO/IEC 9834-8 and providing unique values between systems and over time

4.37
verify
verification
one-to-one process of comparing a single submitted biometric sample against a biometric reference template to determine whether the submitted biometric sample matches the reference template

Note 1 to entry: This is often called a "verification match" or "verifymatch".

4.38
interworking standards
standards that modify the behaviour of the BioAPI Framework (and BioAPI conformance requirements) to support the use of communications links to enable an application to interact with a remote BSP using a standardised protocol

4.39
test-verify
test-verification
one-to-one process of comparing a single biometric sample against a candidate of a biometric reference template at enrollment to determine whether the candidate template matches the test sample

4.40
enroll type
value denoting a pattern of suboperations performed by a BSP during an enroll operation

Documents
Document Sharing Hub
© Standards Australia Limited 2019

Full document details: AS ISO IEC 19784.1-2019

4 Symbols and abbreviated terms

ACBio	Authentication Context for Biometrics
API	Application Programming Interface
BDB	Biometric Data Block
BFP	BioAPI Function Provider
BIR	Biometric Information Record
BPU	Biometric Processing Unit
BSP	Biometric Service Provider
CBEFF	Common Biometric Exchange Formats Framework
FMR	False Match Rate
FPI	Function Provider Interface
GUI	Graphical User Interface
ID	Identity/Identification/Identifier
IRI	Internationalized Resource Identifier (see RFC 3987)
MAC	Message Authentication Code
MOC	Match on Card
PID	Product ID
SB	Security Block
SBH	Standard Biometric Header NOTE This term and abbreviation is imported from ISO/IEC 19785-1.
SPI	Service Provider Interface
UUID	Universally Unique Identifier

5 Conformance

Annex A specifies the conformance requirements for BioAPI components claiming conformance to this document.

5.2 This document uses the C programming language (see ISO/IEC 9899) to specify the interfaces that it defines. A BioAPI component can conform to this document by the provision or use of that interface with languages other than the C programming language, provided that the component on the other side of such an interface can use the interface through the detailed C programming language specification given in this document. (See also 7.1.)

© Standards Australia Limited 2019

Documents
Document Sharing Hub

Full document details: AS ISO IEC 19784.1-2019

6 The BioAPI architecture

6.1 The full BioAPI API/SPI Architectural Model

6.1.1 The BioAPI incorporates an API/SPI model, illustrated in Figure 1.

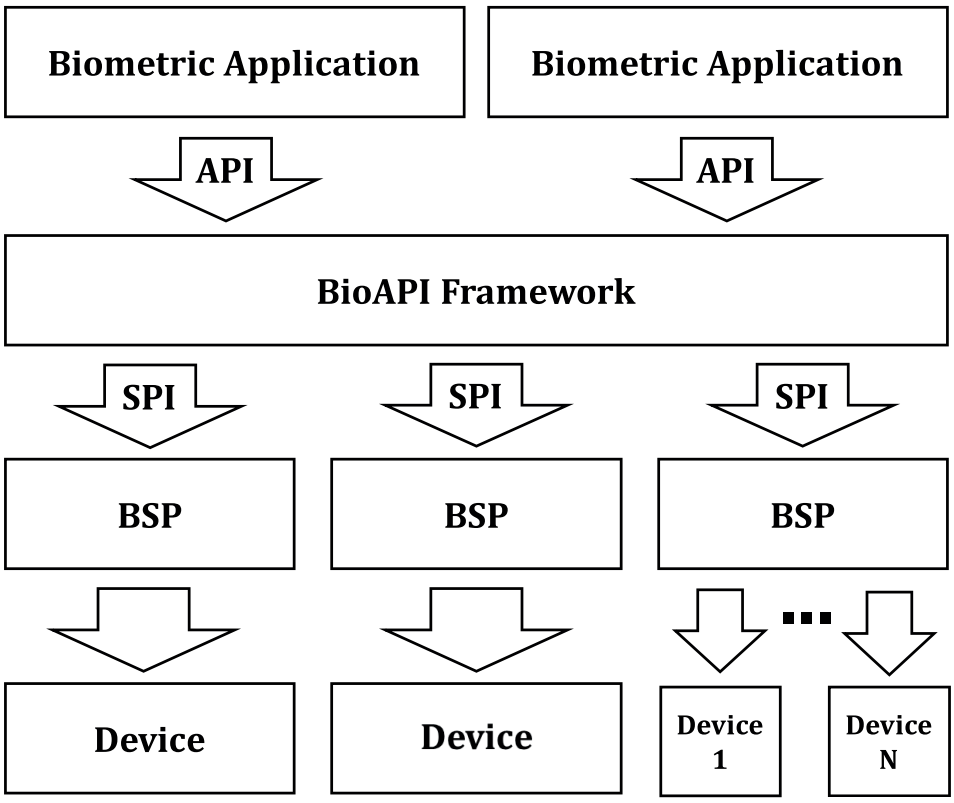

Figure 1 — BioAPI API/SPI Model

NOTE The structure below the SPI is simplified in Figure 1. The options (and standardized interfaces) for the internal structure of the BSPs are presented in Figure 2.

6.1.2 A full BioAPI system consists of BioAPI components providing standardized interfaces. Figure 1 shows interactions between three sorts of BioAPI components: the BioAPI Framework, applications, and monolithic BSPs.

NOTE Standardized interfaces to allow the functions of a BSP to be provided by independent BioAPI components are described in 6.3.

6.1.3 The API specifies the interface between the BioAPI Framework and a biometric application. The application is written to invoke the functions in the API specification (see Clause 8). The BioAPI Framework supports the functions in the API specification.

6.1.4 The SPI provides the interface between the BioAPI Framework and BSPs. The BioAPI Framework invokes the functions in the SPI specification (see Clause 9). The BSP supports the functions of the SPI specification.

6.1.5 The BioAPI Framework provides for management of BSPs and for the mapping of function calls from an API function to an SPI function addressed to the appropriate BSP.

© Standards Australia Limited 2019

Documents Document Sharing Hub

Full document details: AS ISO IEC 19784.1-2019

6.1.6 An application can access the functionalities of a BSP (through the BioAPI Framework) only after the BSP has been loaded and attached (see clauses 8.1.5 8.1.7 (BioAPI 2.1 or less), and 8.1.8 (BioAPI 2.2 or greater). When the application no longer requires the use of the BSP, it is detached and unloaded (see clauses 8.1.6 and 8.1.9).

6.1.7 An application may load (and attach to) more than one BSP at a time. A BSP may be attached to more than one application at a time.

NOTE Interactions between an application and an attached BSP are normally independent of other interactions between that application and other BSPs, or between that BSP and other applications, except when there is contention for a physical device managed by the BSP.

6.1.8 The functions specified in document support the presence in a biometric system of:

a) a singleBioAPI Framework with an associated component registry;

b) dynamic execution and termination of multiple simultaneous biometric applications interworking with that BioAPI Framework;

c) the dynamic installation, deinstallation (and associated loading and unloading) of multiple BSPs interworking with that BioAPI Framework;

d) signaling from a BSP to the BioAPI Framework (and hence to running biometric applications) of events related to the dynamic connection and disconnection of BioAPI Units (see clause 6.6) managed by that BSP;

e) providing graphical information to the application about an ongoing enrollment, verification, or identification operation.

NOTE It is expected that the most common use of this API will be for a single application to control one BSP with at most one BioAPI Unit of each biometric category accessed through that BSP. However, support for an application to access multiple BSPs, each capable of managing multiple BioAPI Units (but only one of each category for a given attached session) will facilitate such applications as physical access control, especially when networked devices are employed.

6.2 The framework-free BioAPI Architectural Model

6.2.1 The framework-free architecture recognises the use of BioAPI-conforming BSPs that provide (a defined subset of) the SPI interface for integration into systems in which other components are not standardised.

6.2.2 The conformance requirements in this case are reduced to the minimum necessary for reliable BSP operation, and apply only to BSPs.

6.2.3 The architecture within a BSP (see 6.3) still applies, but many BSPs intended for use in framework-free systems are likely to be (but need not be) monolithic implementations from a single vendor.

6.3 The BioAPI BSP Architectural Model

6.3.1 BioAPI Units are the abstraction of biometric devices and are the basic building blocks presented to an application by a BSP. They encapsulate and hide software and hardware resources (of various types – sensor device, archive device, etc.).

6.3.2 Each BioAPI Unit models or encapsulates a single (or none) piece of hardware and any necessary software. The following are the currently defined categories of BioAPI Unit:

— sensor unit;

© Standards Australia Limited 2019

Documents
Document Sharing Hub

Full document details: AS ISO IEC 19784.1-2019

— archive unit;

— matching algorithm unit;

— processing algorithm unit.

NOTE It is likely, but not required, that the first two units will be associated with detachable hardware, and that the last two units will not have any associated hardware.

6.3.3 A BioAPI Unit may be managed internally by a BSP (direct management of the BioAPI Unit) or may be managed by an associated BioAPI Function Provider (BFP) (indirect management of the BioAPI Unit). The BioAPI Function Provider Interfaces (FPIs) are specified in subsequent parts of ISO/IEC 19784.

6.3.4 A BFP can manage multiple BioAPI Units of a given category (but not more than one in any attach session for a given application). BFPs are categorized according to the categories of the BioAPI Units that they can manage. The following are the current categories of BFPs:

— sensor BFP;

— archive BFP;

— matching algorithm BFP;

— processing algorithm BFP.

6.3.5 The BioAPI FPIs for each of the categories of BFP are specified in other parts of ISO/IEC 19784. All FPIs support access to one or more (but not more than one in any attach session for a given application) BioAPI Units in the addressed BFP.

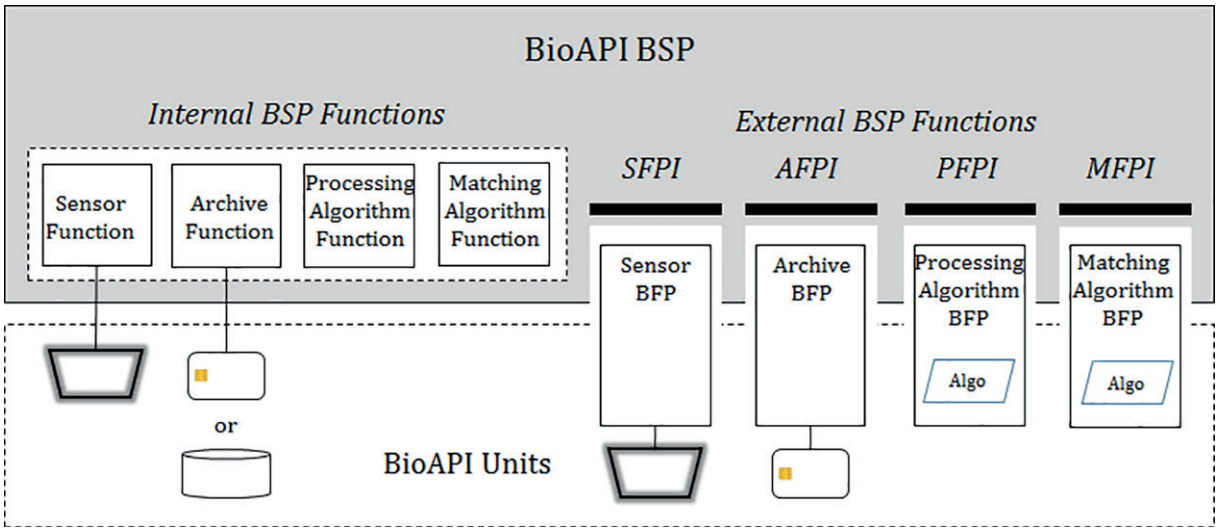

Figure 2 — Illustration of BSP architecture

6.3.6 A BSP that supports multiple BioAPI Units (either directly managed or through BFPs) supports the SPI interface that enables an application to select a particular BioAPI Unit (one of each category) for an attach session. The application may indicate BioAPI_DONT_CARE for the BioAPI Unit of a particular category, in which case the BSP selects the BioAPI Unit to be used.

6.3.7 If the implementer of a BSP claims (see clause 7.71) to support a particular category of BioAPI Unit, then the BSP is required either to be able to manage a BioAPI Unit of that category directly or to be able to interact with a BFP of a corresponding category through an associated (standardized) FPI.

Documents Document Sharing Hub
© Standards Australia Limited 2019

Full document details: AS ISO IEC 19784.1-2019

6.3.8 There is a BioAPI API function and corresponding SPI function (and in some cases FPI functions) (see clauses 8.1.14 and 9.3.1.9) that allow an application to send/request control/status information to/from a BioAPI Unit using the BioAPI API and SPI (and FPI). This is in addition to the normal biometric operations. The parameters of the control functions are not standardized. If either the BSP or BFP (if involved) does not support this control function, an error is returned (see clause 6.7 for further details).

6.3.9 When a BSP attach session or a BSP attach secure session is established, at most one BioAPI Unit of each category is selected (see clause 8.1.7 and 8.1.8).

NOTE 1 This architecture requires that the FPI interfaces allow the BSP to pass to a BFP the identification (provided through the BioAPI and the BioSPI interface) of the precise BioAPI Unit that is to be used. This implies the particular camera, scanner, storage device, etc. that is to be used. Subsequent parts of ISO/IEC 19784 define the necessary functions of the FPI for each category of BFP.

NOTE 2 At the time of standardizing this document, the FPIs for application of ISO/IEC 24761 are not specified yet. Therefore ISO/IEC 24761 is applicable only to the BSPs which implement internal BSP functions and do not have external BFPs attached because an ACBio instance shall be generated in a BioAPI Unit.

6.4 The component registry

NOTE The component registry function in a framework-free system is either not available, or is provided by non-standardised interfaces to platform-dependent functions. This sub-clause is not relevant for such systems.

6.4.1 The component registry contains information about installed BSPs and BFPs.

6.4.2 In the BioAPI model, there is a single BioAPI Framework and a single associated component registry in a biometric system.

NOTE In a real computer system, there may be multiple component registries, either supported by the same (shared) BioAPI Framework code, or by different BioAPI Framework code (perhaps old and new versions of the Framework). This is modelled as multiple biometric systems in a single real computer system.

6.4.3 It is a BioAPI requirement that there shall be no interaction or interference between different biometric systems implemented in a single, real computer system.

NOTE Where a real computer system contains multiple biometric systems, the possible sharing of code and the means by which an application is bound to one or other biometric system is an implementation matter.

6.4.4 The following information can be obtained by an application from BioAPI Framework functions that return information in the component registry (see clauses 8.1.3, 8.1.4, and 8.1.11):

a) information about the BioAPI Framework itself;

b) details of all installed BSPs;

c) details of all installed BFPs.

NOTE Information about installed BFPs can also be obtained by a BSP through a callback mechanism.

6.4.5 The following information can be obtained by an application from BioAPI Framework functions that are passed via the SPI to a specific BSP (see clauses 8.1.12 and 8.1.10):

a) details of all the installed BFPs that are supported by that BSP;

b) details of all BioAPI Units that are in the inserted state and are accessible through that BSP (either directly or through a supported BFP).

6.4.6 The information listed in clauses 6.4.4 and 6.4.5 can be obtained by an application using functions that can be called at the following times:

© Standards Australia Limited 2019

Documents
Document Sharing Hub

Full document details: AS ISO IEC 19784.1-2019

a) information about the framework itself can be obtained at any time after ***BioAPI_Init*** or ***BioAPI_InitEndpoint*** (in BioAPI 2.1 or greater) (see clause 8.1.3);

b) details of all installed BSPs can be obtained at any time after ***BioAPI_Init*** or ***BioAPI_InitEndpoint*** (in BioAPI 2.1 ;r greater) (see clause 8.1.4);

c) details of all installed BFPs can be obtained at any time after ***Bi;API_Init*** or ***BioAPI_InitEndpoint*** (in BioAPI 2.1 or greater) (see clause 8.1.11);

d) details of all the installed BFPs that are supported by each BSP can be obtained at any time after ***BioAPI_BSPLoad*** (see clause 8.1.12) of that BSP;

NOTE 1 Multiple BSPs can be simultaneously loaded by an application.

NOTE 2 This implies that when a ***BioAPI_BSPLoad*** occurs, the BSP uses the FPI to load all the BFPs that it is capable of using.

e) details of all BioAPI Units that are in the inserted state and that are accessible through each BSP (either directly or through a supported BFP) can be obtained (by quoting the BSP UUID) at any time after ***BioAPI_BSPLoad*** (see clause 8.1.10) of that BSP.

6.5 BSP and BFP Installation and De-installation

NOTE The installation and de-installation functions in a framework-free system are either not available, or are provided by non-standardised interfaces to platform-dependent functions. This sub-clause is not relevant for such systems.

6.5.1 On installation of a BSP or BFP, a UUID is required (a parameter of ***BioAPI_Util_InstallBSP*** and ***BioAPI_Util_InstallBFP***) to identify the BSP or BFP. If there is an attempt to install a BSP or BFP with the same UUID as an already installed BSP or BFP, then the install shall fail (return an error). If the same BSP or BFP is installed on multiple biometric systems, then it may or may not (but should) have the same UUID on all systems. The UUID is required to be distinct across all BSPs and BFPs in a single biometric system.

6.5.2 The installation of a BSP is done by calling a BioAPI Framework function (see clause 10.2.1) and supplying details of the BSP (the BSP schema, specified in clause 7.22 (BioAPI 2.0) and 7.23 (BioAPI 2.1 or greater).

NOTE This function call is standardized, but its invocation is typically from an installation wizard application rather than a normal biometric application.

6.5.3 When a BSP is newly installed, it will typically be activated by the installation process using implementation-dependent mechanisms and can then use a callback mechanism (see clause 9.2.2) in order to obtain information from the Framework about installed BFPs.

NOTE This function can be called at any time to enable a BSP to update any internal information that it might maintain. The format and content (and even the existence) or such internal information is not standardized and is purely a matter for the BSP.

6.5.4 After successful installation or de-installation of a BSP or BFP, the next call from any application or BSP retrieving information from the component registry shall return correct information related to a newly installed BSP or BFP and no information about an uninstalled BSP or BFP (see clauses 8.1.4 and 8.1.11).

6.5.5 There are no mechanisms to inform an installed BSP (whether currently in use or not) about the installation of new BFPs or the de-installation (see also clause 6.5.6) of existing BFPs. Responsibility for determining what BFPs are installed resides solely with the BSPs using the BFP Enumeration Handler callback (see clause 9.2.2).

© Standards Australia Limited 2019

Documents Document Sharing Hub

Full document details: AS ISO IEC 19784.1-2019

6.5.6 There is a BioAPI API function (see clause 10.2.1) that can be called by an application (for example, a de-installation wizard) to inform the BioAPI Framework that a BSP has been uninstalled. The BioAPI Framework will update the component registry. There are no standardized functions to inform BSP instances in the working memory of running applications about the de-installation of a BFP that it is using, and the effect of uninstalling a BFP that is in use by a BSP is implementation-dependent.

NOTE The de-installation of a BFP will not normally uninstall related hardware or drivers for BioAPI Units, as such drivers and hardware can also be under the control of other installed BFPs. This area is outside the standardization of BioAPI, and is again implementation-dependent.

6.6 BSP Load and BioAPI Unit Attachment

NOTE These functions in a framework-free system are either not available, or are provided by non-standardised interfaces to platform-dependent functions. This sub-clause is not relevant for such systems.

6.6.1 The actions of an application wishing to perform biometric operations are (in order):

a) initialize access to the BioAPI Framework (see clause 8.1.1);

b) identify and load one or more BSPs (see clauses 8.1.4 and 8.1.5), optionally specifying a callback event handler to receive callbacks when defined events (see clause 7.35) occur related to that BSP (for example, insertion or removal of a BioAPI Unit that can be accessed through that BSP). When callback event handlers are specified, the application will receive notifications about all insertions or removals that the associated BSPs become aware of;

c) attach a BSP together with at most one BioAPI Unit of each category (either directly or indirectly accessed by that BSP);

NOTE 1 'attach a BSP' means 'establish an attach session to a BSP'.

d) following the attach, the other BioAPI function calls can be made with reference to the attached BSP, and will be handled by the attached BSP using the identified BioAPI Units of the required category.

NOTE 2 It is possible for an application to establish multiple simultaneous attach sessions with different BSPs (or even with the same BSP).

NOTE 3 Annex D provides sample calling sequence code.

6.6.2 Loading a BSP (see clause 8.1.5) enables the application to subsequently obtain full information about the BioAPI Units that can be accessed (directly or indirectly) through that BSP, either through a Framework query or by a callback notification from the BSP (or both).

6.6.3 A BioAPI Unit can be used if and only if any hardware or software resource that it depends on is currently available in the system. This is modeled by saying that the BioAPI Unit is in the inserted state.

6.6.4 When an application attaches a BSP it may indicate that the BioAPI Unit to be selected for any particular BioAPI Unit category should be determined by the BSP. This is called "selecting a default BioAPI Unit using BioAPI_DONT_CARE".

NOTE The only difference between selecting a specific BioAPI Unit (that is in the inserted state) of a given category and selecting the default BioAPI Unit using BioAPI_DONT_CARE is that the decision on which BioAPI Unit to use is taken by the BSP. There is no other difference.

6.6.5 The event notification about each inserted BioAPI Unit includes its schema.

NOTE It is possible that the physical insertion of some piece of hardware (for example, a smartcard that can support both archiving and matching) might produce two separate and different event notifications. The application may not be able to relate these two events to the same physical device.

© Standards Australia Limited 2019

Documents
Document Sharing Hub

Full document details: AS ISO IEC 19784.1-2019

6.6.6 On ***BioAPI_BSPAttach/BioAPI_BSPAttachSecure*** (BioAPI 2.2 or greater), the application is required to select at most one BioAPI Unit of each category that is currently in an "inserted" state (or to select BioAPI_DONT_CARE) and is managed by that BSP or by an associated BFP. The BSP then either accesses that BioAPI Unit (for directly managed BioAPI Units), or else interacts with the associated BFP in order to access that BioAPI Unit.

6.6.7 The information content of removal notification includes (see clause 7.38):

a) the BioAPI Unit ID;

b) event type (remove);

c) callback context.

6.6.8 The occurrence of a removal event notification for a BioAPI Unit that forms part of the set of BioAPI Units used in an attach session requires that the application issue no further function calls other than ***BioAPI_BSPDetach*** and ***BioAPI_GetBIRFromHandle***.

6.7 Controlling BioAPI Units

6.7.1 A function (see clauses 8.1.13 and 9.3.1.8) is available in the BioAPI API, SPI, and FPIs that enable an application to control a BioAPI Unit through a BSP. A BSP is not required to support this function, and identifies such support in its schema (see clauses 7.22 (BioAPI 2.0), 7.23 (BioAPI 2.1 or greater), and 7.70).

NOTE The schema may not be available in framework-free systems.

6.7.2 This function contains specific control codes, buffer contents, and return values, but the format and meaning of these are identified by a UUID and may be vendor specific. There are no standardized UUIDs for this function in this version of this document.

6.8 BIR Structure and Handling

6.8.1 BIR Structure

The BIR that crosses the API and SPI is a C data structure stored in memory using pointers to various elements. The BIR structure recommended for storage and transfer between computer systems is a serialization of that C data structure and is shown graphically in Figure 3 below. In addition to the fields specified in the C structure, length fields are included in the recommended storage and transfer format.

NOTE Annex B specifies the BioAPI Patron Format which is recommended for storage and transmission of the BioAPI BIR. Representation of the BioAPI BIR in the API and SPI uses the datastructure specified in clause 7.7.

© Standards Australia Limited 2019

Documents Document Sharing Hub

Full document details: AS ISO IEC 19784.1-2019

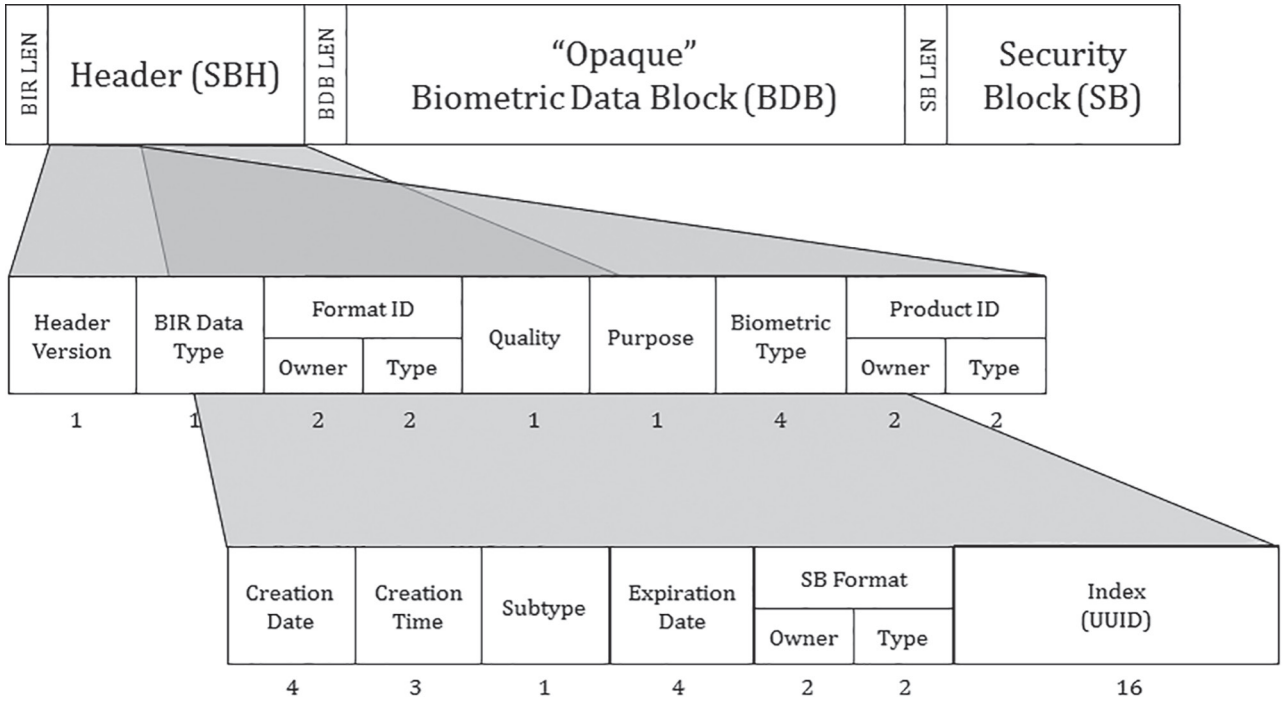

Figure 3 — Biometric Information Record (serialized BIR for storage and transmission)

The Standard Biometric Header (SBH) contains information describing the contents of the following BDB.

The BDB contains the biometric sample whose format is defined by the Format ID field of the SBH. This may be a standard or proprietary format.

NOTE Standard BDB formats are defined in ISO/IEC 19794.

The security block (SB) is optional (its absence is indicated by a zero length field), and contains parameters associated with the signing and/or encryption of the BIR. The format of the SB is determined by the SB Format field of the SBH. The BIR Data Type indicates whether the BIR is signed and/or the BDB is encrypted (see clause 7.13).

The first length field is the length of the entire BIR, not including this length field itself. The second length field, which is located after the header and before the BDB, is the length of the BDB (not including the length field itself). The third length field, which is located after the BDB and before the SB, is the length of the SB (not including the length field itself).

6.8.2 BIR Data Handling

When a BSP creates a new BIR, it returns a "handle" to it. Most local operations can be performed without moving the BIR out of the BSP. (BIRs can be quite large, so this is a performance advantage.) However, if the application needs to manage the BIR (either to store it in an application database, to pass it to another BSP, or to send it to a server for verification/identification, possibly via the Framework using ISO/IEC 24708[6]), it can acquire the BIR using the handle (in the function ***BioAPI_GetBIRFromHandle***). If just the header information is needed, it can be retrieved using ***BioAPI_GetHeaderFromHandle***.

Whenever an application needs to provide a BIR as input to a BioAPI function, it can be done in one of three ways:

a) by reference to its handle (if it is in the BSP being invoked),

b) by reference to its key value (UUID) in an open BIR database (controlled by the BSP being invoked), or

© Standards Australia Limited 2019

Documents
Document Sharing Hub

Full document details: AS ISO IEC 19784.1-2019

c) by supplying the BIR itself using the C structure BioAPI_BIR.

7 BioAPI types and macros

NOTE Many of these types and macros are not relevant for the implementation of a BSP designed to work in a framework-free BioAPI system.

7.1 BioAPI

Definition of the BioAPI calling conventions.

```
#ifdef WIN32
#define BioAPI __stdcall
#else
#define BioAPI
#endif
```

Note that interworking of C programs from different vendors in general depends on choices made (sometimes selectable by statements in the C header) on the in-core representation (for example, padding between data elements, parameter passing mechanisms, and use of registers or stacks). On many operating systems, there is a well understood default for such choices that is in common use, which should be used. Where there is no recognized default, the option that represents data structures without padding between data elements should be used.

7.2 BioAPI_ACBio_PARAMETERS (BioAPI 2.2)

7.2.1 Structure giving information which is used to generate ACBio instances

```
typedef struct bioapi_acbio_parameters {
   uint32_t Challenge[4];
   uint32_t *InitialBPUIOIndexOutput;
   uint32_t *SupremumBPUIOIndexOutput;
} BioAPI_ACBio_PARAMETERS;
```

7.2.2 Definitions

— *Challenge* – Challenge from the validator of a biometric verificatioin when ACBio is used. This value shall be set to the field `controlValue` of type `ACBioContentInformation` in ACBio instances.

— *InitialBPUIOIndexOutput* – The initial value of BPU IO index which is to be assigned to the output from the BioAPI Unit, BFP, or BSP when the ACBio instances are generated. The range between *InitialBPUIOIndexOutput* and *SupremumBPUIOIndexOutput* shall be divided into the number of BSP Units and BFPs which are attached to the BSP, and assigned to the BSP Units and BSPs.

— *SupremumBPUIOIndexOutput* – The supremum of BPU IO indexes which are to be assigned to the output from the BioAPI Unit, BFP, or BSP when the ACBio instances are generated.

7.3 BioAPI_ASN1_BIR (BioAPI 2.2)

A container for biometric data, (or non-biometric data) that may affect a biometric operation, encoded in ASN.1 DER. The BioAPI_ASN1_BIR structure is used when a BioAPI API with security functionality is used. The BioAPI_ASN1_BIR structure associates a length, in bytes, with the address of an arbitrary block of contiguous memory which contains an ASN.1 encoded BIR of type `BioAPI-BIR`, specified in Annex E. The ASN.1 encoded BIR consists of an SBH of type `BioAPIBIRHeader`, a BDB of type `BiometricData`, and an SB of type `CBEFFSecurityBlock`. The BDB may contain raw sample data, partially processed (intermediate) data, completely processed data, score data (resulting from a matching or fusion operation), decision data (resulting from a decision or fusion operation), or biographic data which can be provided as input to a biometric operation to modify its behavior. The BioAPI_ASN1_BIR may be used to enroll a user (thus being stored persistently), or may be used to verify

© Standards Australia Limited 2019

Full document details: AS ISO IEC 19784.1-2019

or identify a user (thus being used transiently). It may also be stored and used later to provide feedback to subsequent biometric operations. Type BioAPI_ASN1_BIR is an alias of type BioAPI_DATA.

NOTE The ASN.1 type BioAPI-BIR corresponds to the C structured type BioAPI_BIR element by element.

```
typedef BioAPI_DATA BioAPI_ASN1_BIR;
```

7.4 BioAPI_ASN1_ENCODED (BioAPI 2.2)

A container for ASN.1 DER encoded data. The BioAPI_ASN1_ENCODED structure is used to express information about cryptographic keys. The BioAPI _ASN1_ ENCODED structure associates a length, in bytes, with the address of an arbitrary block of contiguous memory which contains an ASN.1 encoded data. Type BioAPI_ASN1_ENCODED is an alias of type BioAPI_DATA.

typedef BioAPI_DATA BioAPI_ASN1_ENCODED;

7.5 BioAPI_BFP_LIST_ELEMENT

7.5.1 Identifies a BFP, giving its category and UUID. A list is returned by a BSP when queried for the installed BFPs that it supports.

```
typedef struct bioapi_bfp_list_element  {
        BioAPI_CATEGORY BFPCategory;
        BioAPI_UUID BFPUuid;
   } BioAPI_BFP_LIST_ELEMENT;
```

7.5.2 Definitions

— *BFPCategory* – Defines the category of the BioAPI Unit that the BFP supports.

— *BFPUuid* – UUID of the BFP in the component registry.

7.6 BioAPI_BFP_SCHEMA

7.6.1 Information about a BFP, maintained in the component registry.

```
typedef struct bioapi_bfp_schema {
       BioAPI_UUID BFPUuid;
       BioAPI_CATEGORY BFPCategory;
       BioAPI_STRING BFPDescription;
       uint8_t *Path;
       BioAPI_VERSION SpecVersion;
       BioAPI_STRING ProductVersion;
       BioAPI_STRING Vendor;
       BioAPI_BIR_BIOMETRIC_DATA_FORMAT *BFPSupportedFormats;
       uint32_t NumSupportedFormats;
       BioAPI_BIR_BIOMETRIC_TYPE FactorsMask;
       BioAPI_UUID BFPPropertyID;
       BioAPI_DATA BFPProperty;
   }BioAPI_BFP_SCHEMA;
```

7.6.2 Definitions

— *BFPUuid* – UUID of the BFP.

— *BFPCategory* – Category of the BFP identified by the BFPUuid.

— *BFPDescription* – A NUL-terminated string containing a text description of the BFP.

— *Path* – A pointer to a NUL-terminated string containing the path of the BFP file, including the filename. The path may be a URL. This string shall consist of ISO/IEC 10646 characters encoded in UTF-8 (see ISO/IEC 10646:2017, Annex D).

NOTE 1 When BioAPI_BFP_SCHEMA is used within a function call, the component that receives the call allocates the memory for the *Path* schema element and the calling component frees the memory

© Standards Australia Limited 2019

Documents
Document Sharing Hub
Full document details: AS ISO IEC 19784.1-2019

— *SpecVersion* – Major/minor version number of the FPI specification to which the BFP was implemented.

NOTE 2 FPI specifications will be issued as subsequent, additional parts of the ISO/IEC 19784 series.

— *ProductVersion* – The version string of the BFP software.

— *Vendor* – A NUL-terminated string containing the name of the BFP vendor.

— *BFPSupportedFormats* – A pointer to an array of BioAPI_BIR_BIOMETRIC_DATA_FORMAT structures specifying the supported BDB formats.

— *NumSupportedFormats* – Number of supported formats contained in BFPSupportedFormats.

— *FactorsMask* – A mask which indicates which biometric types are supported by the BFP.

— *BFPPropertyID* – UUID of the format of the following BFP property.

— *BFPProperty* – Address and length of a memory buffer containing the BFP property. The format and content of the BFP property can either be specified by a vendor or can be specified in a related standard.

7.7 BioAPI_BIR

7.7.1 A container for biometric data, or non-biometric data that may affect a biometric operation. A BioAPI_BIR consists of a BioAPI_BIR_HEADER, a BDB, and an optional SB. The BDB may contain raw sample data, partially processed (intermediate) data, completely processed data, score data (resulting from a matching or fusion operation), decision data (resulting from a decision or fusion operation), or biographic data which can be provided as input to a biometric operation to modify its behavior. The BioAPI_BIR may be used to enroll a user (thus being stored persistently), or may be used to verify or identify a user (thus being used transiently). It may also be stored and used later to provide feedback to subsequent biometric operations.

7.7.2 The BDB and SB are an integral number of octets and are of variable length, up to 2**32-1 octets. If the SB contains a signature, it is calculated on the combined BioAPI_BIR_HEADER and BDB.

```
typedef struct bioapi_bir {
       BioAPI_BIR_HEADER Header;
       BioAPI_DATA BiometricData;
       BioAPI_DATA SecurityBlock;        /* SecurityBlock.Data=NULL if no SB */
} BioAPI_BIR;
```

NOTE 1 The BioAPI BIR contains the information needed for a CBEFF Patron Format as defined in the Common Biometric Exchange Formats Framework (CBEFF), ISO/IEC 19785-1. See Annex B for the specification of the BioAPI BIR Patron Format, including internetworking and storage formats.

NOTE 2 The format of the BiometricData and SecurityBlock are determined by the BioAPI_BIR_BIOMETRIC_DATA_FORMAT and BioAPI_BIR_SECURITY_BLOCK_FORMAT elements of the BioAPI_BIR_HEADER respectively.

NOTE 3 CBEFF (ISO/IEC 19785-1) allows the possibility of different BIR formats from that supported by BioAPI. Conversion between the BioAPI BIR format and other BIR formats is specifed in ISO/IEC 19785-1.

7.8 BioAPI_BIR_ARRAY_POPULATION

An array of BIRs, generally used during identification operations (as input to ***BioAPI_Identify*** or ***BioAPI_IdentifyMatch*** as part of BioAPI_IDENTIFY_POPULATION).

```
typedef struct bioapi_bir_array_population {
   uint32_t NumberOfMembers;
   BioAPI_BIR  *Members;  /* A pointer to an array of BIRs */
} BioAPI_BIR_ARRAY_POPULATION;
```

© Standards Australia Limited 2019

Documents Document Sharing Hub

Full document details: AS ISO IEC 19784.1-2019

7.9 BioAPI_BIR_BIOMETRIC_DATA_FORMAT

Defines the format of the data contained within the "opaque" biometric data block (BDB) of the BioAPI BIR, BiometricData element.

```
typedef struct bioapi_bir_biometric_data_format {
   uint16_t FormatOwner;
   uint16_t FormatType;
} BioAPI_BIR_BIOMETRIC_DATA_FORMAT;
```

FormatOwner values are assigned and registered by the CBEFF Registration Authority. Format Type is assigned by the Format Owner and may optionally be registered.

NOTE 1 The BioAPI BIR Biometric Data Format corresponds to a combination of the "CBEFF_BDB_format_owner" and "CBEFF_BDB_format_type" in ISO/IEC 19785-1.

NOTE 2 This structure is primarily used within a BIR header; however, it is also used as an input parameter for functions that capture biometric data.

7.10 BioAPI_BIR_BIOMETRIC_PRODUCT_ID

Provides the product identifier (PID) for the BSP that generated the BDB in the BIR (the BiometricData element).

```
typedef struct bioapi_bir_biometric_product_ID {
   uint16_t ProductOwner;
   uint16_t ProductType;
} BioAPI_BIR_BIOMETRIC_PRODUCT_ID;

   #define BioAPI_NO_PRODUCT_OWNER_AVAILABLE     (0x0000)
   #define BioAPI_NO_PRODUCT_TYPE_AVAILABLE      (0x0000)
```

The condition NO VALUE AVAILABLE shall be indicated by setting the value of all components to zero. This value shall be used only for BIRs that are not originally generated by a BioAPI BSP, but originate from another source and have been transformed into a BioAPI BIR. BSPs shall not use this value.

Product Owner values are assigned and registered by the CBEFF Registration Authority as Biometric Organization identifiers. Product Type is assigned by the Product Owner and may optionally be registered.

NOTE 1 Product IDs are analogous to Format IDs and the registration process is the same. A single vendor can register for one Format/Product Owner value (a Biometric Organization identifier) that can be used in both fields.

NOTE 2 The BioAPI BIR Biometric Product ID corresponds to the "CBEFF_BDB_product_owner" and "CBEFF_BDB_product_type" in ISO/IEC 19785-1.

7.11 BioAPI_BIR_BIOMETRIC_TYPE (BioAPI 2.0)

This subclause applies only when the BioAPI version number in use is 2.0.

A mask that describes the set of biometric types (factors) contained within a BioAPI BIR or supported by a BSP.

```
typedef uint32_t BioAPI_BIR_BIOMETRIC_TYPE;

   #define BioAPI_NO_TYPE_AVAILABLE       (0x00000000)
   #define BioAPI_TYPE_MULTIPLE           (0x00000001)
   #define BioAPI_TYPE_FACIAL_FEATURES    (0x00000002)
   #define BioAPI_TYPE_VOICE              (0x00000004)
   #define BioAPI_TYPE_FINGERPRINT        (0x00000008)
   #define BioAPI_TYPE_IRIS               (0x00000010)
   #define BioAPI_TYPE_RETINA             (0x00000020)
   #define BioAPI_TYPE_HAND_GEOMETRY      (0x00000040)
```

© Standards Australia Limited 2019

Documents
Document Sharing Hub

Full document details: AS ISO IEC 19784.1-2019

```
#define BioAPI_TYPE_SIGNATURE_DYNAMICS (0x00000080)
#define BioAPI_TYPE_KEYSTOKE_DYNAMICS  (0x00000100)
#define BioAPI_TYPE_LIP_MOVEMENT       (0x00000200)
#define BioAPI_TYPE_THERMAL_FACE_IMAGE (0x00000400)
#define BioAPI_TYPE_THERMAL_HAND_IMAGE (0x00000800)
#define BioAPI_TYPE_GAIT               (0x00001000)
#define BioAPI_TYPE_OTHER              (0x40000000)
#define BioAPI_TYPE_PASSWORD           (0x80000000)
```

NOTE 1 BioAPI_TYPE_MULTIPLE is used to indicate that the biometric samples contained within the BDB (BIR BiometricData) include biometric samples from more than one type of biometric sensor unit (e.g., fingerprint and facial data). Location of the individual samples within the BDB is specified by the Format Owner and identified by the value of the Format Type.

NOTE 2 The condition NO VALUE AVAILABLE is indicated by setting the value to zero. BIRs that are not originally created by BioAPI BSPs should use this value when transformed into a BioAPI BIR if Biometric Type information is not available in the original source record. Transformed BIRs whose biometric type does not correspond to one of the defined types shall use the value for OTHER.

NOTE 3 The BioAPI BIR Biometric Type corresponds to the "CBEFF_BDB_biometric_type" in ISO/IEC 19785-1.

7.12 BioAPI_BIR_BIOMETRIC_TYPE (BioAPI 2.1)

This subclause applies only when the BioAPI version number in use is 2.1 or greater.

A mask that describes the set of biometric types (factors) contained within a BioAPI BIR or supported by a BSP.

In the following definition of BioAPI_BIR_BIOMETRIC_TYPE, the condition NO VALUE AVAILABLE is indicated by setting the value to zero. BIRs that are not originally created by BioAPI BSPs should use this value when transformed into a BioAPI BIR if Biometric Type information is not available in the original source record. Transformed BIRs whose biometric type does not correspond to one of the defined types shall use the value for OTHER.

```
typedef uint32_t BioAPI_BIR_BIOMETRIC_TYPE;

#define BioAPI_NO_BIOTYPE_AVAILABLE            (0x00000000)
#define BioAPI_TYPE_MULTIPLE_BIOMETRIC_TYPES   (0x00000001)
#define BioAPI_TYPE_FACE                       (0x00000002)
#define BioAPI_TYPE_VOICE                      (0x00000004)
#define BioAPI_TYPE_FINGER                     (0x00000008)
#define BioAPI_TYPE_IRIS                       (0x00000010)
#define BioAPI_TYPE_RETINA                     (0x00000020)
#define BioAPI_TYPE_HAND_GEOMETRY              (0x00000040)
#define BioAPI_TYPE_SIGNATURE_SIGN             (0x00000080)
#define BioAPI_TYPE_KEYSTROKE                  (0x00000100)
#define BioAPI_TYPE_LIP_MOVEMENT               (0x00000200)
#define BioAPI_TYPE_GAIT                       (0x00001000)
#define BioAPI_TYPE_VEIN                       (0x00002000)
#define BioAPI_TYPE_DNA                        (0x00004000)
#define BioAPI_TYPE_EAR                        (0x00008000)
#define BioAPI_TYPE_FOOT                       (0x00010000)
#define BioAPI_TYPE_SCENT                      (0x00020000)
#define BioAPI_TYPE_OTHER                      (0x40000000)
#define BioAPI_TYPE_PASSWORD                   (0x80000000)
```

NOTE 1 BioAPI_TYPE_MULTIPLE_BIOMETRIC_TYPES is used to indicate that the biometric samples contained within the BDB (BIR BiometricData) include biometric samples from more than one type of biometric sensor unit (e.g., fingerprint and facial data). Location of the individual samples within the BDB is specified by the Format Owner and identified by the value of the Format Type.

NOTE 2 The BioAPI BIR Biometric Type corresponds to the "CBEFF_BDB_biometric_type" in ISO/IEC 19785-1.

© Standards Australia Limited 2019

Documents Document Sharing Hub

Full document details: AS ISO IEC 19784.1-2019

NOTE 3 Although “password” is not a biometric characteristic, BioAPI_TYPE_PASSWORD is included as a valid BioAPI_BIR_BIOMETRIC_TYPE to support its use a) for development & test environments, and b) as an additional authentication factor within a BioAPI application.

NOTE 4 The names of several biometric types are different in BioAPI 2.1 or greater from the names used in BioAPI 2.0, with no change in semantics, and several new biometric types are present in BioAPI 2.1 or greater. Finally, the "thermal" types are present in BioAPI 2.0 but are absent from BioAPI 2.1 or greater. These differences are in order to align with ISO/IEC 19785-1.

NOTE 5 The names of the following values differ between BioAPI 2.0 and BioAPI 2.1 or greater, but their semantics is the same. The BioAPI 2.1 or greater names are aligned with ISO/IEC 19785-1 (CBEFF).

Name in BioAPI 2.0	Name in BioAPI 2.1 or greater	Encoding
MULTIPLE	MULTIPLE_BIOMETRIC_TYPES	0x00000001
FACIAL_FEATURES	FACE	0x00000002
FINGERPRINT	FINGER	0x00000008
SIGNATURE_DYNAMICS	SIGNATURE_SIGN	0x00000080
KEYSTROKE_DYNAMICS	KEYSTROKE	0x00000100

NOTE 6 The following values are not present BioAPI 2.0 but are present in BioAPI 2.1 or greater for consistency with ISO/IEC 19785-1 (CBEFF).

Added value	Encoding
NO_BIOTYPE_AVAILABLE	0x00000000
VEIN	0x00002000
DNA	0x00004000
EAR	0x00008000
FOOT	0x00010000
SCENT	0x00020000

7.13 BioAPI_BIR_DATA_TYPE

7.13.1 This data type is used for three different purposes:

a) it identifies the type of biometric sample (raw, intermediate, processed, score data, decision data or biographic data) that is contained in the BDB;

b) it identifies whether the BioAPI BIR is encrypted and/or signed;

c) it identifies whether or not an index value is included as part of the BIR header.

NOTE If the BIR has been encrypted by a BSP, it may not be readable by an application or by another BSP.

7.13.2 Exactly one of the three flags "raw", "intermediate" and "processed" shall be set. If a BIR carrying a BIR data type with multiple flags set is passed to the BioAPI Framework in a parameter of a function call, a BioAPIERR_INVALID_BIR error shall be returned.

NOTE BIRs that are not originally created by a BioAPI BSP but have been transformed from another data format and for which sample type information is not available may not have set a flag. (BioAPI BSPs set one of the three flags.)

7.13.3 Each of the flags "encrypted" and "signed" may or may not be set.

7.13.4 The ‘index’ flag shall be set if an index is present in the BIR header and not set if no index is present in the BIR header.

© Standards Australia Limited 2019

Documents
Document Sharing Hub

Full document details: AS ISO IEC 19784.1-2019

```
typedef uint8_t BioAPI_BIR_DATA_TYPE;

   #define BioAPI_BIR_DATA_TYPE_RAW              (0x01)
   #define BioAPI_BIR_DATA_TYPE_INTERMEDIATE     (0x02)
   #define BioAPI_BIR_DATA_TYPE_PROCESSED        (0x04)
   #define BioAPI_BIR_DATA_TYPE_SCORE            (0x08)
   #define BioAPI_BIR_DATA_TYPE_DECISION         (0x09)
   #define BioAPI_BIR_DATA_TYPE_BIOGRAPHIC       (0x0A)
   #define BioAPI_BIR_DATA_TYPE_ENCRYPTED        (0x10)
   #define BioAPI_BIR_DATA_TYPE_SIGNED           (0x20)
   #defineBioAPI_BIR_INDEX_PRESENT               (0x80)
```

NOTE 1 The BioAPI BIR Data Type corresponds to a combination of the "CBEFF_BDB_processed_level", "CBEFF_BDB_encryption_options", and "CBEFF_BIR_integrity_options" in ISO/IEC 19785-1.

NOTE 2 BioAPI_BIR_DATA_TYPE_DECISION (BioAPI 2.2 or greater) and BioAPI_BIR_DATA_TYPE_BIOGRAPHIC (BioAPI 2.2 or greater) have two bits on while others have only one bit on.

NOTE 3 BioAPI_BIR_DATA_TYPE_SCORE is used in BioAPI 2.2 or greater.

7.14 BioAPI_BIR_HANDLE

A handle to refer to a BioAPI BIR or an ASN.1 encoded BIR that exists within a BSP.

NOTE A handle that identifies a BIR is a positive non-zero value. Other values of BioAPI_BIR_HANDLE are reserved for exception indications (currently only -1 and -2).

```
typedef int32_t BioAPI_BIR_HANDLE;

   #define BioAPI_INVALID_BIR_HANDLE          (-1)
   #define BioAPI_UNSUPPORTED_BIR_HANDLE      (-2)
```

7.15 BioAPI_BIR_HEADER

Standard information which describes the content of the opaque biometric data (BDB) that follows. This information is readable by the application and is provided to allow it to make processing and routing decisions regarding the BIR. The header is not encrypted by the BSP.

```
typedef struct bioapi_bir_header {
   BioAPI_VERSION HeaderVersion;
   BioAPI_BIR_DATA_TYPE Type;
   BioAPI_BIR_BIOMETRIC_DATA_FORMAT Format;
   BioAPI_QUALITY Quality;
   BioAPI_BIR_PURPOSE Purpose;
   BioAPI_BIR_BIOMETRIC_TYPE FactorsMask;
   BioAPI_BIR_BIOMETRIC_PRODUCT_ID ProductID;
   BioAPI_DTG CreationDTG;
   BioAPI_BIR_SUBTYPE Subtype;
   BioAPI_DATE ExpirationDate;
   BioAPI_BIR_SECURITY_BLOCK_FORMAT SBFormat;
   BioAPI_UUID Index;
} BioAPI_BIR_HEADER;
```

NOTE 1 The BioAPI BIR Header corresponds to the SBH in CBEFF, ISO/IEC 19785-1.

NOTE 2 Expiration date corresponds to the 'Valid to' portion of the "CBEFF_BDB_validity_period" in ISO/IEC 19785-1. The Index field corresponds to the "CBEFF_BDB_index" in ISO/IEC 19785-1.

© Standards Australia Limited 2019

Documents Document Sharing Hub

Full document details: AS ISO IEC 19784.1-2019

NOTE 3 It is possible that a BioAPI BIR may exist that has not been created by a BioAPI BSP but has been transformed from another data format. In this case, some of the header fields that are optional in CBEFF (ISO/IEC 19785-1) but required by BioAPI may not be present. For this reason, NO_VALUE_AVAILABLE or default values have been identified for these fields (within their corresponding data structures). However, all BIRs created by BioAPI BSPs shall include valid data for these fields and shall not use the NO VALUE AVAILABLE value. (The exceptions to this are BioAPI_QUALITY and BioAPI_BIR_SUBTYPE, which are optional in the BioAPI BIR header.) If such a non-BioAPI generated BIR is provided as an input parameter to a BioAPI BSP, the BSP may return an "Invalid BIR" error.

NOTE 4 The storage format for the BIR includes explicit length fields, which are not necessary in the C structure. See Annex B for the BIR storage format.

7.16 BioAPI_BIR_PURPOSE

7.16.1 A value which defines the purpose for which the BioAPI BIR is intended (when used as an input to a BioAPI function) or is suitable (when used as an output from a BioAPI function or within the BIR header).

```
typedef uint8_t BioAPI_BIR_PURPOSE;

   #define BioAPI_PURPOSE_VERIFY                          (1)
   #define BioAPI_PURPOSE_IDENTIFY                        (2)
   #define BioAPI_PURPOSE_ENROLL                          (3)
   #define BioAPI_PURPOSE_ENROLL_FOR_VERIFICATION_ONLY    (4)
   #define BioAPI_PURPOSE_ENROLL_FOR_IDENTIFICATION_ONLY  (5)
   #define BioAPI_PURPOSE_AUDIT                           (6)
   #define BioAPI_PURPOSE_DECIDE                          (7)
   #define BioAPI_NO_PURPOSE_AVAILABLE                    (0)
```

The condition NO VALUE AVAILABLE is indicated by setting the value to zero. This value is used only for BIRs that are not originally generated by a BioAPI BSP, but originate from another source and have been transformed into a BioAPI BIR. BSPs shall not use this value.

NOTE BioAPI_PURPOSE_DECIDE is used in BioAPI 2.2 or greater.

7.16.2 The Purpose value is utilized in two ways. First, it is used as an input parameter to allow the application to indicate to the BSP the purpose for which the resulting BIR is intended, thus allowing the BSP to perform the appropriate capturing and/or processing to create the proper BIR for this purpose. The second use is within the BIR header to indicate to the application (or to the BSP during subsequent operations) what purpose the BIR is suitable for. For example, some BSPs use different BDB formats depending on whether the data is to be used for verification or identification, the latter generally including additional information to enhance speed or accuracy. Similarly, many BSPs use different data formats depending on whether the data is to be used as a sample for immediate verification or as a reference template for future matching (i.e., enrollment).

NOTE The BioAPI BIR Purpose in a BIR header corresponds to the "CBEFF_BDB_purpose" in ISO/IEC 19785-1. The names differ slightly since the BioAPI BIR contains a single BDB, but the semantics are the same.

7.16.3 Restrictions on the use of BIR data of a particular purpose include:

a) All purposes are valid in the BIR header.

b) Purposes of BioAPI_PURPOSE_VERIFY and BioAPI_PURPOSE_IDENTIFY are only valid as input to the ***BioAPI_Capture*** function.

c) Purposes of BioAPI_PURPOSE_ENROLL, BioAPI_PURPOSE_ENROLL_FOR_VERIFICATION_ONLY, and BioAPI_PURPOSE_ENROLL_FOR_IDENTIFICATION_ONLY are only valid as input to the ***BioAPI_Capture***, ***BioAPI_Enroll***, and ***BioAPI_Import*** functions.

d) The BioAPI_PURPOSE_AUDIT purpose is not valid as input to any function, but is only used in the BIR header.

© Standards Australia Limited 2019

Documents
Document Sharing Hub

Full document details: AS ISO IEC 19784.1-2019

e) The ***BioAPI_Process, BioAPI_CreateTemplate***, ***BioAPI_ProcessWithAuxBIR*** (BioAPI 2.1 or less), ***BioAPI_ProcessUsingAuxBIRs*** (BioAPI 2.2 or greater), ***BioAPI_Decide*** (BioAPI 2.2 or greater)***, and BioAPI_Fuse*** (BioAPI 2.2 or greater) functions do not have Purpose as an input parameter, but read the Purpose field from the BIR header of the input *CapturedBIR*.

f) The ***BioAPI_Process*** and ***BioAPI_ProcessUsingAuxBIRs*** (BioAPI 2.2 or greater) function may accept as input any intermediate BIR with a Purpose of BioAPI_PURPOSE_VERIFY or BioAPI_PURPOSE_IDENTIFY, and shall output only BIRs with the same purpose as the input BIR.

g) The ***BioAPI_CreateTemplate*** function may accept as input any intermediate BIR with a Purpose of BioAPI_PURPOSE_ENROLL, BioAPI_PURPOSE_ENROLL_FOR_VERIFICATION_ONLY, or BioAPI_PURPOSE_ENROLL_FOR_IDENTFICATION_ONLY, and shall output only BIRs with the same Purpose as the input BIR.

h) If a BIR is suitable for enrollment for either subsequent verification or subsequent identification, then the output BIR shall have a Purpose of BioAPI_PURPOSE_ENROLL.

i) All score BIRs and decision BIRs must have the Decide purpose. Biographic BIRs may have any purpose. No other types of BIRs may have the Decide purpose.

7.17 BioAPI_BIR_SECURITY_BLOCK_FORMAT

Defines the format of the data contained within the security block (SB) of the BioAPI BIR (SecurityBlock element).

```
typedef struct bioapi_bir_security_block_format {
   uint16_t SecurityFormatOwner;
   uint16_t SecurityFormatType;
} BioAPI_BIR_SECURITY_BLOCK_FORMAT;
```

If neither the 'encrypt' or 'signed' flag are set within the BioAPI_BIR_DATA_TYPE field of the BIR Header, then the SecurityFormatOwner and SecurityFormatType are each set to 0x0000 and no security block is present.

SecurityFormatOwner values are assigned and registered by the CBEFF Registration Authority as Biometric Organization identifiers. Security Format Type is assigned by the Security Format Owner (the Biometric Organization) and may optionally be registered.

NOTE 1 Security Format IDs are analogous to Format IDs and the registration process is the same. A single vendor can register for one Format/Product/Security Owner value (a Biometric Organization identifier) that can be used in all associated fields.

NOTE 2 The content of the Security Block itself can include a digital signature or message authentication code (calculated on the BIR Header + BDB), BDB encryption parameters (e.g., encryption algorithm, key length), and/or BIR integrity parameters (e.g., algorithm ID, keyname, version).

NOTE 3 The BioAPI BIR Security Block Format in a BioAPI BIR header corresponds to the "CBEFF_SB_format_owner" and "CBEFF_SB_format_type" in ISO/IEC 19785-1.

7.18 BioAPI_BIR_SUBTYPE (BioAPI 2.0)

This subclause applies only when the BioAPI version number in use is 2.0.

7.18.1 This identifies a subtype within the BDB type (specified in the BioAPI_BIR_BIOMETRIC_TYPE). Its values and their meaning are defined by the specifier of that BDB type.

7.18.2 Each of the feature flags BioAPI_BIR_SUBTYPE_LEFT and BioAPI_BIR_SUBTYPE_RIGHT may or may not be set (zero, one, or both).

7.18.3 Either none or exactly one of the five finger subtypes may be set.

© Standards Australia Limited 2019

Documents Document Sharing Hub

Full document details: AS ISO IEC 19784.1-2019

```
typedef uint8_t BioAPI_BIR_SUBTYPE;

   #define BioAPI_BIR_SUBTYPE_LEFT             (0x01)
   #define BioAPI_BIR_SUBTYPE_RIGHT            (0x02)
   #define BioAPI_BIR_SUBTYPE_THUMB            (0x04)
   #define BioAPI_BIR_SUBTYPE_POINTERFINGER    (0x08)
   #define BioAPI_BIR_SUBTYPE_MIDDLEFINGER     (0x10)
   #define BioAPI_BIR_SUBTYPE_RINGFINGER       (0x20)
   #define BioAPI_BIR_SUBTYPE_LITTLEFINGER     (0x40)
   #define BioAPI_BIR_SUBTYPE_MULTIPLE         (0x80)
   #define BioAPI_NO_SUBTYPE_AVAILABLE         (0x00)
```

NOTE 1 The condition NO VALUE AVAILABLE is indicated by setting the value to zero. BIRs that are not originally created by a BioAPI BSP but have been transformed from another data format and for which subtype information is not available may use this value.

NOTE 2 The BioAPI BIR Subtype corresponds to the "CBEFF_BDB_biometric_subtype" in ISO/IEC 19785-1.

NOTE 3 This structure is primarily used within a BIR header; however, it is also used as an input parameter for functions that capture biometric data. The BioAPI_NO_SUBTYPE_AVAILABLE value is used in the BIR header when either this field is not applicable or information is not available. BioAPI_NO_SUBTYPE_AVAILABLE is also used as a function parameter when the application allows the BSP to determine which subtype is to be captured.

7.19 BioAPI_BIR_SUBTYPE (BioAPI 2.1)

7.19.1 This subclause applies only when the BioAPI version number in use is 2.1 or greater.

7.19.2 This identifies a subtype within the BDB type (specified in the BioAPI_BIR_BIOMETRIC_TYPE). Its values and their meaning are defined by the specifier of that BDB type.

NOTE In the BioAPI 2.1 definition of this type, multiple bits can be set. This definition is aligned with ISO/IEC 19785-1.

```
typedef uint8_t BioAPI_BIR_SUBTYPE;

   #define BioAPI_BIR_SUBTYPE_VEIN_ONLY_MASK   (0x80)
   #define BioAPI_BIR_SUBTYPE_LEFT_MASK        (0x01)
   #define BioAPI_BIR_SUBTYPE_RIGHT_MASK       (0x02)
   #define BioAPI_BIR_SUBTYPE_THUMB            (0x04)
   #define BioAPI_BIR_SUBTYPE_POINTERFINGER    (0x08)
   #define BioAPI_BIR_SUBTYPE_MIDDLEFINGER     (0x10)
   #define BioAPI_BIR_SUBTYPE_RINGFINGER       (0x20)
   #define BioAPI_BIR_SUBTYPE_LITTLEFINGER     (0x40)
   #define BioAPI_BIR_SUBTYPE_VEIN_PALM        (0x04)
   #define BioAPI_BIR_SUBTYPE_VEIN_BACKOFHAND  (0x08)
   #define BioAPI_BIR_SUBTYPE_VEIN_WRIST       (0x10)
   #define BioAPI_NO_SUBTYPE_AVAILABLE         (0x00)
```

7.19.3 This structure is a bitmask, with the bits defined as shown below. Bit 7 is used to indicate the interpretation of the lower-order bits. Bit positions zero (0) and one (1) always indicate left and right respectively. When bit position 7 is not set, these bit positions may apply to any biometric type; however, bit positions 2-6 are specific to the finger and vein biometric types. When bit position 7 is set, then the remaining bit positions apply to the vein biometric type only.

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0 (Any)	Little	Ring	Middle	Pointer	Thumb	Right	Left
1 (Vein only)	Reserved	Reserved	Wrist	Back of Hand	Palm	Right	Left

NOTE Vein biometric data can be obtained from the fingers (Bit 7 set to zero), or from the wrists, palm, or backs of either hand (Bit 7 set to 1).

© Standards Australia Limited 2019

Documents
Document Sharing Hub

Full document details: AS ISO IEC 19784.1-2019

7.19.4 Zero or more bits may be set. The abstract value NO VALUE AVAILABLE is indicated by setting all bits to zero.

NOTE The abstract value NO VALUE AVAILABLE can be used with BIRs that are not originally created by a BioAPI BSP but have been transformed from another data format. It can also be used when none of the other values are applicable or information is not available.

7.19.5 The bit positions BioAPI_BIR_SUBTYPE_THUMB through BioAPI_BIR_SUBTYPE_LITTLEFINGER can be used to identify the instance(s) of a biometric type related to the fingers.

NOTE The BioAPI BIR Subtype corresponds to the "CBEFF_BDB_biometric_subtype" in ISO/IEC 19785-1.

7.19.6 If one or more of the bit positions BioAPI_BIR_SUBTYPE_THUMB through BioAPI_BIR_SUBTYPE_LITTLEFINGER are set, then either or both the bit positions BioAPI_BIR_SUBTYPE_LEFT_MASK and BioAPI_BIR_SUBTYPE_RIGHT_MASK shall also be set. When only the former is set, the subtype value designates one or more fingers of the left hand. When only the latter is set, the subtype value designates one or more fingers of the right hand. When both are set, the value designates one or more fingers of both hands (the same finger locations in both).

NOTE It is not possible to designate, for example, a set of fingers consisting of the index finger of the left hand and the medium finger of the right hand.

7.19.7 When a value with multiple finger bits set occurs in a BIR header (or is used as an input parameter of a BioAPI function that performs capture), it is not strictly required that all the specified finger instances be actually present in the BDB of the BIR (or be actually captured). However, it is recommended that all the designated finger instances be present in the BDB except in the case of missing fingers and similar exceptional situations. Likewise, if the subtype field of a BIR designates multiple fingers, it is acceptable for the BDB of that BIR to include data about an extra finger if the subject happens to have six fingers in his or her hand.

7.19.8 If none of the finger bit positions BioAPI_BIR_SUBTYPE_THUMB through BioAPI_BIR_SUBTYPE_LITTLEFINGER are set, then the bit positions BioAPI_BIR_SUBTYPE_LEFT_MASK and BioAPI_BIR_SUBTYPE_RIGHT_MASK can be used to identify an instance of a biometric type for which there is one left instance and one right instance (for example, iris), or also the single instance of a biometric type for which there is only one instance (for example, face).

7.19.9 Vein data taken from one or more fingers is indicated by setting bit position 7 to zero, setting one or both of bit positions 0 and 1, and setting one or more of the finger bit positions (3-7). Vein data taken from one of the other sources (wrist, palm, or back of hand) is indicated by setting bit position 7 to one, setting one or both of bit positions 0 and 1, and setting one of bit positions 3-5 as appropriate).

NOTE 1 BIRs that are not originally created by a BioAPI BSP but have been transformed from another data format and for which subtype information is not available can use the NO VALUE AVAILABLE condition.

NOTE 2 The BioAPI BIR Subtype corresponds to the "CBEFF_BDB_biometric_subtype" in ISO/IEC 19785-1.

NOTE 3 This structure is primarily used within a BIR header; however, it is also used as an input parameter for functions that capture biometric data. The BioAPI_NO_SUBTYPE_AVAILABLE value is used in the BIR header when subtype information is not available. BioAPI_NO_SUBTYPE_AVAILABLE is also used as a function parameter when the application allows the BSP to determine which subtype is to be captured.

7.19.10 The use of the bit position BioAPI_BIR_SUBTYPE_MULTIPLE is not recommended when the BioAPI version number in use is 2.1 or greater.

7.20 BioAPI_BIR_SUBTYPE_MASK (BioAPI 2.1)

This subclause applies only when the BioAPI version number in use is 2.1 or greater.

The type BioAPI_BIR_SUBTYPE_MASK is defined as follows.

© Standards Australia Limited 2019

Documents Document Sharing Hub

Full document details: AS ISO IEC 19784.1-2019

```
typedef uint32_t BioAPI_BIR_SUBTYPE_MASK;

      #define BioAPI_BIR_SUBTYPE_LEFT_BIT                  (0x0001)
      #define BioAPI_BIR_SUBTYPE_RIGHT_BIT                 (0x0002)
      #define BioAPI_BIR_SUBTYPE_LEFT_THUMB_BIT            (0x0004)
      #define BioAPI_BIR_SUBTYPE_LEFT_POINTERFINGER_BIT    (0x0008)
      #define BioAPI_BIR_SUBTYPE_LEFT_MIDDLEFINGER_BIT     (0x0010)
      #define BioAPI_BIR_SUBTYPE_LEFT_RINGFINGER_BIT       (0x0020)
      #define BioAPI_BIR_SUBTYPE_LEFT_LITTLEFINGER_BIT     (0x0040)
      #define BioAPI_BIR_SUBTYPE_RIGHT_THUMB_BIT           (0x0080)
      #define BioAPI_BIR_SUBTYPE_RIGHT_POINTERFINGER_BIT   (0x0100)
      #define BioAPI_BIR_SUBTYPE_RIGHT_MIDDLEFINGER_BIT    (0x0200)
      #define BioAPI_BIR_SUBTYPE_RIGHT_RINGFINGER_BIT      (0x0400)
      #define BioAPI_BIR_SUBTYPE_RIGHT_LITTLEFINGER_BIT    (0x0800)
      #define BioAPI_BIR_SUBTYPE_LEFT_VEIN_PALM_BIT        (0x00001000)
      #define BioAPI_BIR_SUBTYPE_LEFT_VEIN_BACKOFHAND_BIT  (0x00002000)
      #define BioAPI_BIR_SUBTYPE_LEFT_VEIN_WRIST_BIT       (0x00004000)
      #define BioAPI_BIR_SUBTYPE_RIGHT_VEIN_PALM_BIT       (0x00008000)
      #define BioAPI_BIR_SUBTYPE_RIGHT_VEIN_BACKOFHAND_BIT (0x00010000)
      #define BioAPI_BIR_SUBTYPE_RIGHT_VEIN_WRIST_BIT      (0x00020000)
```

NOTE 1 The VEIN values are not present in BioAPI 2.0.

The bit positions BioAPI_BIR_SUBTYPE_LEFT_BIT and BioAPI_BIR_SUBTYPE_RIGHT_BIT can be used to identify the instance of a biometric modality for which there is one "left" instance and one "right" instance.

NOTE 2 Iris and hand geometry are examples of such modalities.

The other bit positions (from BioAPI_BIR_SUBTYPE_LEFT_THUMB_BIT through BioAPI_BIR_SUBTYPE_RIGHT_LITTLEFINGER_BIT) can be used to identify the instance of a biometric modality related to the fingers.

NOTE 3 Fingerprint is one such modality.

The value zero (no bit positions set) can be used with biometric modalities for which there is only one instance.

NOTE 4 Signature/sign is one such modality.

7.21 BioAPI_BOOL

This data type is used to indicate a true or false condition.

```
typedef uint8_t BioAPI_BOOL;

   #define BioAPI_FALSE     (0)
   #define BioAPI_TRUE      (!BioAPI_FALSE)
```

7.22 BioAPI_BSP_SCHEMA (BioAPI 2.0)

This subclause applies only when the BioAPI version number in use is 2.0.

7.22.1 Information about a BSP, maintained in the component registry.

```
typedef struct bioapi_bsp_schema {
       BioAPI_UUID BSPUuid;
       BioAPI_STRING BSPDescription;
       uint8_t *Path;
       BioAPI_VERSION SpecVersion;
       BioAPI_STRING ProductVersion;
       BioAPI_STRING Vendor;
       BioAPI_BIR_BIOMETRIC_DATA_FORMAT *BSPSupportedFormats;
       uint32_t NumSupportedFormats;
       BioAPI_BIR_BIOMETRIC_TYPE FactorsMask;
       BioAPI_OPERATIONS_MASK Operations;
```

© Standards Australia Limited 2019

Documents
Document Sharing Hub

Full document details: AS ISO IEC 19784.1-2019

```
        BioAPI_OPTIONS_MASK Options;
        BioAPI_FMR PayloadPolicy;
        uint32_t MaxPayloadSize;
        int32_t DefaultVerifyTimeout;
        int32_t DefaultIdentifyTimeout;
        int32_t DefaultCaptureTimeout;
        int32_t DefaultEnrollTimeout;
        int32_t  DefaultCalibrateTimeout;
        uint32_t MaxBSPDbSize;
        uint32_t MaxIdentify;
}BioAPI_BSP_SCHEMA;
```

7.22.2 Definitions

— *BSPUuid* – UUID of the BSP.

— *BSPDescription* – A NUL-terminated string containing a text description of the BSP.

— *Path* – A pointer to a NUL-terminated string containing the path of the file containing the BSP executable code, including the filename. The path may be a URL. This string shall consist of ISO/IEC 10646 characters encoded in UTF-8 (see ISO/IEC 10646:2017, Annex D).

NOTE 1 When BioAPI_BSP_SCHEMA is used within a function call, the component that receives the call allocates the memory for the *Path* schema element and the calling component frees the memory.

— *SpecVersion* – Major/minor version number of the BioAPI specification to which the BSP was implemented.

— *ProductVersion* – The version string of the BSP software.

— *Vendor* – A NUL-terminated string containing the name of the BSP vendor.

— *BSPSupportedFormats* – A pointer to an array of BioAPI_BIR_BIOMETRIC_DATA_FORMAT structures specifying the supported BDB formats.

— *NumSupportedFormats* – Number of supported formats contained in BSPSupportedFormats.

— *FactorsMask*– A mask which indicates which biometric types are supported by the BSP.

— *Operations* – A mask which indicates which operations are supported by the BSP.

— *Options* – A mask which indicates which options are supported by the BSP.

— *PayloadPolicy* – Threshold setting (maximum FMR value) used to determine when to release the payload after successful verification.

— *MaxPayloadSize* – Maximum payload size (in bytes) that the BSP can accept.

— *DefaultVerifyTimeout* – Default timeout value in milliseconds used by the BSP for *BioAPI_Verify* operations when no timeout is specified by the application.

— *DefaultIdentifyTimeout* – Default timeout value in milliseconds used by the BSP for *BioAPI_Identify* and *BioAPI_IdentifyMatch* operations when no timeout is specified by the application.

— *DefaultCaptureTimeout* - Default timeout value in milliseconds used by the BSP for *BioAPI_Capture* operations when no timeout is specified by the application.

— *DefaultEnrollTimeout* - Default timeout value in milliseconds used by the BSP for *BioAPI_Enroll* operations when no timeout is specified by the application.

— *DefaultCalibrateTimeout* - Default timeout value in milliseconds used by the BSP for sensor calibration operations when no timeout is specified by the application.

— *MaxBSPDbSize* – Maximum size of a BSP-controlled BIR database.

NOTE 2 Applies only when a BSP is only capable of directly managing a single archive unit.

© Standards Australia Limited 2019

Documents Document Sharing Hub

Full document details: AS ISO IEC 19784.1-2019

NOTE 3 A value of zero means that no information about the database size is being provided for one of the following three reasons:

a) databases are not supported,

b) it is capable of managing multiple units (either directly or through a BFP interface), each of which may have a different "maximum size" and information about these units will be provided as part of the insert notification (part of Unit Schema), or

c) one archive unit is supported, but the information is not given here – it will be provided in the insert notification.

— *MaxIdentify* – Largest population supported by the identify function. Unlimited = FFFFFFFF.

7.22.3 See clause 10.1.2 and 10.2.1 for further explanation of schema elements and for BSP insertion of information into the component registry.

7.23 BioAPI_BSP_SCHEMA (BioAPI 2.1)

7.23.1 This subclause applies only when the BioAPI version number in use is 2.1 or greater.

7.23.2 Information about a BSP, maintained in the component registry.

```
typedef struct bioapi_bsp_schema {
   BioAPI_UUID BSPUuid;
   BioAPI_STRING BSPDescription;
   uint8_t *Path;
   BioAPI_VERSION SpecVersion;
   BioAPI_STRING ProductVersion;
   BioAPI_STRING Vendor;
   BioAPI_BIR_BIOMETRIC_DATA_FORMAT *BSPSupportedFormats;
   uint32_t NumSupportedFormats;
   BioAPI_BIR_BIOMETRIC_TYPE FactorsMask;
   BioAPI_OPERATIONS_MASK Operations;
   BioAPI_OPTIONS_MASK Options;
   BioAPI_FMR PayloadPolicy;
   uint32_t MaxPayloadSize;
   int32_t DefaultVerifyTimeout;
   int32_t DefaultIdentifyTimeout;
   int32_t DefaultCaptureTimeout;
   int32_t DefaultEnrollTimeout;
   int32_t DefaultCalibrateTimeout;
   uint32_t MaxBSPDbSize;
   uint32_t MaxIdentify;
   uint32_t MaxNumEnrollInstances;
   uint8_t  *HostingEndpointIRI;
   BioAPI_UUID BSPAccessUUID;
}BioAPI_BSP_SCHEMA;
```

7.23.3 Definitions

— *BSPUuid* – UUID of the BSP.

— *BSPDescription* – A NUL-terminated string containing a text description of the BSP.

— *Path* – A pointer to a NUL-terminated string containing the path of the file containing the BSP executable code, including the filename. The path may be a URL. This string shall consist of ISO/IEC 10646 characters encoded in UTF-8 (see ISO/IEC 10646:2017, Annex D).

NOTE When BioAPI_BSP_SCHEMA is used within a function call, the component that receives the call allocates the memory for the *Path* schema element and the calling component frees the memory.

— *SpecVersion* – Major/minor version number of the BioAPI specification to which the BSP was implemented.

© Standards Australia Limited 2019

Documents
Document Sharing Hub

Full document details: AS ISO IEC 19784.1-2019

— *ProductVersion* – The version string of the BSP software.

— *Vendor* – A NUL-terminated string containing the name of the BSP vendor.

— *BSPSupportedFormats* – A pointer to an array of BioAPI_BIR_BIOMETRIC_DATA_FORMAT structures specifying the supported BDB formats.

— *NumSupportedFormats* – Number of supported formats contained in BSPSupportedFormats.

— *FactorsMask* – A mask which indicates which biometric types are supported by the BSP.

— *Operations* – A mask which indicates which operations are supported by the BSP.

— *Options* – A mask which indicates which options are supported by the BSP.

— *PayloadPolicy* – Threshold setting (maximum FMR value) used to determine when to release the payload after successful verification.

— *MaxPayloadSize* – Maximum payload size (in bytes) that the BSP can accept.

— *DefaultVerifyTimeout* – Default timeout value in milliseconds used by the BSP for ***BioAPI_Verify*** operations when no timeout is specified by the application.

— *DefaultIdentifyTimeout* – Default timeout value in milliseconds used by the BSP for ***BioAPI_Identify*** and ***BioAPI_IdentifyMatch*** operations when no timeout is specified by the application.

— *DefaultCaptureTimeout* – Default timeout value in milliseconds used by the BSP for ***BioAPI_Capture*** operations when no timeout is specified by the application.

— *DefaultEnrollTimeout* – Default timeout value in milliseconds used by the BSP for ***BioAPI_Enroll*** operations when no timeout is specified by the application.

— *DefaultCalibrateTimeout* – Default timeout value in milliseconds used by the BSP for sensor calibration operations when no timeout is specified by the application.

— *MaxBSPDbSize* – Maximum size of a BSP-controlled BIR database.

NOTE 2 Applies only when a BSP is only capable of directly managing a single archive unit.

NOTE 3 A value of zero means that no information about the database size is being provided for one of the following three reasons:

a) databases are not supported;

b) it is capable of managing multiple units (either directly or through a BFP interface), each of which may have a different "maximum size" and information about these units will be provided as part of the insert notification (part of Unit Schema); or

c) one archive unit is supported, but the information is not given here – it will be provided in the insert notification.

— *MaxIdentify* – Largest population supported by the identify function. Unlimited = FFFFFFFF.

— *MaxNumEnrollInstances* – The maximum number of distinct instances that a BSP can create reference templates for in one enroll operation. This information can be useful to an application that uses the application-controlled GUI feature.

— *HostingEndpointIRI* – An IRI identifying the framework whose component registry contains a registration of the BSP. This parameter shall be ignored by frameworks conforming to this document and shall be set to NULL by an application. It is provided to support interworking standards, which may specify the use of identical BSPs present on multiple computers from within an application running on the same or a different computer.

Documents
Document Sharing Hub
© Standards Australia Limited 2019

Full document details: AS ISO IEC 19784.1-2019

— *BSPAccessUUID* – A UUID, unique within the scope of an application, which the application may use to refer to the BSP as an alternative to the BSP product UUID. This parameter shall be ignored by frameworks conforming to this document and can be set to any UUID value by an application. It is provided to support interworking standards, which may specify the use of identical BSPs present on multiple computers from within an application running on the same or a different computer.

NOTE 4 The "BSPAccess_UUID" and the "HostingendpointIRI" are part of the definition of the C type **BioAPI_BSP_SCHEMA**, but are not part of the BSP schema information stored in the component registry (see Table 3).

7.23.4 See subclause 10.1.2 and 10.2.1 for further explanation of schema elements and for BSP insertion of information into the component registry.

7.24 BioAPI_CANDIDATE

One of a set of candidates returned by ***BioAPI_Identify*** or ***BioAPI_IdentifyMatch*** indicating a successful match.

```
typedef struct bioapi_candidate {
 BioAPI_IDENTIFY_POPULATION_TYPE Type;
 union {
     BioAPI_UUID *BIRInDataBase;
     uint32_t *BIRInArray;
 } BIR;
 BioAPI_FMR  FMRAchieved;
} BioAPI_CANDIDATE;
```

7.25 BioAPI_CATEGORY

This bitmask describes the category of BFP or BioAPI Unit. A BFP or BioAPI Unit shall belong to exactly one category.

```
typedef uint32_t BioAPI_CATEGORY;

#define BioAPI_CATEGORY_ARCHIVE           (0x00000001)
#define BioAPI_CATEGORY_MATCHING_ALG      (0x00000002)
#define BioAPI_CATEGORY_PROCESSING_ALG    (0x00000004)
#define BioAPI_CATEGORY_SENSOR            (0x00000008)
```

The highest significant bit is reserved and does not indicate a category of BFP or BioAPI Unit.

7.26 BioAPI_DATA

7.26.1 The BioAPI_DATA structure is used to associate a length, in bytes, with the address of an arbitrary block of contiguous memory.

```
typedef struct bioapi_data{
    uint32_t Length; /* in bytes */
    void *Data;
} BioAPI_DATA;
```

7.26.2 Definitions

— *Length* – Length of the data buffer in bytes.

— *Data* – A pointer to the start of an arbitrary length data buffer.

7.27 BioAPI_DATE

Defines the date when the BIR was created or when it is no longer valid.

```
typedef struct bioapi_date {
    uint16_t Year; /* valid range: 1900 - 9999 */
```

© Standards Australia Limited 2019

Documents
Document Sharing Hub

Full document details: AS ISO IEC 19784.1-2019

```
    uint8_t Month; /* valid range: 01 - 12 */
    uint8_t Day; /* valid range: 01 - 31, consistent with associated month/year */
} BioAPI_DATE;
    #define BioAPI_NO_YEAR_AVAILABLE    (0)
    #define BioAPI_NO_MONTH_AVAILABLE   (0)
    #define BioAPI_NO_DAY_AVAILABLE     (0)
```

The condition NO VALUE AVAILABLE is indicated by setting all fields to zero. When used within the Creation DTG within a BIR header, if no date information is available, then the NO_DATE_AVAILABLE value shall be used.

NOTE 1 The year 2000 AD is represented by a Year value of 2000.

NOTE 2 When used for ExpirationDate in a BIR header, corresponds to the "Valid to" portion of the "CBEFF_BDB_validity_period" in ISO/IEC 19785-1.

NOTE 3 Date formats are consistent with ISO 8601[2].

7.28 BioAPI_DB_ACCESS_TYPE

This bitmask describes a biometric application's desired level of access to a BSP-controlled BIR database. The BSP may use the mask to determine what lock to obtain on the BIR database.

```
typedef uint32_t BioAPI_DB_ACCESS_TYPE;
    #define BioAPI_DB_ACCESS_READ       (0x00000001)
    #define BioAPI_DB_ACCESS_WRITE      (0x00000002)
```

7.29 BioAPI_DB_MARKER_HANDLE

A handle to a BIR database marker.

A marker is an internal (not standardized) data structure managed by a BSP, which dynamically points to a record in an open BSP-controlled BIR database. A marker handle is created and returned to a biometric application by the functions ***BioAPI_DbOpen*** and ***BioAPI_DbGetBIR***. The internal state of the marker includes the open database handle and also the position of a record in that open database. All markers (and their handles) held by a biometric application to an open BIR database become invalid when the database is closed by the biometric application.

```
typedef uint32_t BioAPI_DB_MARKER_HANDLE;
```

7.30 BioAPI_DB_HANDLE

A handle to an open BIR database initially provided by a BSP to a biometric application and later used by that application to address the database through the BSP.

```
typedef int32_t BioAPI_DB_HANDLE;

   #define BioAPI_DB_INVALID_HANDLE     (-1)
   #define BioAPI_DB_DEFAULT_HANDLE     (0)
   #define BioAPI_DB_DEFAULT_UUID_PTR   (NULL)
```

7.31 BioAPI_DBBIR_ID

A structure providing the handle to a BIR database controlled by a BSP and the UUID of a BIR in that database.

```
typedef struct bioapi_dbbir_id {
   BioAPI_DB_HANDLE DbHandle;
   BioAPI_UUID KeyValue;
} BioAPI_DBBIR_ID;
```

NOTE This type is used as an element of BioAPI_INPUT_BIR.

© Standards Australia Limited 2019

Documents Document Sharing Hub

Full document details: AS ISO IEC 19784.1-2019

7.32 BioAPI_DTG

Defines the date and time when the BIR was created.

```
typedef struct bioapi_DTG {
       BioAPI_DATE Date;
       BioAPI_TIME Time;
} BioAPI_DTG;
```

NOTE The BioAPI DTG in a BIR header corresponds to the "CBEFF_creation_date" in ISO/IEC 19785-1.

7.33 BioAPI_ENCRYPTION_ALG (BioAPI 2.2)

Identifies encryption algorithm supported by a BioAPI Unit. This BioAPI type shall be the XML value notation of ASN.1 identifier (see ISO/IEC 8824-1) assigned to encryption algorithms. Example to specify AES with 128 bit keys in CBC mode, the char would be "2.16.840.1.101.3.4.1.2" which is the XML value notation for.

```
typedef char *BioAPI_ENCRYPTION_ALG;

   #define BioAPI_ENCRYPTION_ALG_NOT_SUPPORTED NULL
```

7.34 BioAPI_ENCRYPTION_INFO (BioAPI 2.2)

7.34.1 Structure giving information of the cryptographic algorithm and key(s) of a BioAPI Unit or a biometric application which is used to encrypt/decrypt biometric data

```
typedef struct bioapi_encryption_info {
   BioAPI_ENCRYPTION_ALG ENCAlg;
   BioAPI_KEY_INFO *ENCKeyInfo;
   uint32_t *NumberOfElementsEncKeyInfo;
} BioAPI_ENCRYPTION_INFO;
```

7.34.2 Definitions

— *ENCAlg* – Encryption algorithm.

— *ENCKeyInfo* – An array of key information for encryption.

— *NumberOfElementsEncKeyInfo* – The number of elements in the array of EncKeyInfo.

7.35 BioAPI_ERROR_INFO (BioAPI 2.1)

This subclause applies only when the BioAPI version number in use is 2.1 or greater.

This structure is used by the function ***BioAPI_GetLastErrorInfo***, and holds information about the most recent error condition that caused a BioAPI function call to return an error code.

```
typedef struct bioapi_error_info {
   int32_t ErrorCode;
   BioAPI_STRING ErrorMessage;
   BioAPI_STRING ErrorSourceName;
} BioAPI_ERROR_INFO;
```

The member ErrorCode is the same value that was returned by the last BioAPI function call that returned an error code. The value of the remaining members are not standardized.

© Standards Australia Limited 2019

Documents
Document Sharing Hub

Full document details: AS ISO IEC 19784.1-2019

7.36 BioAPI_EVENT

This enumeration defines the event types that can be raised by a BSP, BFP, or BioAPI Unit. Biometric applications can define event handling callback functions of type *BioAPI_EventHandler* to receive and manage these events. Callback functions are registered using the ***BioAPI_BSPLoad*** function. Example events include the addition (insertion) or removal of a biometric sensor. Events are asynchronous.

BioAPI_NOTIFY_SOURCE_PRESENT and BioAPI_NOTIFY_SOURCE_REMOVED events are generated by devices (sensor units) that can detect when the user may be available to provide a biometric sample (e.g., the user has placed a finger on a fingerprint device). BioAPI_NOTIFY_SOURCE_PRESENT indicates that a sample may be available, while BioAPI_NOTIFY_SOURCE_REMOVED indicates that a sample is probably no longer available. There is no requirement that these events must occur in pairs; several BioAPI_NOTIFY_SOURCE_PRESENT events may occur in succession.

```
typedef uint32_t BioAPI_EVENT;

   #define BioAPI_NOTIFY_INSERT          (1)
   #define BioAPI_NOTIFY_REMOVE          (2)
   #define BioAPI_NOTIFY_FAULT           (3)
   #define BioAPI_NOTIFY_SOURCE_PRESENT  (4)
   #define BioAPI_NOTIFY_SOURCE_REMOVED  (5)
```

7.37 BioAPI_EVENT_MASK

This type defines a mask with bit positions for each event type. The mask is used to enable/disable event notification, and to indicate what events are supported by a BSP.

```
typedef uint32_t BioAPI_EVENT_MASK;

   #define BioAPI_NOTIFY_INSERT_BIT          (0x00000001)
   #define BioAPI_NOTIFY_REMOVE_BIT          (0x00000002)
   #define BioAPI_NOTIFY_FAULT_BIT           (0x00000004)
   #define BioAPI_NOTIFY_SOURCE_PRESENT_BIT  (0x00000008)
   #define BioAPI_NOTIFY_SOURCE_REMOVED_BIT  (0x00000010)
```

When the BioAPI version number in use is 2.0, it can be impossible to mask an INSERT event coming from an attach session of a BSP, because the event can occur just after a ***BioAPI_BSPLoad*** call, before any ***BioAPI_EnableEvents*** call has had any chance to be processed. This is because ***BioAPI_EnableEvents*** requires a handle which is provided by ***BioAPI_BSPAttach***/***BioAPI_BSPAttachSecure*** (BioAPI 2.2 or greater), and ***BioAPI_BSPAttach***/***BioAPI_BSPAttachSecure*** (BioAPI 2.2 or greater) itself shall follow ***BioAPI_BSPLoad***. An INSERT event will be raised by the BSP on the ***BioAPI_BSPLoad*** call if a BioAPI Unit is already "inserted", and this event will reach the application before the application can call ***BioAPI_EnableEvents***.

7.38 BioAPI_EventHandler

7.38.1 This is the event handler interface that an application is required to support if it wishes to receive asynchronous notification of events such as the availability of a biometric sample or a fault detected by a BSP. The event handler function is registered with the BioAPI Framework as part of the ***BioAPI_BSPLoad*** function. This is the caller's event handler for all general BSP events over all of the caller's attach sessions with the loaded BSP. General event notifications are processed through the BioAPI Framework.

The event handler function and any functions invoked (directly or indirectly) by that function shall not issue BioAPI calls. These circular calls may result in deadlock in numerous situations; hence the event handler shall be implemented without using BioAPI services (however, enumeration functions can be used at any time).

The BioAPI_EventHandler can be invoked multiple times in response to a single event occurring in the associated BSP. The event handler and the calling application shall track receipt of event notifications and ignore duplicates. An event notification provides the following information:

Documents
Document Sharing Hub
© Standards Australia Limited 2019

Full document details: AS ISO IEC 19784.1-2019

```
typedef BioAPI_RETURN (BioAPI *BioAPI_EventHandler)
       (const BioAPI_UUID *BSPUuid,
       BioAPI_UNIT_ID UnitID,
       void* AppNotifyCallbackCtx,
       const BioAPI_UNIT_SCHEMA *UnitSchema,
       BioAPI_EVENT EventType);
```

7.38.2 Definitions

— *BSPUuid (input) – The UUID of the BSP raising the event.*

— *UnitID (input) – The unit ID of the BioAPI Unit associated with the event.*

— *AppNotifyCallbackCtx) – A generic pointer to context information that was provided in the call to BioAPI_BSPLoad that established the event handler.*

— *UnitSchema (input) – A pointer to the unit schema of the BioAPI Unit associated with the event.*

— *EventType (input) – The BioAPI_EVENT that has occurred.*

If the *EventType* is BioAPI_NOTIFY_INSERT, then a unit schema shall be provided (that is, *UnitSchema* shall point to a variable of type BioAPI_UNIT_SCHEMA). Otherwise, *UnitSchema* shall be NULL.

When the application receives a call to an event handler that carries a unit schema, the application shall not call ***BioAPI_Free*** to free the memory block containing the unit schema or a memory block pointed to by any of its members.

7.39 BioAPI_FMR

A 32-bit integer value (N) that indicates a probable False Match Rate of $N/(2^{31}-1)$. The larger the value, the worse the result. Negative values are used to signal exceptional conditions. The only negative value currently defined is minus one.

```
typedef int32_t BioAPI_FMR;

   #define BioAPI_NOT_SET     (-1)
```

NOTE FMR is used within BioAPI as a means of setting thresholds and returning scores (see clause C.4).

7.40 BioAPI_FRAMEWORK_SCHEMA

7.40.1 The framework schema entry for a BioAPI Framework as recorded in the component registry.

```
typedef struct bioapi_framework_schema {
       BioAPI_UUID FrameworkUuid;
       BioAPI_STRING FwDescription;
       uint8_t *Path;
       BioAPI_VERSION SpecVersion;
       BioAPI_STRING ProductVersion;
       BioAPI_STRING Vendor;
       BioAPI_UUID FwPropertyId;
       BioAPI_DATA FwProperty;
   } BioAPI_FRAMEWORK_SCHEMA;
```

7.40.2 Definitions

— *FrameworkUuid* – UUID of the Framework component.

— *FwDescription* – A NUL-terminated string containing a text description of the Framework.

— *Path* – A pointer to a NUL-terminated string containing the path of the file containing the Framework executable code, including the filename. The path may be a URL. This string shall consist of ISO/IEC 10646 characters encoded in UTF-8 (see ISO/IEC 10646:2017, Annex D).

© Standards Australia Limited 2019

Documents
Document Sharing Hub

Full document details: AS ISO IEC 19784.1-2019

NOTE When BioAPI_FRAMEWORK_SCHEMA is used within a function call, the component that receives the call allocates the memory for the *Path* schema element and the calling component frees the memory.

— *SpecVersion* – Major/minor version number of the BioAPI specification to which the Framework was implemented.

— *ProductVersion* – The version string of the Framework software.

— *Vendor* – A NUL-terminated string containing the name of the Framework vendor.

— *FwPropertyID* – UUID of the format of the following Framework property.

— *FwProperty* – Address and length of a memory buffer containing the Framework property. The format and content of the Framework property can either be specified by a vendor or can be specified in a related standard.

7.40.3 See clause 10.1.1 for further explanation of framework schema elements.

7.41 BioAPI_GUI_BITMAP (BioAPI 2.0)

This subclause applies only when the BioAPI version number in use is 2.0.

7.41.1 This structure provides a graphic for display by the application.

```
typedef struct bioapi_gui_bitmap {
   uint32_t Width; /* Width of bitmap in pixels (number of pixels for each line) */
   uint32_t Height; /* Height of bitmap in pixels (number of lines) */
   BioAPI_DATA Bitmap;
} BioAPI_GUI_BITMAP;
```

7.41.2 Definitions

— *Bitmap - a series of 8-bit grayscale pixels (where '00' = black and 'FF' = white), read from left to right, top to bottom, as follows*:

Table 1 — GUI Bitmap Format

Byte Position	Meaning	Description
0	line 0, pixel 0	first pixel of first line
1	line 0, pixel 1	second pixel of first line
...	...	
width-1	line 0, pixel(width-1)	last pixel of first line
width	line 1, pixel 0	first pixel of second line
...	...	
(width * height) - 1	line (width – 1), pixel (height – 1)	last line, last pixel

NOTE This is used with the application-controlled GUI option. See BioAPI_GUI_STATE_CALLBACK and BioAPI_GUI_STREAMING_CALLBACK descriptions under 7.54 and 7.55.

7.42 BioAPI_GUI_BITMAP (BioAPI 2.1)

7.42.1 This subclause applies only when the BioAPI version number in use is 2.1 or greater.

7.42.2 This structure provides a graphic for display by the application.

```
typedef struct bioapi_gui_bitmap {
   BioAPI_BIR_SUBTYPE_MASK SubtypeMask;
       /* The subtype (or subtypes) of the sample represented
       by the bitmap */
```

© Standards Australia Limited 2019

Documents Document Sharing Hub

Full document details: AS ISO IEC 19784.1-2019

```
   uint32_t Width;
       /* Width of bitmap in pixels (number of pixels for each line) */
   uint32_t Height;
       /* Height of bitmap in pixels (number of lines) */
   BioAPI_DATA Bitmap;
} BioAPI_GUI_BITMAP;
```

7.42.3 Definitions

— *Bitmap* – A series of 8-bit grayscale pixels (where '00' = black and 'FF' = white), read from left to right, top to bottom, as follows:

Table 1 — GUI bitmap format

Byte position	Meaning	Description
0	line 0, pixel 0	first pixel of first line
1	line 0, pixel 1	second pixel of first line
...	...	
Width -1	line 0, pixel (**Width** -1)	last pixel of first line
Width	line 1, pixel 0	first pixel of second line
...	...	
(**Width** * **Height**) - 1	line (**Width** - 1), pixel (**Height** - 1)	last line, last pixel

NOTE This is used with the application-controlled GUI option. See BioAPI_GUI_STATE_EVENT_HANDLER and BioAPI_GUI_PROGRESS_EVENT_HANDLER descriptions in 7.86.

7.43 BioAPI_GUI_BITMAP_ARRAY (BioAPI 2.1)

This subclause applies only when the BioAPI version number in use is 2.1 or greater.

The type BioAPI_GUI_BITMAP_ARRAY is defined as follows:

```
typedef struct bioapi_gui_bitmap_array {
   uint32_t           NumberOfMembers;
   BioAPI_GUI_BITMAP      *Bitmaps;
       /* A pointer to an array of BioAPI_GUI_BITMAPs */
} BioAPI_GUI_BITMAP_ARRAY;
```

7.44 BioAPI_GUI_ENROLL_TYPE (BioAPI 2.1)

This subclause applies only when the BioAPI version number in use is 2.1 or greater.

The type BioAPI_GUI_ENROLL_TYPE indicates the enroll type of a BSP, and is defined as follows:

```
typedef uint32_t BioAPI_GUI_ENROLL_TYPE;

   #define BioAPI_GUI ENROLL_TYPE TEST_VERIFY      (0x00000001)
   #define BioAPI_GUI ENROLL_TYPE MULTIPLE_CAPTURE (0x00000002)
```

The enroll type of a BSP denotes a pattern of suboperations performed by the BSP during an enroll operation.

The bit position BioAPI_GUI ENROLL_TYPE TEST_VERIFY indicates, when set, that the BSP supports test verification at enrollment. During an enroll operation, an application receives GUI state event notification callbacks for two capture suboperations, GUI state event notification callbacks for a process suboperation, GUI state event notification callbacks for a create template suboperation, and GUI state event notification callbacks for a verify match suboperation;

The bit position BioAPI_GUI ENROLL_TYPE MULTIPLE_CAPTURE indicates, when set, that the BSP supports multiple capture suboperations at enrollment. During an enroll operation, an application

© Standards Australia Limited 2019

Documents
Document Sharing Hub

Full document details: AS ISO IEC 19784.1-2019